Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

PREMIER MINISTRE

Secrétariat général
de la défense
et de la sécurité nationale

*Agence nationale de la sécurité
des systèmes d'information*

Paris, le 25 mars 2015

N° DAT-24/ANSSI/SDE/NP

Nombre de pages du document
(y compris cette page) : 15

Note technique

Recommandations de configuration matérielle de postes clients et serveurs x86

Public visé:

Développeur	
Administrateur	✓
RSSI	✓
DSI	
Utilisateur	

Informations

Avertissement

Ce document rédigé par l'ANSSI présente les « **Recommandations de configuration matérielle de postes clients et serveurs x86** ». Il est téléchargeable sur le site www.ssi.gouv.fr. Il constitue une production originale de l'ANSSI. Il est à ce titre placé sous le régime de la « Licence ouverte » publiée par la mission Etalab (www.etalab.gouv.fr). Il est par conséquent diffusable sans restriction.

Ces recommandations sont livrées en l'état et adaptées aux menaces au jour de leur publication. Au regard de la diversité des systèmes d'information, l'ANSSI ne peut garantir que ces informations puissent être reprises sans adaptation sur les systèmes d'information cibles. Dans tous les cas, la pertinence de l'implémentation des éléments proposés par l'ANSSI doit être soumise, au préalable, à la validation de l'administrateur du système et/ou des personnes en charge de la sécurité des systèmes d'information.

Personnes ayant contribué à la rédaction de ce document:

Contributeurs	Rédigé par	Approuvé par	Date
LAM, BAS, BSS	BSS	SDE	25 mars 2015

Évolutions du document :

Version	Date	Nature des modifications
1.0	25 mars 2015	Version initiale.

Pour toute remarque:

Contact	Adresse	@mél	Téléphone
Division Assistance Technique	51 bd de La Tour-Maubourg 75700 Paris Cedex 07 SP	communication@ssi.gouv.fr	01 71 75 84 04

Table des matières

1 Glossaire

AES	*Advanced Encryption Standard*, standard de chiffrement avancé.
AES-NI	*Advanced Encryption Standard – New Instructions*, nouvelles instructions – standard de chiffrement avancé.
AMT	*Active Management Technology*, Technologie d'administration active.
ASLR	*Address Space Layout Randomisation*, distribution aléatoire de l'espace d'adressage.
BIOS	*Basic Input/Output System*, système entrée/sortie basique.
DMA	*Direct Memory Access*, accès mémoire direct.
GCM	*Galois Counter Mode*, mode de Galois/compteur.
IOMMU	*Input/Output Memory Management Unit*, unité de gestion de mémoire d'entrée/sortie.
MMU	*Memory Management Unit*, unité de gestion de mémoire.
NX	*No-eXecute*, pas d'exécution.
OS	*Operating System*, système d'exploitation.
PAE	*Physical Address Extension*, extension d'adressage physique.
PIC	*Position Independent Code*, code indépendant de la position.
PIE	*Position Independent Executable*, exécutable indépendant de la position.
SMAP	*Supervisor Mode Access Prevention*, prévention de l'accès en mode superviseur.
SMEP	*Supervisor Mode Execution Prevention*, prévention contre l'exécution en mode superviseur.
SMM	*System Management Mode*, mode de gestion matérielle.
TPM	*Trusted Platform Module*, module de plate-forme de confiance.
UEFI	*Unified Extensible Firmware Interface*, interface micrologicielle extensible unifiée.
XD	*eXecute Disable*, désactivation de l'exécution.

2 Préambule

Les processeurs et composants matériels offrent de plus en plus de fonctionnalités. Cet accroissement fonctionnel va de pair avec une complexité de plus en plus importante du matériel, ce qui affecte globalement la sécurité du système.

Certaines de ces nouvelles fonctionnalités sont susceptibles d'engendrer des vulnérabilités du fait de défauts de spécification claire ou de problèmes d'implémentation. Si de nombreuses fonctions ajoutées au fil du temps constituent une menace au regard des vulnérabilités qu'elles peuvent engendrer, d'autres en revanche peuvent contribuer à améliorer la sécurité des systèmes d'exploitation qui les utilisent.

Ce document est constitué de deux parties :
- le premier chapitre traite essentiellement de mécanismes matériels présents ou prévus sur architecture x86. Certains sont accompagnés de recommandations quant à leur usage ;
- le deuxième chapitre aborde des éléments de configuration qu'il convient de vérifier lors du paramètrage du BIOS d'un ordinateur.

Les recommandations émises dans cette note technique, particulièrement celles associées à l'activation de fonctions matérielles, sont établies à l'état de l'art des connaissances sur leur sécurité et leurs vulnérabilités. Bien que certaines apportent une réelle plus-value en terme de durcissement sur le système, ces recommandations ne peuvent être considérées seules comme gage de fiabilité quant à leur robustesse. En effet, aucune évaluation de sécurité de ces fonctions n'a aujourd'hui été réalisée par l'ANSSI.

Cette note se concentre sur les fonctions de sécurité implantées au sein des processeurs et ne prétend donc pas à l'exhaustivité quant aux mécanismes de sécurité dont la mise en œuvre s'appuie sur des composants externes à ceux-ci. En particulier le TPM ainsi que les mécanismes qui l'utilisent (par exemples : Intel TXT ou le SecureBoot) ne sont pas abordés ici.

3 Support matériel

Ce chapitre décrit des fonctionnalités matérielles déjà présentes, ou proches de l'être, sur architecture x86 et pouvant avoir des conséquences positives ou négatives sur la sécurité de l'ensemble du système.

Le marché x86 est essentiellement contrôlé par deux acteurs : Intel et AMD. Bien que leurs processeurs et composants demeurent largement compatibles entre-eux [1], des divergences peuvent apparaître en terme de délai sur la disponibilité d'une fonctionnalité entre l'un et l'autre. Il est nécessaire d'en tenir compte et notamment de se renseigner sur la présence éventuelle des options en fonction du fournisseur de matériel.

3.1 32 ou 64 bits ?

La très grande majorité des processeurs x86 supportent aujourd'hui au moins deux modes de fonctionnement :
- le mode protégé (`protected mode`), cantonné à un adressage virtuel sur 32 bits (4 octets) ;
- le mode long (`long mode`), qui permet un adressage plus large sur 64 bits [2] (8 octets).

1. Les instructions et méthodes de virtualisation matérielle font exception.
2. Seuls 48 bits sont effectivement exploités, les 16 bits non utilisés faisant partie des adresses « non-canoniques ».

L'utilisation du mode 64 bits présente des avantages par rapport au mode protégé :
- l'espace d'adressage plus important donne plus de latitude dans le choix des adresses, en particulier lorsqu'il est fait de manière aléatoire (ASLR) ;
- la présence systématique de bits de protections NX/XD sur les pages mémoires, permettant de les déclarer non-exécutables [3] ;
- l'adressage relatif par registre `rip`, plus naturel (et performant) pour du code qui peut être chargé voire exécuté indépendamment de sa position (PIC, PIE) ;
- l'accroissement de l'espace d'adressage du noyau et des processus (de 4GiB à 128TiB) permet d'augmenter les performances de programmes nécessitant de grandes quantités de mémoire, comme certaines bases de données.

R1

> Le processeur choisi doit être capable de supporter les instructions 64 bits AMD (dénommée aussi `AMD64`, `Intel 64` ou `x86_64`).

Les jeux d'instructions 32 bits restent très largement compatibles entre Intel et AMD. Celui associé au mode 64 bits peut revêtir différents noms, et suivant les systèmes d'exploitation on le retrouve sous les dénominations `AMD64`, `x86_64`, `EM64T` ou `Intel 64`. Il ne doit pas être confondu avec l'architecture `IA-64` (Intel Itanium 64 bits), tombée en désuétude.

R2

> Lorsque la machine le supporte, privilégier des systèmes en version 64 bits plutôt qu'en 32 bits.

Il est toutefois important de noter que le mode 64 bits abandonne partiellement le mécanisme de segmentation de la mémoire, propre au mode 32 bits. Certains projets [4] exploitent avantageusement les possibilités offertes par la segmentation pour durcir significativement les systèmes d'exploitation ; son abandon partiel constitue de ce point de vue une régression en matière de sécurité.

3.2 Bit NX/XD – droit d'exécution mémoire

Les droits d'accès sur les pages mémoire physiques sont usuellement spécifiés au travers de 3 bits :

- `R` (« read ») : droit d'accès en lecture ;
- `W` (« write ») : droit d'accès en écriture ;
- `NX/XD` (« no-execute » ou encore « execute-disable ») : indique si le contenu de la mémoire est exécutable ou non.

Ces bits permettent de donner des accès en lecture, écriture ou exécution au besoin. Les bits *R* et *W* sont arrivés en même temps que le jeu d'instructions 32 bits et sont donc supportés par les processeurs et la MMU depuis très longtemps. En revanche, le bit de protection NX/XD est apparu plus tardivement. Son utilisation en 32 bits requiert la présence d'un mécanisme particulier : la fonction PAE. Ce bit est quasi-systématiquement présent en 64 bits.

3. La protection NX/XD est également disponible en mode 32 bits, à condition d'activer le mode PAE (voir ci-dessous).
4. La segmentation est notamment utilisée comme mesure technique de durcissement système par Grsecurity via *UDEREF*, ou par l'hyperviseur Xen pour protéger les accès à sa mémoire depuis les noyaux des domU.

R3

Si le mode 32 bits est le mode d'exécution retenu pour le système d'exploitation, le processeur et sa MMU doivent être capables de supporter le mode PAE.

3.3 Hyper-Threading

La technologie *Hyper-Threading* permet de multiplier par deux le nombre de cœurs d'exécution présents sur un ordinateur en exploitant diverses optimisations matérielles. Bien que cette technologie semble intéressante (elle permet de doubler « gratuitement » la quantité d'unités d'exécution), elle donne accès à des canaux cachés.

Comme toute technologie qui partage des ressources (caches, unités de prédictions de branchement, cœurs d'exécution, etc.) entre différents processus pouvant avoir des niveaux de confiance différents, elle expose des éléments sensibles à des attaques par canaux auxiliaires. Des exemples de construction et d'exploitation existent et sont toujours d'actualité [4, 5, 9]. Ils permettent :

- d'établir des canaux de communication entre processus coopératifs sans que le système d'exploitation ne puisse les détecter ou les bloquer ;
- de récupérer des éléments sensibles (par exemple des clés privées lorsque les cœurs effectuent des opérations cryptographiques) sans coopération du processus ciblé.

Les cas où une machine manipule plusieurs contextes de sécurité différents sont assez courants : frontaux d'accès SSL/TLS (passerelles de serveurs Web notamment), serveurs mandataires et mandataires inverses, passerelles de connexions VPN, etc.

R4

Lorsque les services hébergés sur l'hôte sont susceptibles de gérer des contextes de sécurité distincts pouvant être compromis par les canaux cachés précédemment mentionnés, les ressources matérielles associées à chaque contexte doivent leur être dédiées :

- par la désactivation de la fonctionnalité qui permet d'exploiter ces canaux cachés (par exemple en mettant à *off* l'option pour l'*Hyper-Threading* dans le BIOS) ;
- par l'emploi d'une machine physique, d'un composant ou d'un mécanisme dédié à la manipulation des éléments sensibles (clés secrètes, clés privées, empreintes de mots de passe) ;
- par l'attribution de ces ressources matérielles aux processus au travers des APIs fournies par le système d'exploitation (CPU pinning, paramètrage de l'ordonnanceur, etc.).

3.4 SMEP

La fonction SMEP sert à empêcher les tentatives d'exécution de code situé dans un espace mémoire moins privilégié que celui du contexte d'exécution du processeur ; ce dernier levant une faute matérielle le cas échéant. Cette fonction complexifie l'exploitation de certaines vulnérabilités du système d'exploitation car elle empêche le noyau d'exécuter directement du code résidant dans l'espace mémoire utilisateur naturellement plus exposé aux attaques que le noyau lui-même.

La fonction SMEP est disponible en mode 32 et 64 bits, sur les processeurs Intel depuis les modèles *Ivy Bridge* ; elle n'est actuellement pas supportée par les processeurs AMD.

Les dernières versions des systèmes d'exploitation majoritaires supportent SMEP : sous Windows à partir de la version 8, sous Mac OS X à partir de la version 10.8 (Mountain Lion) et pour le noyau Linux à partir de la version 3.0 (disponible à partir de la version 12.04 pour Ubuntu, Wheezy pour Debian et la version 7 pour la Red Hat Enterprise Linux).

R5

Il est recommandé d'utiliser un processeur et un système d'exploitation capable de supporter la fonctionnalité SMEP.

3.5 SMAP

La fonction SMAP complète la fonction SMEP en ce qu'elle empêche, pour un contexte d'exécution de privilège donné, d'accéder à des pages mémoires en lecture/écriture marquées à un niveau de privilège plus faible. SMAP permet donc de détecter l'accès en lecture/écriture par le noyau à des pages mémoires situées en espace utilisateur ; elle limite ainsi les possibilités offertes à un attaquant de manipuler les données accédées par un noyau dans le but de le compromettre.

La fonction SMAP est apparue avec les premiers modèles *Broadwell* de processeurs Intel ; elle n'est actuellement pas supportée par les processeurs AMD. Le support logiciel est arrivé avec la version 3.7 du noyau Linux (disponible à partir de la version 12.04.3 LTS pour Ubuntu, Jessie pour Debian et la version 7 pour la Red Hat Enterprise Linux). Windows et Mac OS X n'ont pas encore annoncé de support pour cette fonctionnalité.

R6

Il est recommandé d'utiliser un processeur capable de supporter la fonctionnalité SMAP.

3.6 AES-NI

AES-NI désigne une famille d'instructions non privilégiées permettant de calculer le résultat d'une opération de chiffrement/déchiffrement en utilisant une implémentation matérielle partielle de l'algorithme AES. La mise en œuvre retenue par Intel consiste à implémenter chaque tour de l'AES en une instruction plutôt que de proposer une unique instruction de calcul d'une opération AES complète. Cette approche permet au programme d'avoir un contrôle plus fin de la réalisation des opérations cryptographiques faites par le processeur.

L'instruction `pclmulqdq` (*Carry-less Multiplication Quad to Quad*) complète la famille AES-NI en calculant une multiplication binaire sans retenue de deux opérandes de taille 64 bits, ce qui permet d'accélérer certains modes opératoires de chiffrement tels que GCM.

Outre le gain significatif en performances qu'offrent ces instructions par rapport à des implémentations logicielles, en particulier pour le chiffrement de disques ou de protocoles réseau tels que TLS ou IPsec, l'avantage qu'apportent les instructions AES-NI réside dans la protection contre les attaques

par canaux auxiliaires temporels basées sur les latences de cache. Dans certaines conditions favorables, ces attaques peuvent permettre à un processus utilisateur quelconque de récupérer la clé secrète de chiffrement d'un autre processus en quelques secondes par mesure des temps d'accès aux données en cache du CPU partagé entre les deux processus [5].

Les AES-NI ont été introduites par Intel en 2010 avec les premiers modèles *Westmere* réservés au Xeon haut de gamme. Elles ont ensuite été progressivement intégrées à la génération *Ivy Bridge* (*Core i5* et *i7* notamment), et sont normalement présentes sur les CPU Intel modernes. AMD les implémente depuis la microarchitecture *Bulldozer*.

L'usage correct de ces instructions dans le logiciel nécessite une bonne connaissance de la façon dont l'AES s'opère. Par conséquent, il est préférable d'utiliser une implémentation existante plutôt que d'exploiter directement les instructions AES-NI.

Aujourd'hui de nombreuses bibliothèques cryptographiques ont mis à jour leur code afin de pouvoir en profiter :

- *Crypto API* du noyau Linux à partir du 2.6.30 ;
- *OpenCrypto API* à partir de FreeBSD 9 ;
- *cryptographic framework* à partir d'OpenBSD 4.9 ;
- *CNG/Cryptography API : Next Generation* à partir de Windows 7 et Windows Server 2008R2 ;
- *OpenSSL* à partir de la version 1.0.1 au travers de l'API *EVP* ;
- *NSS* à partir de la 3.12.9 ;
- *Java* à partir de l'OpenJDK 8 pour la VM serveur (ou n'importe quel provider AES-NI avec l'API PKCS#11).

R7

Il est recommandé d'utiliser un processeur capable de supporter le jeu d'instruction AES-NI.

3.7 RDRAND et RDSEED

RDRAND et *RDSEED* sont deux nouvelles instructions non privilégiées des jeux d'instruction 32 et 64 bits x86. Elles permettent à n'importe quel processus du système de demander de 16 à 64 bits d'aléa au processeur. De la finalité de cet aléa va dépendre le choix de l'instruction à privilégier : l'aléa retourné par *RDRAND* est destiné à un usage direct tandis que celui de *RDSEED* est plutôt réservé à l'initialisation d'un générateur pseudo-aléatoire (obtention d'une graine).

L'instruction *RDRAND* est arrivée avec les processeurs *Ivy Bridge* Intel (2012). L'instruction *RDSEED* est arrivée plus tard avec les processeurs *Broadwell* (2014). AMD n'a pas communiqué sur le support éventuel d'instructions équivalentes par leurs processeurs. La documentation Intel [3] donne une description plus détaillée de la façon dont ces instructions fonctionnent.

Il est cependant vivement recommandé de ne pas reposer exclusivement sur celles-ci pour fournir de l'aléa mais de les combiner avec les méthodes déjà présentes au niveau du système d'exploitation.

Elles sont cependant très utiles lors du démarrage, période pendant laquelle l'entropie accumulée par le système est souvent faible.

R8

> Il est recommandé d'utiliser une architecture capable de fournir une source d'aléa matérielle.
> Cette source doit être utilisée conjointement à d'autres sources indépendantes afin d'éviter tout biais relatif lors de la génération d'aléa.

Le support de *RDRAND* peut contribuer à améliorer le fonctionnement de sources d'aléas telles `/dev/[u]random` sous GNU/Linux, mais il conseillé de continuer à utiliser ces interfaces plutôt que l'instruction directement : *RDRAND* ne doit pas se substituer à un générateur d'aléa déjà existant mais bien le compléter.

3.8 Virtualisation matérielle

Bien que les grands principes de virtualisation aient été posés il y a plusieurs décennies [5], ceux-ci se retrouvent de plus en plus souvent déclinés à une multitude de domaines. Cela conduit à l'apparition de « nouveaux » usages dont les risques et conséquences ne sont pas toujours connus et maîtrisés.

Les éléments mentionnés dans la présente note technique viennent en complément de la note traitant des Problématiques de sécurité associées à la virtualisation des systèmes d'information [6] disponible sur le site de l'ANSSI.

3.8.1 Virtualisation de contexte d'exécution

Suivant les constructeurs cette technologie porte différents noms : VT-x chez Intel ou AMD-V chez AMD. Cette technologie a fait l'objet de plusieurs études [6, 8, 10]. Elle a été introduite à l'origine pour résoudre les limitations inhérentes aux architectures x86 concernant la virtualisation des instructions privilégiées. Correctement utilisée, elle permet l'élaboration d'hyperviseurs capables d'appliquer des vérifications de durcissement et d'intégrité sur le système sous-jacent. À l'inverse, elle peut être exploitée par certaines formes de *rootkits* pour se rendre difficilement détectables vis-à-vis du système virtualisé, anéantissant ainsi tout effort d'analyse et de rémédiation.

Elle a commencé à apparaître en 2005, mais n'a été réellement accessible sur les processeurs qu'à partir de 2009. Il est nécessaire de consulter la page dédiée [2] sur le site d'Intel afin de savoir si le processeur choisi la supporte ou pas. AMD l'a introduite avec ses *Athlon 64 rev F* et CPU *Barcelona*. Elle ne doit pas être confondue avec d'autres technologies aux noms proches (VT-d et AMD-Vi) qui couvrent d'autres aspects de la virtualisation.

R9

> Par principe de précaution, en réponse au risque mentionné supra, la fonction de virtualisation de contexte d'exécution doit être désactivée lorsqu'elle n'est pas strictement nécessaire au fonctionnement du système.

5. Notamment avec l'apparition des MMU et de leur concept de mémoire virtuelle.
6. http://www.ssi.gouv.fr/fr/guides-et-bonnes-pratiques/recommandations-et-guides/securite-du-poste-de-travail-et-des-serveurs/problematiques-de-securite-associees-a-la-virtualisation-des-systemes-d.html.

La procédure à suivre pour désactiver cette option peut varier d'un matériel à l'autre, il est donc nécessaire de bien consulter la documentation associée à chacun.

3.8.2 Virtualisation des entrées/sorties

La virtualisation des entrées/sorties désigne un mécanisme similaire à celui déjà rencontré pour la mémoire virtuelle sur la quasi totalité des architectures matérielles d'aujourd'hui : l'usage d'une unité de contrôle des accès mémoires en fonction d'un contexte préalablement configuré. Là où une MMU se préoccupe des accès pour un contexte d'exécution (généralement un processus), une IOMMU va gérer les accès mémoires effectués pour un périphérique d'entrée/sortie donné.

Cette fonctionnalité est dénommée VT-d chez Intel et AMD-Vi chez AMD. Initialement conçue pour des applications associées à la virtualisation matérielle [7], elle peut être avantageusement utilisée par un système d'exploitation classique en bloquant les attaques issues de périphériques malveillants disposant d'accès mémoire direct (DMA). Ces accès sont une menace importante pour la sécurité physique des systèmes car les périphériques qui en disposent sont capables de lire et d'écrire directement en mémoire physique, sans possibilité de contrôle par le système d'exploitation [1, 7].

Bien que son efficacité dépende de sa position au sein de l'architecture matérielle, une IOMMU permet dans la plupart des cas d'empêcher la prise de contrôle d'une machine via des requêtes DMA issues de périphériques malveillants. En complément des fonctions de cloisonnement apportées par cette unité de gestion mémoire des entrées/sorties, deux mécanismes permettent de restreindre les privilèges d'accès offerts aux périphériques d'entrée/sortie :

- *Interrupt Remapping* : ce mécanisme permet de virtualiser les requêtes d'interruptions matérielles de la même façon que les requêtes DMA ;
- *Access Control Services* : ces services contrôlent les communications inter-périphériques, afin d'éviter par exemple que deux interfaces réseaux puissent s'échanger des informations sans supervision par le système d'exploitation.

Cette fonctionnalité est essentiellement réservée aux machines haut de gamme et requiert des « chipsets » particuliers sur la carte-mère : ce sont les spécifications matérielles dans leur globalité qu'il faut regarder et non uniquement celles du CPU. La fonction IOMMU doit être précisée dans les spécifications du fabriquant de l'ordinateur. L'activation de la technologie VT-x n'est pas nécessaire afin de profiter des avantages de l'IOMMU, VT-d/AMD-Vi peut être utilisé sans les services de virtualisation de contextes d'exécution.

Mac OS X a introduit le support de l'IOMMU dans Moutain Lion ; quant à Windows, le support n'est pas annoncé pour l'instant dans ses versions basiques, mais est disponible par Hyper-V dans Windows Server 2012. Linux, VMWare et Xen peuvent également s'en servir suivant leur configuration.

R10

Le service d'IOMMU doit être activé lorsque l'architecture matérielle le supporte.

Forcer l'usage du service d'IOMMU peut entraîner le dysfonctionnement d'une partie du système (périphériques et contrôleurs non reconnus, erreurs d'entrées/sorties, etc.). L'administrateur est invité à effectuer une validation fonctionnelle du système une fois cette fonction activée.

7. Elle n'est d'ailleurs présente qu'en complément de la technologie VT-x.

4 Configuration générale du BIOS/UEFI

Ce chapitre vise à donner quelques recommandations en rapport avec la configuration du BIOS/UEFI d'un ordinateur x86 compatible PC sous l'angle de la sécurité. Il vise à s'assurer que tous les mécanismes pouvant contribuer positivement à la sécurité du système soient correctement activés et accessibles au système d'exploitation.

4.1 Contrôle d'accès au menu de configuration

Afin d'empêcher une personne non autorisée de désactiver ou contourner les mesures techniques indiquées ci-dessous, l'accès au BIOS de l'ordinateur doit être protégé par mot de passe.

R11

> Tous les accès en écriture au paramètrage du BIOS (activation/désactivation de fonctions, ordre de démarrage des périphériques, heure système, etc.) doivent être protégés par un mot de passe.
>
> Celui-ci doit être unique à chaque machine.

Les mots de passe doivent dans la mesure du possible être choisis en tenant compte des recommandations de la note technique Recommandations de sécurité relatives aux mots de passe [8] disponible sur le site de l'ANSSI.

4.2 Activation de périphériques et connecteurs

Le nombre croissant de périphériques et de connecteurs présents sur les ordinateurs augmente mécaniquement la surface d'attaque à laquelle ces derniers sont exposés. La menace potentielle que constituent ces périphériques dépend de leur type, du protocole matériel utilisé ou encore du bus sur lequel ils sont raccordés.

Certains BIOS permettent de configurer l'exposition de la plupart des ports et des composants périphériques vis-à-vis du système d'exploitation.

R12

> Lorsque cela est possible, en application du principe de minimisation, les composants non requis au bon fonctionnement du système doivent être désactivés.
> Cette désactivation peut porter sur la configuration de connecteurs externes (port(s) série, contrôleurs USB, interfaces de disques externes comme eSATA ou Thunderbolt, etc.) et éventuellement certains périphériques internes ou sans-fil comme des contrôleurs disques, des cartes Wifi/Bluetooth ou multimédia intégrées.

La désactivation de composants inutilisés est bénéfique pour la sécurité à au moins deux titres :
– elle réduit l'exposition de la machine à des attaques par des canaux de communication non nécessairement maîtrisés, en particulier quand elle est située en environnement peu ou pas sûr ;

8. http://www.ssi.gouv.fr/fr/guides-et-bonnes-pratiques/recommandations-et-guides/securite-du-poste-de-travail-et-des-serveurs/mot-de-passe.html.

– les périphériques désactivés ne sont pas détectés par le système d'exploitation. Celui-ci ne charge donc pas les modules, *drivers* et *firmwares* nécessaires à leur fonctionnement, lesquels peuvent contenir des vulnérabilités exploitables.

4.3 Ordre de démarrage

La possibilité d'amorcer une machine sur un périphérique différent de celui sur lequel est installé le système d'exploitation expose potentiellement l'intégrité de ce dernier. En l'absence de mécanismes à même d'empêcher l'amorçage d'un système d'exploitation non maîtrisé sur la machine, il est nécessaire de figer le périphérique sur lequel est installé le système d'exploitation principal en tant que périphérique d'amorçage unique.

R13

L'ordre de démarrage des périphériques susceptibles d'amorcer le système doit mettre en priorité la plus haute le composant sur lequel est installé le système final. Les autres périphériques ne doivent pas intervenir dans la chaîne de démarrage.

4.4 Bit NX/XD

Certains BIOS x86 Intel permettent de désactiver le bit spécifiant le droit d'exécution [9]. La fonctionnalité est dénommée de différentes façons : « Enable NX bit », « Enable execute-disable bit », etc. Il convient de consulter le manuel associé au matériel pour repérer la bonne option.

R14

La fonction de contrôle d'accès en exécution d'une page mémoire doit être activée dans le BIOS.

4.5 Fonction VT-x

La plupart des BIOS permettent de configurer l'activation ou la désactivation de la fonctionnalité VT-x/AMD-V. La terminologie employée dépend cependant du constructeur, elle est souvent présentée sous des noms peu explicites (« Virtualization technology », « Virtualization extensions », ou encore « Vanderpool »). Elle est souvent confondue avec la fonction VT-d (IOMMU).

R15

Comme recommandé précédemment, la fonction de virtualisation de contexte (VT-x) doit être désactivée par défaut. L'activation de la fonctionnalité est une mesure d'exception qui doit être étudiée en fonction du contexte, comme expliqué dans le chapitre traitant de la virtualisation matérielle.

4.6 Fonction VT-d

Lorsque le chipset de la carte mère dispose d'une IOMMU, le BIOS peut proposer une option de configuration visant à permettre son activation. Elle est souvent désignée sous les termes « Enable IOMMU », « Enable VT-d » ou encore « IO virtualization ». La documentation du matériel doit être

9. Il s'agit en fait du bit de non-exécution, mais la finalité reste la même : autoriser ou non l'exécution de la page pointée.

consultée afin de trouver l'option associée.

R16

> Comme recommandé supra, l'IOMMU doit être activée sur les systèmes supportant le mécanisme de virtualisation des entrées/sorties.

4.7 Tableau de fonctionnalités

Fonctionnalité	Utilisation	Remarques
64 bits	Recommandée	Disponible sur Intel et AMD après 2004. Nécessite l'usage d'un système d'exploitation 64 bits.
NX/XD	Recommandée	Requiert PAE en 32 bits ; supporté nativement en 64 bits. Disponible sur Intel et AMD depuis 2004. Supporté par tous les systèmes d'exploitation récents.
Hyper-Threading	Déconseillée	Canaux cachés difficilement maîtrisables.
SMEP	Recommandée	Disponible uniquement sur Intel depuis 2013 (*Ivy Bridge*). Supporté nativement sous Windows 8, Mac OS X 10.8 ou Linux 3.0.
SMAP	Recommandée	Disponible uniquement sur Intel depuis 2014 (*Broadwell*). Supporté nativement par Linux 3.7.
AES-NI	Recommandée	Disponible depuis 2010 sur Intel (*Westmere*), et depuis 2011 sur AMD (*Bulldozer*). Supporté nativement par la plupart des bibliothèques cryptographiques.
RDRAND/RDSEED	Voir remarque	Utilisation uniquement en complément d'autres sources d'aléa sur le système.
VT-x/AMD-V	Déconseillée	Activation uniquement lorsque le système nécessite l'usage de mécanismes de virtualisation d'exécution, sans solution alternative possible.
VT-d/AMD-Vi	Recommandée	Prendre garde aux dysfonctionnements éventuels du système lors de l'activation de l'IOMMU.

Références

[1] Damien Aumaitre, Christophe Devine. *Subverting Windows 7 x64 Kernel with DMA attacks*, 2010. http://esec-lab.sogeti.com/dotclear/public/publications/10-hitbamsterdam-dmaatacks.pdf.

[2] Intel Corporation. *About Intel Virtualization Technology*. http://ark.intel.com/Products/VirtualizationTechnology.

[3] Intel Corporation. *DRNG Software Implementation Guide*, 2014. https://software.intel.com/en-us/articles/intel-digital-random-number-generator-drng-software-implementation-guide.

[4] Dag Arne Osvik, Adi Shamir, Eran Tromer. *Cache Attacks and Countermeasures : the Case of AES*, 2005. http://www.tau/ac.il/~tromer/papers/cache.pdf.

[5] Colin Percival. *Cache Missing for Fun and Profit*, 2005. http://www.daemonology.net/papers/htt.pdf.

[6] Joanna Rutkowska. *Subverting Vista Kernel for Fun and Profit*, 2006. http://blackhat.com/presentations/bh-usa-06/BH-US-06-Rutkowska.pdf.

[7] Fernand Lone Sang, Vincent Nicomette, Yves Deswarte, Loïc Duflot. *Attaques DMA peer-to-peer et contremesures*, 2011. https://www.sstic.org/media/SSTIC2011/SSTIC-actes/attaques_dma_peer-to-peer_et_contremesures/SSTIC2011-Article-attaques_dma_peer-to-peer_et_contremesures-lone-sang_duflot_nicomette_deswarte.pdf.

[8] Tatsuya Takehisa, Hiroki Nogawa, Masakatu Morii. *AES Flow Interception : Key Snooping Method on Virtual Machine*, 2011. https://eprint.iacr.org/2011/428.pdf.

[9] Yinqian Zhang, Ari Juels, Michael K. Reiter, Thomas Ristenpart. *Cross-VM Side Channels and Their Use to Extract Private Keys*, 2012. http://dx.doi.org/10.1145/2382196.2382230.

[10] Éric Lacombe, Vincent Nicomette, Yves Deswarte. *Une approche de virtualisation assistée par le matériel pour protéger l'espace noyau d'actions malveillantes*, 2009. http://actes.sstic.org/SSTIC09/Une_approche_de_virtualisation_assistee_par_le_materiel_pour_proteger_l_espace_noyau_d-actions_malveillantes/SSTIC09-article-E-Lacombe-V-Nicomette-Y-Deswarte-Une_approche_de_virtualisation_assistee_par_le_materiel_pour_proteger_l_espace_noyau_d-actions_malveillantes.pdf.