Utilisation de Chipsec pour valider la sécurité de plate-formes matérielles

Arnaud Malard et Yves-Alexis Perez
arnaud.malard@ssi.gouv.fr
yves-alexis.perez@ssi.gouv.fr

ANSSI

Résumé. Dans le cadre de l'achat des postes de travail informatiques du gouvernement français, l'ANSSI a mis en œuvre avec la DAE (direction des achats de l'État) un processus permettant de s'assurer que ces machines soient conformes à des exigences de sécurité matérielles. Celles-ci, rédigées par l'ANSSI, vérifient notamment que les constructeurs de PC maîtrisent les composants matériels embarqués, qu'ils proposent des fonctionnalités de sécurité minimales ou encore que les firmware (BIOS/UEFI) soient correctement configurés. Pour s'assurer du respect des exigences techniques, un outil dédié aurait pu être développé mais l'ANSSI a choisi d'utiliser un outil existant et fiable : Chipsec, projet libre écrit en Python et soutenu par Intel. Chipsec est très modulable et permet aussi bien de relever les registres de configuration de contrôleurs matériels que d'en interpréter les valeurs pour remonter un premier état de sécurisation. L'autre atout de Chipsec, qui a poussé l'ANSSI à l'utiliser, est qu'il est fortement maintenu, aussi bien au niveau des processeurs supportés qu'au niveau des registres à vérifier selon les dernières vulnérabilités bas niveau découvertes.

1 Contexte : détection de vulnérabilités liées au matériel

En 2015, du trafic réseau suspect est détecté dans le réseau d'une administration française. Ce trafic réseau est rapidement lié au logiciel Computrace, installé de façon non autorisée sur des postes clients récemment acquis et déployés par le service informatique.

1.1 Computrace

Computrace est un logiciel de lutte contre le vol des ordinateurs portables. Maintenant nommé Absolute Home & Office, il est composé de deux parties :

— un agent logiciel, s'exécutant au-dessus du système d'exploitation, qui maintient un lien permanent avec les serveurs de la société Absolute afin de transmettre la position géographique de l'ordinateur et vérifier s'il n'a pas été déclaré volé ;

— un module persistant, activé au niveau du BIOS (ou firmware UEFI) de la machine, entre autres chargé d'injecter l'agent logiciel lors du démarrage du système d'exploitation (Windows en particulier), afin d'empêcher que le formatage du disque dur ne désactive les fonctionnalités d'anti-vol.

Les possibilités de l'agent logiciel ne se limitent pas à la fonctionnalité de vérification du statut de la machine et à la transmission de la position géographique (connue grâce à l'éventuel présence d'un GPS ou bien par triangulation grâce aux points d'accès wifi présents à proximité). L'agent s'exécute avec des privilèges élevés et est capable d'actions sur le système qui peuvent s'apparenter à celles d'un rootkit, en particulier la possibilité d'exécuter des commandes et d'installer de nouveaux logiciels.

Les privilèges élevés conférés au logiciel (ainsi qu'au module persistant) ont mené à l'examen de celui-ci par l'entreprise Kaspersky à deux reprises, en 2009 puis en 2014. Les investigations se sont traduites par la publication d'un article [18] ainsi qu'une présentation [19]. Ces documents détaillent des vulnérabilités identifiées dans Computrace, qui peuvent mener à une prise de contrôle complète du système par un attaquant.

1.2 Investigation

L'activation silencieuse du logiciel Computrace a été tracée jusqu'à un communiqué [20] de l'entreprise Lenovo, fabricante des postes de travail déployés au sein de l'administration concernée. Dans certaines versions de BIOS, Lenovo a *involontairement* activé dans le BIOS le module de persistance, ce qui se traduit par l'injection dans le processus de démarrage de Windows du premier étage de l'agent Computrace, qui cherche ensuite à vérifier l'activation du service auprès des serveurs de la société. Une des machines concernées par cette version de BIOS a été utilisée pour générer le *master* d'installation des postes du service, qui a ensuite servi à installer un certain nombre de machines, déployées par la suite au sein du service. C'est ce qui a mené à la génération de trafic réseau vers les serveurs de l'entreprise Absolute.

1.3 Réponse immédiate

Une fois l'origine du problème identifié, la résolution a principalement consisté en une coordination avec Lenovo et le service informatique afin de désactiver le module persistant, mettre à jour les BIOS des machines concernées et éviter une réinstallation de l'agent.

1.4 Réponse à plus long terme

Même si l'origine malveillante de cet épisode n'est pas démontrée, il montre l'importance d'avoir un suivi des composants matériels en plus des composants logiciels, au sein d'un parc informatique. En effet, même si les postes informatiques sont considérés comme du matériel, ils contiennent une quantité importante de codes informatiques, sous une forme communément appelée *firmware* (ou micrologiciels). On pense bien évidemment au BIOS qui s'exécute au démarrage sur le processeur principal, mais il existe aussi les logiciels s'exécutant sur d'autres processeurs, par exemple celui des périphériques (carte réseau, carte graphique etc.), des co-processeurs tels que le Intel CSME (*Converged Security & Management Engine*), ou encore les microcodes du processeur lui-même.

Comme l'ont d'ailleurs prouvé les années récentes avec la kyrielle de vulnérabilités touchant les processeurs (Spectre, Meltdown, Foreshadow etc.), tous ces codes nécessitent parfois des mises à jours et donc un suivi, qui est délicat et assez rarement effectué.

2 Rédaction d'exigences de sécurité

2.1 Gestation

L'ANSSI, en lien avec la DAE (direction des achats de l'État, administration chargée entre autre de passer des marchés publics pour répondre aux besoins de l'État français) a donc commencé la rédaction d'exigences de sécurité qui s'appliqueraient aux achats de matériel informatique (postes de travail en particulier).

Les exigences de sécurité matérielles, maintenant publiées sur le site de l'ANSSI [3], ont été conçues afin de répondre sur la durée aux enjeux de sécurité du matériel. Ces exigences ont été conçues en coopération avec les partenaires concernés : ministères acquéreurs, constructeurs, ainsi que des homologues internationaux ayant le même type de besoins. Les exigences issues du processus de gestation devaient indiquer aux constructeurs les souhaits de l'administration, tout en restant acceptables et réalisables techniquement, et sans induire un surcoût trop élevé.

Pour des raisons de simplicité, ces exigences s'adressaient dans un premier temps uniquement aux postes de travail (fixes et portables) sur architecture x86 (processeurs Intel et compatibles, comme AMD).

Ces exigences ont ensuite été intégrées au CCTP d'un marché public initié par la DAE afin d'approvisionner les administrations centrales de l'État en matériel client (postes de travail fixes et nomades). Les candidats

soumettant une réponse à ce marché public devaient s'engager à fournir des matériels conformes aux exigences de sécurité. Cette conformité serait ensuite vérifiée par les services du « pouvoir adjudicateur » (l'ANSSI, par délégation de la DAE).

2.2 Détails techniques

Les exigences sont regroupées en quatre familles, et chaque exigence est associée à un objectif de sécurité qui est détaillé dans le document [3]. Sans les lister de façon exhaustive, voici quelques détails sur les différentes familles.

Maîtrise de la plate-forme Cette famille d'exigences cherche à garantir la maîtrise de la plate-forme par son propriétaire, c'est à dire ici l'État français (et par délégation le service informatique, incarné par un administrateur système). À contrario, on cherche à limiter l'influence d'autres acteurs, qu'ils soient connus comme le constructeur de la plate-forme ou le fournisseur du processeur ou d'un système d'exploitation, ou inconnu. On y retrouve donc la possibilité pour l'administration de choisir le système d'exploitation qu'elle déploiera sur les machines acquises, ou encore la fourniture d'un inventaire complet des composants matériels constituant la plate-forme (périphériques, interfaces de communication etc.). L'inventaire permet à la fois de s'assurer que la plate-forme ne dispose pas de composants non connus du constructeur, mais aussi de faire une veille de sécurité sur le parc installé.

Caractéristiques matérielles Ces exigences rassemblent des composants de sécurité dont l'administration souhaite disposer sur les plates-formes acquises. On y retrouve actuellement deux composants, optionnels dans le monde x86 mais qui apportent des garanties de sécurité intéressantes quand elles sont utilisées par le système. Le premier est le TPM (*Trusted Platform Module*), exposant au système un certain nombre de fonctionnalité de sécurité (par exemple le scellement et descellement de secrets, en fonction de l'état de la machine). Il est imposé que ce composant soit certifié Critères Communs selon le profil de protection défini par le TCG [1]), afin d'avoir de bonnes garanties en matière de sécurité (ceci impose en particulier qu'il s'agisse d'un composant matériel dédié et non d'un code logiciel s'exécutant sur un composant existant comme le CPU

1. *Trusted Computing Group*, le consortium à l'origine des spécifications du TPM

ou le chipset). Le deuxième composant est une I/OMMU (*Input/Output Memory Management Unit*) qui permet d'isoler les périphériques d'entrées/sorties et en particulier limiter leurs accès à la mémoire centrale.

Caractéristiques du firmware Cette catégorie regroupe le gros des exigences et couvre à la fois les fonctionnalités de sécurité offertes par le BIOS, comme la configuration d'un mot de passe au démarrage, le changement de l'ordre de démarrage ou encore le changement des clés *SecureBoot*, mais aussi les solutions de prise en main à distance et la sécurité et la protection du firmware lui-même.

En effet, un certain nombre de vulnérabilités ont touché le BIOS ces dernières années. Comme il s'agit du premier code exécuté par le processeur principal au démarrage de la machine et qu'il n'est pas sous contrôle du propriétaire de la plate-forme, il est important que le constructeur, responsable de son écriture, en assure la sécurité, tant en amont (au moment de son développement et de son déploiement) qu'en aval. De plus, le BIOS est responsable de la configuration initiale de la machine avant de passer le relai au chargeur de démarrage (*bootloader*) et au système d'exploitation. Un défaut de configuration ou un manque de verrouillage de la plate-forme peut permettre à un attaquant ayant déjà des privilèges élevés sur la plate-forme (typiquement *ring 0*) de les augmenter encore, voire de garder une persistance sur celle-ci. Les exigences imposent donc le maintien du BIOS et de la configuration de la plate-forme à l'état de l'art, qui progresse en permanence.

Maintien en condition de sécurité Enfin, pour complèter les exigences et assurer une pérennité dans le temps des systèmes acquis par l'administration, des exigences imposent le suivi par le constructeur des vulnérabilités (logicielles et matérielles) touchant ses plate-formes, et la correction de celles-ci dans un délai raisonnable.

3 Moyens de tests

Les constructeurs soumettant une offre en réponse à l'appel d'offre s'engagent à fournir un matériel conforme aux exigences, mais il est important de vérifier cette conformité, que ce soit a priori ou a posteriori. En effet, les exigences sont parfois complexes car elles touchent des aspects très bas niveau du matériel, et l'état de l'art progresse constamment : de nouvelles vulnérabilités sont découvertes, ou de nouvelles protections sont ajoutées par les fournisseurs.

Un test systématique du matériel a donc été mis en place par l'ANSSI avant ajout des matériels au catalogue. Ces tests portent sur l'ensemble des exigences présentes dans le CCTP et comprennent à la fois une partie manuelle (vérification de la documentation fournie par le constructeur et des fonctionnalités de personnalisation offertes par le firmware, par exemple) et une partie outillée (en particulier pour la vérification de la présence de vulnérabilités connues).

L'outillage utilisé lors du déroulement de ces tests comporte deux clés USB amorçables, l'une permettant de tester et valider le comportement du *SecureBoot* (et en particulier la possibilité de changer les clés de plate-forme), l'autre permettant de collecter des informations sur la configuration du BIOS et de la plate-forme.

Ce deuxième support repose fortement sur l'outil Chipsec [13], publié par Intel et détaillé ci-après. Le support est une version bootable sur clé USB d'une distribution Linux Debian sur laquelle est ensuite déployé Chipsec.

La génération des supports amorçables est détaillée dans le dépôt du projet [2].

3.1 Pourquoi utiliser Chipsec ?

À l'image de Lojax [9], l'unique rootkit UEFI détecté dans la nature, les compromissions bas niveau touchant notamment le BIOS sont difficilement détectables. Ceci est dû à la structure complexe d'une image UEFI, à la difficulté de comparer les binaires la constituant et à identifier ceux potentiellement vulnérables ou malveillants. Chipsec assure une certaine prévention à l'aide d'une liste de vulnérabilités connues et des références de configurations.

Chipsec est un framework Python open source (publié en mars 2014 [23]) dédié à l'analyse du niveau de sécurité des plate-formes Intel x86 en vérifiant les mécanismes de sécurité bas niveau, les paramètres de configuration de composants matériel ou encore des micro-logiciels (BIOS/UEFI). Chipsec est multi-plate-forme, il peut être lancé depuis Windows, Linux et MacOS, ainsi que directement depuis un Shell UEFI.

Chipsec est composé de deux scripts principaux : `chipsec_main.py` et `chipsec_util.py`.

Le premier a été conçu pour détecter automatiquement les mauvaises sécurisations à travers une série de modules (ou plugins). Chacun de ces derniers a une fonction particulière et cherche à détecter une configuration permettant l'exploitation d'une vulnérabilité spécifique. Chacune des

vulnérabilités testées a généralement fait l'objet d'une publication et elles ont une finalité commune : exécuter du code malveillant sur le système d'exploitation depuis un accès plus bas niveau, c'est-à-dire en modifiant le BIOS, en exécutant du code malveillant en mode SMM, ou en exploitant des options CPU mal documentées.

```
[CHIPSEC] ************************  SUMMARY  ************************
[CHIPSEC] Time elapsed            0.080
[CHIPSEC] Modules total           25
[CHIPSEC] Modules failed to run   0:
[CHIPSEC] Modules passed          17:
[+] PASSED: chipsec.modules.common.smrr
[+] PASSED: chipsec.modules.common.memlock
[+] PASSED: chipsec.modules.common.smm
[+] PASSED: chipsec.modules.common.spi_desc
[+] PASSED: chipsec.modules.common.spd_wd
[+] PASSED: chipsec.modules.common.bios_kbrd_buffer
[+] PASSED: chipsec.modules.common.bios_wp
[+] PASSED: chipsec.modules.common.bios_ts
[+] PASSED: chipsec.modules.common.spi_lock
[+] PASSED: chipsec.modules.common.ia32cfg
[+] PASSED: chipsec.modules.common.bios_smi
[+] PASSED: chipsec.modules.common.rtclock
[+] PASSED: chipsec.modules.common.spi_fdopss
[+] PASSED: chipsec.modules.memconfig
[+] PASSED: chipsec.modules.smm_dma
[+] PASSED: chipsec.modules.debugenabled
[+] PASSED: chipsec.modules.remap
[CHIPSEC] Modules information     1:
[#] INFORMATION: chipsec.modules.common.cpu.cpu_info
[CHIPSEC] Modules failed          1:
[-] FAILED: chipsec.modules.common.spi_access
[CHIPSEC] Modules with warnings   1:
[!] WARNING: chipsec.modules.common.cpu.spectre_v2
[CHIPSEC] Modules not implemented 4:
[*] NOT IMPLEMENTED: chipsec.modules.common.me_mfg_mode
[*] NOT IMPLEMENTED: chipsec.modules.common.uefi.s3bootscript
[*] NOT IMPLEMENTED: chipsec.modules.common.uefi.access_uefispec
[*] NOT IMPLEMENTED: chipsec.modules.common.secureboot.variables
[CHIPSEC] Modules not applicable  1:
[*] NOT APPLICABLE: chipsec.modules.common.sgx_check
[CHIPSEC] ************************************************************
```

Fig. 1. Résultats de chipsec_main

Un analyste peut directement lire les résultats finaux pour savoir si sa plate-forme est conforme aux exigences et peut s'appuyer sur la sortie de chacun des modules chipsec pour comprendre précisément pourquoi un test a échoué (voir Fig. 2 et Fig. 3).

Ce mode est celui principalement utilisé dans le cadre des tests des machines acquises pour le marché DAE.

Le second script, `chipsec_util`, est davantage utilisé pour réaliser des opérations manuelles en communiquant de manière ciblée avec un matériel. Dans le cadre des évaluations DAE, pour comprendre le résultat d'un plugin, pour extraire une donnée non prise en charge par l'un des plugins ou encore pour extraire le BIOS, cet outil s'est avéré être très

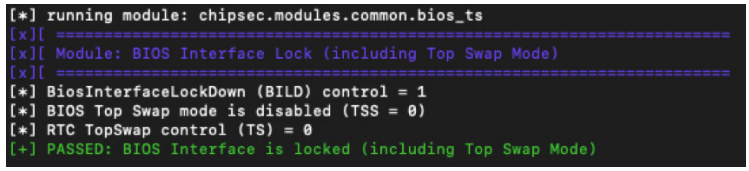


Fig. 2. Paramètre testé par un module conforme

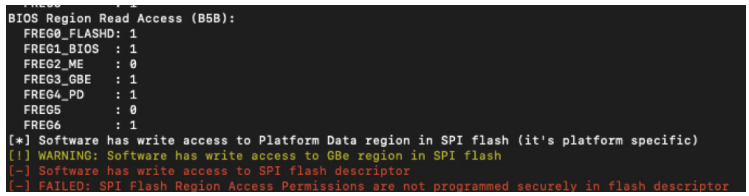


Fig. 3. Paramètre non-conforme détecté par Chipsec

précieux. Le nombre de protocoles et composants pris en charge est assez important : PCI, MMIO, SPI, UEFI, TPM, IOMMU etc.

Présenter ces options nécessiterait un article complet et nous invitons donc le lecteur à se documenter et tester lui-même s'il souhaite approfondir le sujet.

3.2 Les composants à interroger

Une plate-forme récente se compose de plusieurs éléments matériels dont la configuration peut fortement influer sur la sécurité de l'ensemble. Pour émettre un avis sur une plate-forme, Chipsec a besoin de communiquer avec plusieurs composants (voir Fig. 4).

Le processeur (CPU) C'est le composant principal, il exécute le code du système d'exploitation et des applications utilisateurs. Il peut s'exécuter dans plusieurs modes plus ou moins privilégiés dont certains nous intéressent particulièrement ici.

Sur un processeur de la famille x86, on trouvera d'abord le CPL (*Current Privilege Level*, aussi appelé *ring*) qui peut être

3 pour le mode utilisateur, non privilégié ;

0 pour le mode noyau, privilégié.

Le CPL a une influence sur les instructions accessibles au processeur, mais aussi sur les accès à la mémoire, au travers de la MMU.

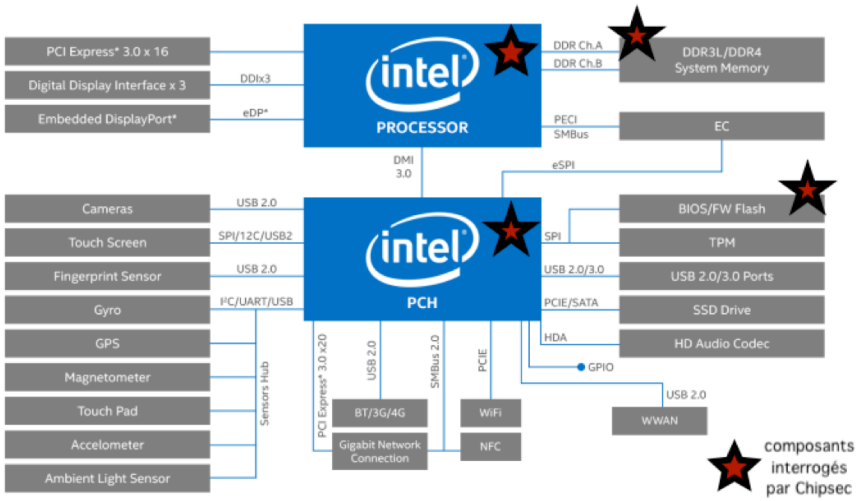


Fig. 4. Représentation d'une plate-forme Intel munie d'un PCH

On trouve ensuite un mode particulier de fonctionnement du processeur, le SMM (*System Management Mode*). Ce mode, initialement dédié à la gestion très bas niveau du sytème (gestion de la température par exemple), est isolé du reste du fonctionnement du système. Une description détaillée se trouve dans le *Intel® 64 and IA-32 Architectures Software Developer's Manual* [12] (volume 3C, chapitre 34), nous en donnerons simplement une description rapide.

Le processeur entre en SMM suite au déclenchement d'une SMI (*System Management Interrupt*), une interruption matérielle qui entraîne automatiquement et de façon transparente pour l'OS une bascule vers le SMM. Le système d'exploitation ou tout autre application en cours d'exécution est suspendue pendant le traitement de l'interruption, puis l'exécution est reprise sans que le système n'ait eu conscience de l'interruption. Lors de l'exécution en SMM, le processeur utilise une zone spéciale de la mémoire, la SMRAM, *System Management RAM*, réservée au stockage volatile du code (par exemple les *SMI Handlers*) et de ses données.

L'adresse de base de la SMRAM en mémoire physique est nommée `SMBASE` et vaut `0x30000` au reset du processeur, mais peut être modifiée en via un registre interne du CPU (le `SMBASE` register). Cette modification n'est possible que façon indirecte, en écrivant depuis le SMM la nouvelle valeur du registre à un offet particulier (`0x7ef8`) dans la sauvegarde des

registres stockée en SMRAM. La nouvelle valeur sera restaurée dans le registre à la sortie du SMM et prise en compte lors des prochaines SMI.

L'accès à la SMRAM est normalement impossible lorsque le processeur ne s'exécute pas en SMM.

Le chipset Il sert d'interface entre le CPU et les périphériques (parfois internes au chipset) et comprend donc plusieurs contrôleurs pour des protocoles divers (PCI, SPI, SATA etc.). Sur des systèmes plus anciens il était composé de deux puces discrètes, le *northbridge* (ou *Memory Controller Hub*, MCH) et le *southbridge* (ou *I/O Controller Hub*, ICH), mais il est maintenant constitué d'une seule, le PCH (*Platform Controller Hub*).

Le PCH comprend de nombreux registres de configuration pour les différents contrôleurs qui le composent. En règle générale ces contrôleurs sont logiquement situés sur le bus PCI Express (PCIe) et les registres de configuration sont accessibles au travers de ce bus, soit directement dans le *PCI Configuration Space* soit par projection dans la mémoire physique d'espace mémoire exposé par le périphérique.

La mémoire vive et son contrôleur L'espace mémoire adressable (parfois appellé « mémoire physique », du point de vue du processeur) d'un système récent est divisé en plusieurs zones, qui pointent par exemple vers les barrettes mémoire, mais aussi vers des zones de configuration ou vers de la mémoire exposée par des périphériques. Le contrôleur mémoire orchestre les accès et les route vers différentes destinations (barrettes, bus PCIe etc.) tout en gérant des droits d'accès.

La mémoire flash et son contrôleur La mémoire flash est une puce discrète sur la carte mère qui comprend généralement 8 à 32 Mo de mémoire en technologie NOR. Cette mémoire est divisée en différentes [2] régions, comme des partitions d'un disque, pour héberger différents types de données, y compris de la configuration.

Sur des plate-formes x86 cette mémoire n'est pas habituellement pas accessible directement par le processeur car elle utilise le protocole SPI (*Serial Peripheral Interface*). Le processeur passe donc par un contrôleur SPI (habituellement situé dans le chipset), lui même exposé sur le bus PCIe.

2. initialement quatre (BIOS, Management Engine, Gigabit Ethernet, Platform data), maintenant beaucoup plus

3.3 Zones de configurations

Notions de registre Nous aborderons souvent la notion de registre matériel dans cet article. Il s'agit tout simplement d'un espace physique bien délimité (mémoire d'un contrôleur de périphérique, mémoire du CPU, PCH etc.) stockant des données de configuration spécifiques.

Un registre définit généralement, à travers des valeurs définies, le comportement d'un contrôleur de périphérique vis-à-vis de son périphérique ou les droits associés à celui-ci. Certaines valeurs n'ont pas d'importance sur le niveau de sécurité mais d'autres sont cruciales. À titre d'exemple, il existe une valeur de registre responsable du verrouillage de la mémoire flash SPI stockant le BIOS, qui empêche donc ainsi son altération depuis un système d'exploitation. L'absence de verrouillage rend possible l'installation d'une porte dérobée non détectable par les antivirus car non présente sur le disque dur.

En parcourant la documentation des chipset Intel, nous comprenons rapidement à quel point il est difficile de verrouiller complètement l'ensemble des composants d'une plate-forme. En effet, le nombre de registres et valeurs possible est très important et ils ont souvent des dépendances entre eux qui ne sont pas forcément simples à identifier. Le bit `a1` d'un registre `A` peut être nativement bien positionné, mais si le bit `b1` du registre `B` est responsable du verrouillage de plusieurs registres dont `A`, alors le bit `a1` pourrait à son tour être modifié alors que celui-ci protège peut-être aussi d'autres registres. Les dépendances sont donc très importantes et complexes à vérifier, et c'est l'un des points forts de l'outil Chipsec.

Lorsque ses résultats sont parcourus, il est absolument nécessaire qu'ils soient tous validés, puisqu'un défaut de verrouillage à un endroit peut entraîner des problèmes en cascade.

Un dernier élément important à préciser est que ces valeurs de registres sont définies généralement via le BIOS dès son chargement au démarrage de la plate-forme. Ainsi, des paramètres de sécurité non conformes nécessitent généralement une mise à jour du BIOS.

Moyens d'accès aux registres de configuration

Registres du CPU : Le processeur expose un certain nombre de registres de configuration internes, qui sont accessibles par plusieurs moyens :

— via la réponse à l'instruction `cpuid` (qui énumère les diverses fonctionnalités supportées par le CPU, comme AES-NI ou encore le bit NX) ;

— via des *Model Specific Register* (MSR), qui sont des registres spécifiques aux processeurs Intel et sont accessibles via les instructions `rdmsr` et `wrmsr`.

Ces deux moyens sont détaillés dans le *Intel 64 and IA-32 Architectures Software Developer's Manual* [12] (Volume 1, chapitre 20 pour `cpuid` et volume 4 pour les MSR).

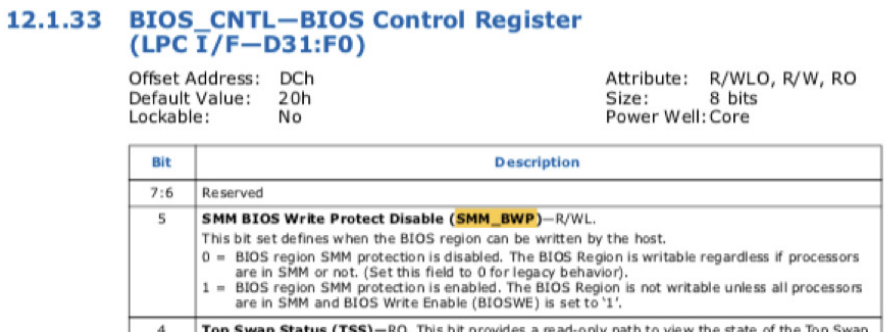
Espace de configuration PCI : Les périphériques PCI Express exposent sur le bus un espace de configuration spécifique (le *PCI Configuration Space*) qui stocke un certain nombre d'informations (telles que le *Vendor ID* et le *Product ID* du périphérique) mais aussi des données de configurations. Le contrôleur mémoire d'une plate-forme Intel récente est ainsi un périphérique PCI Express (situé historiquement dans le *north bridge*, puis dans le PCH, et enfin directement dans le processeur lui-même). La documentation Intel indique les registres présents et configurables, par exemple ici l'accès au registre `BIOS_CNTL` ou *Bios Control Register* (voir Fig. 5).

Offset	Mnemonic	Register Name	Default	Attribute
90h–93h	GEN4_DEC	LPC I/F Generic Decode Range 4	00000000h	R/W
94h–97h	ULKMC	USB Legacy Keyboard / Mouse Control	00002000h	RO, R/WC, R/W
98h–9Bh	LGMR	LPC I/F Generic Memory Range	00000000h	R/W
A0h–CFh		Power Management (See Section 12.8.1)		
D0h–D3h	BIOS_SEL1	BIOS Select 1	00112233h	R/W, RO
D4h–D5h	BIOS_SEL2	BIOS Select 2	4567h	R/W
D8h–D9h	BIOS_DEC_EN1	BIOS Decode Enable 1	FFCFh	R/W, RO
DCh	BIOS_CNTL	BIOS Control	20h	R/WLO, R/W, RO

Fig. 5. Datasheet Intel : Données d'accès au registre `BIOS_CNTL`

Toutes les valeurs contenues dans ces registres telle que `SMM_BWP` par exemple (bit 5) du registre de 8 bits `BIOS_CNTL` sont présentées dans ce document (voir Fig. 6).

Mémoire projetée en mémoire physique : Pour certains périphériques, l'espace de configuration PCI ne suffit pas (ou n'est pas présent, dans le cas de bus différents), et le périphérique lui-même dispose de mémoire avec des données de configuration. L'espace mémoire du périphérique est dit projeté dans l'espace mémoire du processeur : les accès mémoire à



12.1.33 BIOS_CNTL—BIOS Control Register (LPC I/F—D31:F0)

Offset Address: DCh
Default Value: 20h
Lockable: No
Attribute: R/WLO, R/W, RO
Size: 8 bits
Power Well: Core

Bit	Description
7:6	Reserved
5	**SMM BIOS Write Protect Disable (SMM_BWP)**—R/WL. This bit set defines when the BIOS region can be written by the host. 0 = BIOS region SMM protection is disabled. The BIOS Region is writable regardless if processors are in SMM or not. (Set this field to 0 for legacy behavior). 1 = BIOS region SMM protection is enabled. The BIOS Region is not writable unless all processors are in SMM and BIOS Write Enable (BIOSWE) is set to '1'.
4	**Top Swap Status (TSS)**—RO. This bit provides a read-only path to view the state of the Top Swap

Fig. 6. Datasheet Intel : Valeurs du registre `BIOS_CNTL`

cette zone ne sont pas dirigés vers les barrettes mémoires mais vers le périphérique lui-même (*Memory Mapped I/O*, MMIO). L'adresse de base de cette zone mémoire est habituellement stockée dans le *PCI Configuration Space*, dans un registre de type `BAR` (*Base Address Register*).

C'est par exemple le cas du contrôleur SPI permettant d'accéder à la mémoire flash du BIOS, dont la configuration est accessible via le registre `SPIBAR`. La documentation Intel indique alors tous les registres disponibles à cette adresse mémoire via la notation `SPIBAR + Offset` (voir Fig. 7).

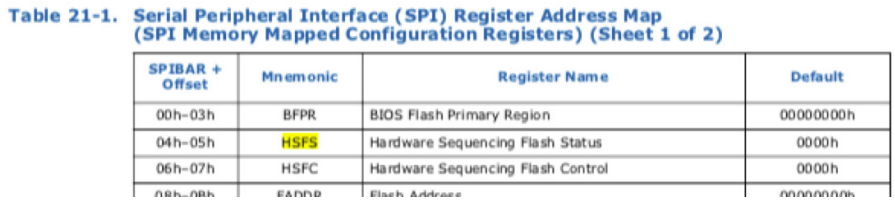


Table 21-1. Serial Peripheral Interface (SPI) Register Address Map (SPI Memory Mapped Configuration Registers) (Sheet 1 of 2)

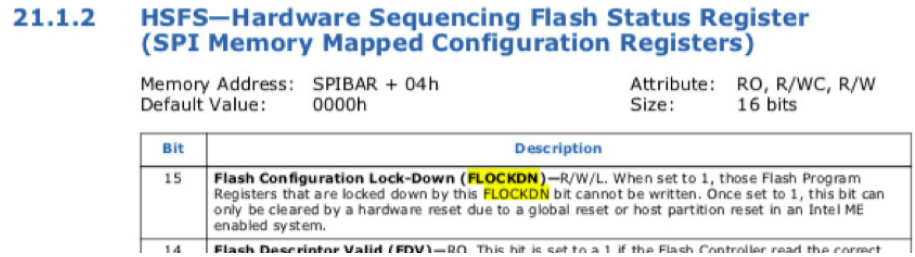
SPIBAR + Offset	Mnemonic	Register Name	Default
00h–03h	BFPR	BIOS Flash Primary Region	00000000h
04h–05h	HSFS	Hardware Sequencing Flash Status	0000h
06h–07h	HSFC	Hardware Sequencing Flash Control	0000h
08h–0Bh	FADDR	Flash Address	00000000h

Fig. 7. Datasheet Intel : registres du contrôleur SPI

Une quarantaine de registres de configuration existe notamment pour la flash SPI, si nous souhaitons vérifier la valeur de `FLOCKDN`, nous retrouvons les informations précises pour accéder à celle-ci et en l'occurrence au sein du registre `HSFS` ou *Hardware Sequencing Flash Status* (voir Fig. 8).

3.4 Extraction et manipulation des registres avec Chipsec

L'un des intérêts de Chipsec est de fournir une documentation technique (adresses et valeurs de variables, ...) propre à chaque plate-forme et évitant à un analyste de devoir lire les spécifications de celle-ci pour permettre de



21.1.2 HSFS—Hardware Sequencing Flash Status Register (SPI Memory Mapped Configuration Registers)

Memory Address: SPIBAR + 04h — Attribute: RO, R/WC, R/W
Default Value: 0000h — Size: 16 bits

Bit	Description
15	**Flash Configuration Lock-Down (FLOCKDN)**—R/W/L. When set to 1, those Flash Program Registers that are locked down by this FLOCKDN bit cannot be written. Once set to 1, this bit can only be cleared by a hardware reset due to a global reset or host partition reset in an Intel ME enabled system.
14	**Flash Descriptor Valid (FDV)**—RO. This bit is set to a 1 if the Flash Controller read the correct

Fig. 8. Datasheet Intel : champs du registre HSFS

manipuler les registres. Ainsi des fichiers de configuration propres à chaque modèle de chipset (format XML) spécifient les adresses des registres et de leurs variables ainsi que les adresses mémoire MMIO.

L'accès aux informations de configuration nécessite en général des accès bas niveau et donc privilégiés, que ce soit via des instructions (`cpuid` ou `rdmsr`) ou des accès à la mémoire physique ou au bus PCI. Chipsec dispose donc d'un driver (module kernel `chipsec.ko` sous Linux) pour effectuer les opérations privilégiées, qui sert de relais au code Python s'exécutant en espace utilisateur (mais avec les droits `root`). L'appel au driver est alors initié via un IOCTL sur le nœud de périphérique créé par le driver. Celui-ci transfère ensuite les appels IOCTL aux périphériques ciblés.

Pour chaque type d'accès (PCI/MMIO/CPU/...), un module Python est implémenté pour permettre un accès ou une écriture des données de configuration. Un module Python existe également pour manipuler les registres de la flash SPI du BIOS (MMIO) dans le but d'en extraire le contenu. Par souci de place, nous ne détaillerons pas dans ce papier le fonctionnement précis de Chipsec et nous invitons le lecteur à lire le MISC de mai/juin 2020 dans le lequel nous proposons un article sur ce sujet. L'idée de l'article est d'une part d'expliquer comment fonctionne techniquement Chipsec et d'autre part en quoi Chipsec est puissant et pourquoi il est inutile de tenter une extraction de configuation bas-niveau soi-même (sous Linux).

3.5 Détection des plate-formes vulnérables via Chipsec

Nous n'allons pas présenter la totalité des vulnérabilités testées par Chipsec mais uniquement celles qui ont eu un certain retentissement et donc généralement ayant un impact important sur la plate-forme. Vous pouvez toutefois retrouver un document complet détaillant les tests menés

par Chipsec sur le github de l'ANSSI [1] et indiquant également les sorties que nous sommes censés obtenir pour que la plate-forme soit conforme.

Détection des vulnérabilités propres à la Flash SPI Chipsec est en mesure de détecter les vulnérabilités liées à la sécurisation de la mémoire flash SPI accueillant le firmware du BIOS à l'aide de plusieurs modules :

`chipsec.modules.common.bios_wp` et `bios_smi` : Le registre le plus connu aujourd'hui est celui protégeant l'écriture des régions de la flash SPI une fois le système démarré : `BIOS_CNTL` du PCH. Contenant 3 bits d'une importance cruciale pour protéger le firmware des altérations, il est probablement le registre le plus exploité par les vulnérabilités découvertes jusqu'à maintenant.

```
[*] running module: chipsec.modules.common.bios_wp
[x][ ================================================================================
[x][ Module: BIOS Region Write Protection
[x][ ================================================================================
[*] BC = 0x2A << BIOS Control (b:d.f 00:31.0 + 0xDC)
    [00] BIOSWE           = 0 << BIOS Write Enable
    [01] BLE              = 1 << BIOS Lock Enable
    [02] SRC              = 2 << SPI Read Configuration
    [04] TSS              = 0 << Top Swap Status
    [05] SMM_BWP          = 1 << SMM BIOS Write Protection
[+] BIOS region write protection is enabled (writes restricted to SMM)
```

Fig. 9. Contrôle du registre `BIOS_CNTL` via `bios_wp`

Chipsec vérifie notamment ces valeurs à travers le module `bios_wp` (Fig. 9) en vérifiant les bits :

`BIOSWE` *BIOS Write Enable* si ce bit est à 1, un accès à la flash SPI en écriture sera possible depuis le système d'exploitation (avec les privilèges du *ring 0*), sinon l'écriture est impossible ;

`BLE` *BIOS Lock Enable*

1 à chaque tentative de modification du bit `BIOSWE`, une SMI générée par le PCH empêchera tout simplement l'opération (le bit `BIOSWE` est remis à 0 automatiquement par du code s'exécutant en SMM) ;

0 depuis l'espace noyau, il sera possible de modifier le bit `BIOSWE` pour déverrouiller la flash SPI et donc altérer son contenu.

Il est légitime de se demander pourquoi ne pas verrouiller totalement la flash SPI, sans possibilité de déverrouillage, afin d'éviter toute modification

ultérieure par un malware. La raison principale est la nécessité de prévoir des mises à jour. En effet, la flash SPI comprend un certain nombre d'éléments logiciels (comme le BIOS) et de configuration qui doivent pouvoir être mis à jour. Il faut donc prévoir des modes légitimes d'écritures dans la mémoire flash, après validation.

En pratique, suite à une demande de mise à jour et au redémarrage de la machine, le code du BIOS se charge de passer `BIOSWE` à 1 provoquant ainsi une SMI (dans le cas où `BLE` vaut 1) et déclenchant alors le passage en SMM et l'exécution d'un code spécifique contrôlé par le BIOS lui-même et donc à priori légitime. Celui-ci se charge de vérifier la validité de la mise à jour pour l'appliquer par la suite.

La protection `BLE` a cependant déjà pu être contournée à plusieurs reprises notamment à travers l'attaque SpeedRacer [17]. Celle-ci consiste à être plus rapide que le contrôle en jouant avec plusieurs cœurs du processeur afin d'écrire dans la région du BIOS avant qu'il y ait déclenchement de la SMI.

Une autre attaque encore plus astucieuse nommée Sandman [16] consiste quant à elle à désactiver complètement les SMI pour empêcher le passage en SMM et permettre la modification persistante de `BIOSWE`. Concrètement, cette vulnérabilité peut être exploitée pour contourner *Secureboot* par exemple [15].

Chipsec vérifie naturellement que les SMI sont activées via le module `bios_smi` (Fig. 10).

```
[*] running module: chipsec.modules.common.bios_smi
[x][ ================================================================================
[x][ Module: SMI Events Configuration
[x][ ================================================================================
[+] SMM BIOS region write protection is enabled (SMM_BWP is used)

[*] Checking SMI enables..
    Global SMI enable: 1
    TCO SMI enable   : 1
[+] All required SMI events are enabled

[*] Checking SMI configuration locks..
[+] TCO SMI configuration is locked (TCO SMI Lock)
[+] SMI events global configuration is locked (SMI Lock)

[+] PASSED: All required SMI sources seem to be enabled and locked
```

Fig. 10. Contrôle du verrouillage de SMI via `bios_smi`

Une protection plus sûre est donc nécessaire et c'est ce que propose `SMM_BWP`. Ce bit du registre `BIOS_CNTL`, vérifié par Chipsec, permet ainsi de protéger le BIOS d'une telle attaque en protégeant sa modification même si le contrôle SMI n'est pas configuré ou échoue :

`SMM_BWP` *SMM BIOS Write Protect Disable* si ce bit est à 1, la région SPI propre au BIOS ne pourra être modifiée que si le processeur s'exécute en mode SMM et donc que le code provient du BIOS (typiquement utilisé lors d'une mise à jour pour réécrire le contenu de la mémoire flash avec le nouveau BIOS).

Alors que les attaques présentées jusque-là ne nécessitaient qu'un accès privilégié au noyau (*ring 0*), il est nécessaire, avec cette protection en place, d'exécuter du code en espace SMM pour altérer le BIOS, compliquant ainsi les exploitations. L'accès à un tel espace nécessite d'une part une élévation privilège à partir de l'espace noyau et d'autre part une exécution de code arbitraire en SMM [25].

`chipsec.modules.common.uefi.s3bootscript` La question suivante est donc de savoir s'il est possible de désactiver cette nouvelle protection en passant `SMM_BWP` de 1 à 0. En 2015 des chercheurs sont parvenus à infecter un firmware EFI depuis le mode noyau alors que la flash SPI était protégée par les bits de protection proposés par le registre `BIOS_CNTL` [26] [24]. Deux vulnérabilités découvertes par les chercheurs et propres au mode de démarrage *ACPI S3 Resume* (sortie de veille de la plate-forme) sont exploitées dans leur scénario.

Durant la phase de démarrage normal et notamment durant la phase `DXE` (chargement des drivers EFI), l'état du CPU et du chipset, comprenant donc les registres du PCH tel que `BIOS_CNTL`, sont sauvegardés en mémoire dans une structure appelée *UEFI boot script table*. Cette table est ensuite utilisée lors de la phase d'une sortie de mise en veille (*S3 resume*) afin de restaurer plus rapidement l'état de la platforme, en évitant notamment de devoir recharger tous les drivers EFI.

La première vulnérabilité (la seconde sera abordée plus tard) exploite le fait que la sauvegarde de la table est réalisée durant la phase de boot **avant que les registres aient été complètement verrouillés** par le firmware EFI. Ainsi, au réveil de la plate-forme plusieurs registres dont `BIOS_CNTL` ont un état déverrouillé et donc une valeur de `BIOSWP` ou `SMM_BWP` modifiables depuis le mode noyau. L'altération du firmware EFI devient alors possible après avoir reparamétré les valeurs du registre.

Des mesures de sécurité ont depuis été poussées par Intel en recommandant par exemple d'effectuer la sauvegarde de l'état du CPU et du

chipset **après** la demande de mise en veille *S3 suspend* et donc lorsque les registres sont correctement verrouillés. Il reste cependant beaucoup de plate-formes vulnérables dans la nature étant donné la faible quantité d'utilisateurs ou administrateurs sensibilisés aux mises à jour de BIOS.

Chipsec vérifie notamment si la plate-forme testée est vulnérable via le module `s3bootscript` et le module `tools.uefi.s3script_modify` aide également à effectuer des tests concrets de modification des scripts.

Pour renforcer davantage la sécurité de la flash SPI, plusieurs registres supplémentaires du contrôleur SPI, `PRx` (*Protected Range*), spécifiques à chaque région de la flash SPI, ont vu le jour pour définir les droits d'écriture (`WP`) et lecture (`RP`) sur chacune des régions et ainsi empêcher toute altération de la cible prioritaire, le contenu de la flash SPI. Cette protection, dont le code fonctionne indépendamment du mode SMM, empêche l'écriture des régions de la flash via un code malveillant s'exécutant dans ce mode.

Le module Chipsec `bios_wp` (Fig. 11) inclut également la vérification de ces registres.

```
[*] BIOS Region: Base = 0x00580000, Limit = 0x00FFFFFF
SPI Protected Ranges
------------------------------------------------------------
PRx (offset) | Value    | Base     | Limit    | WP? | RP?
------------------------------------------------------------
PR0 (74)     | 00000000 | 00000000 | 00000000 | 0   | 0
PR1 (78)     | 8FFF0FF0 | 00FF0000 | 00FFFFFF | 1   | 0
PR2 (7C)     | 00000000 | 00000000 | 00000000 | 0   | 0
PR3 (80)     | 00000000 | 00000000 | 00000000 | 0   | 0
PR4 (84)     | 00000000 | 00000000 | 00000000 | 0   | 0

[!] SPI protected ranges write-protect parts of BIOS region (other parts of BIOS can be modified)
```

Fig. 11. Contrôle des registres *Protected Range* via `bios_wp`

Malheureusement, selon les constructeurs, il n'est pas rare que cette protection soit désactivée ou activée uniquement pour certaines régions, il est donc là encore très important de vérifier en pratique qu'elle est mise en place correctement.

`chipsec.modules.common.spi_lock` De nouveau, il existe un moyen de contourner cette dernière protection. Les registres `PRx` étant inclus dans la configuration du contrôleur SPI, ils sont protégés par le biais du registre du contrôleur SPI `HSFS` (*Hardware Sequencing Flash Status and Control*). Ainsi si le bit `FLOCKDN` de celui-ci est à 0 alors les registres propres à la flash SPI (tels que `PRx`) ne sont plus protégés en écriture et peuvent donc subir des modifications. Chipsec vérifie donc cette valeur via le module `spi_lock` (Fig. 12).

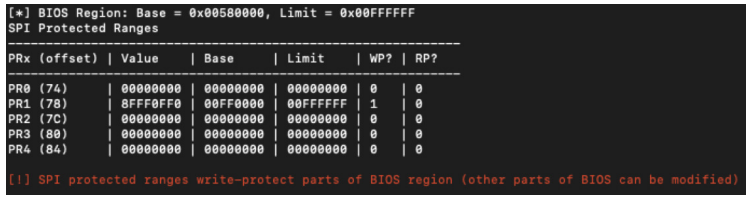


Fig. 12. Contrôle du verrouillage de la configuration du contrôleur SPI

Les différents registres et bits vérifiés ont été consolidés dans un mindmap (Fig. 13) et restent disponibles sur le github de l'ANSSI [6].

Détection des vulnérabilités propres à la SMRAM Comme on l'a vu ci-dessus, le SMM est un mode particulièrement privilégié du processeur, et le code qui s'exécute à ce niveau doit avoir un niveau d'intégrité important.

Chipsec est en mesure de détecter les vulnérabilités liées à la sécurisation de la mémoire SMRAM à l'aide de plusieurs modules.

chipsec.modules.common.smm Le mécanisme de sécurité empêchant l'accès à l'espace SMRAM depuis un mode autre que SMM est défini dans le registre `SMRAC` (*System Management RAM Control Register*) du contrôleur mémoire, accessible en PCI.

Les deux bits qui nous intéressent sont `D_OPEN` (*SMM Space Open*) couplé au bit `D_LCK` (*SMM Space Locked*) qui empêche la modification de celui-ci (ainsi que d'autres registres importants). Le module `smm` (Fig. 14) s'assure que `D_LCK` soit à 1 et `D_OPEN` à 0.

Des recherches menées sur le sujet il y a plus de dix ans [21] ont montré qu'il était tout de même possible d'accéder à la SMRAM malgré les bits de `SMRAMC` correctement positionnés. En effet, sur une architecture disposant d'un *north bridge*, les chercheurs ont pu déposer leur propre code SMM qui était exécuté suite au déclenchement d'une SMI. Le code malveillant ainsi exécuté en SMM pourra contourner des mécanismes de sécurité système. L'exploitation, très complexe, est possible du fait que le registre `SMRAMC` peut être modifié depuis le mode noyau. Sur les nouvelles plate-formes disposant d'un PCH, le contrôle par `SMRAMC` n'étant plus réalisé par le *north bridge* mais par le CPU, l'exploitation ne semble plus possible.

Une deuxième vulnérabilité [8], découverte en 2009 par les mêmes chercheurs et contournant également `SMRAMC`, exploite cette fois-ci le cache du CPU et le fait qu'il y stocke du code SMM. Il est ainsi possible depuis

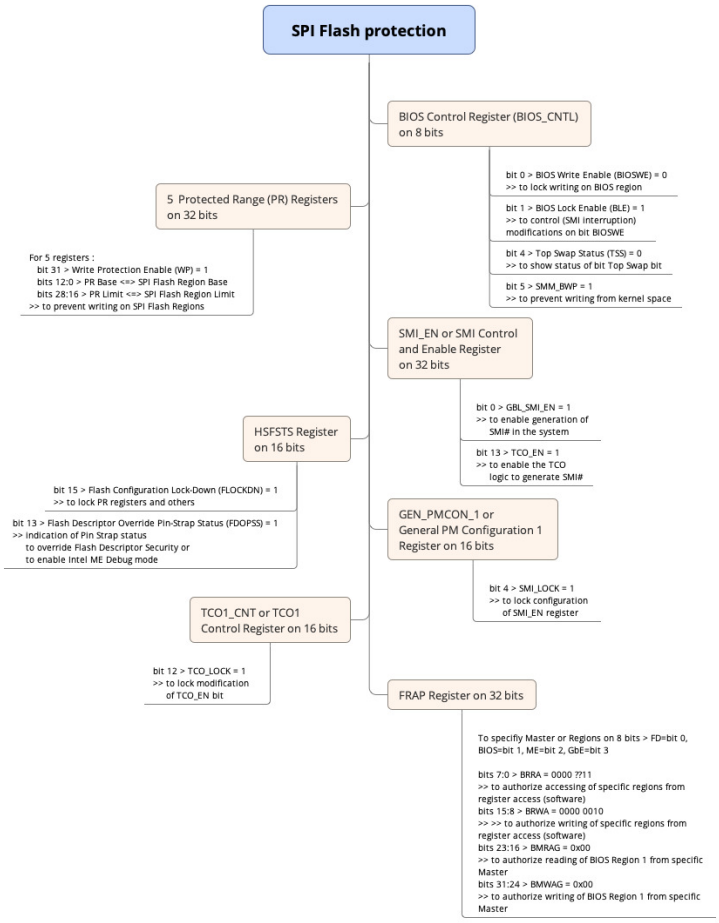


Fig. 13. Registres et bits protégeant la Flash SPI

le mode noyau de forcer le CPU à exécuter une charge malveillante en modifiant ce cache suite à une SMI.

Ces deux vulnérabilités ont fait l'objet d'une présentation très complète [22] ainsi que d'un article Phrack [7].

`chipsec.modules.common.smrr` Suite à ces vulnérabilités qui ont fait beaucoup de bruit, Intel a ajouté un nouveau registre processeur pour effectuer des contrôles de l'état de la SMRAM. Ainsi le MSR `SMRR` (*System Management Range Register*) a vu le jour et apporte les sécurités suivantes :

— `SMRR` détermine la manière dont sera mis en cache la SMRAM ;

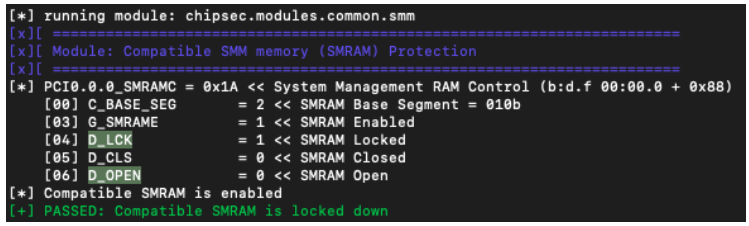


Fig. 14. Vérification du registre `SMRAMC`

— `SMRR` ne peut être modifié que par du code s'exécutant en mode SMM ;
— si le CPU est en mode SMM, il vérifie alors les valeurs de `SMRR` pour déterminer comment la SRAM doit être mis en cache ;
— si le CPU n'est pas en mode SMM, il considère alors que la mémoire SMRAM ne doit pas être mise en cache et tous les accès en lecture renvoient une valeur fixe (et inexploitable) alors que ceux en écriture sont ignorés.

Le registre `SMRR`, configuré par le BIOS au démarrage, est vérifié par le module Chipsec `smrr` (Fig. 15).

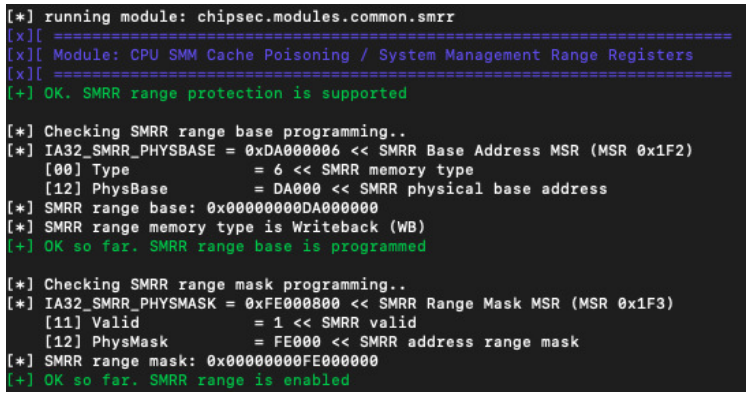


Fig. 15. Vérification du registre `SMRR`

Des tests sont également réalisés pour s'assurer qu'il n'est pas possible d'accéder à la mémoire SMRAM (Fig. 16) depuis le mode noyau.

`chipsec.modules.smm_dma` Sur les chipsets et processeurs récents, et dans le but de protéger la plate-forme des attaques DMA (accès direct à la mémoire physique) ciblant la SMRAM depuis du matériel physique

```
[*] Trying to read memory at SMRR base 0xDA000000..
[+] PASSED: SMRR reads are blocked in non-SMM mode
```

Fig. 16. Vérification de l'accès au registre SMRR

(via PCIe, Thunderbolt etc.), le mécanisme de protection `TSEG` (*Extended SMRAM Space*) a été implémenté. TSEG définit un espace mémoire physique (Fig. 17) protégé (incluant la SMRAM). Plusieurs registres (activation, adresse de base, taille etc.) du controleur mémoire permettent de configurer finement cette zone.

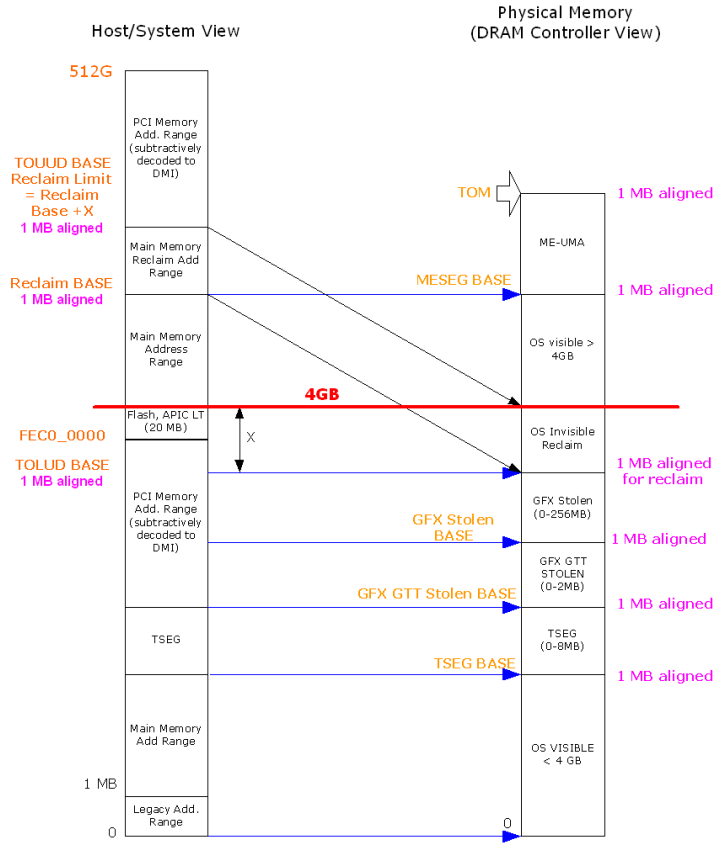


Fig. 17. Agencement de l'espace mémoire sur plate-forme Intel

Chipsec s'assure (Fig. 18), à travers son module `smm_dma`, que l'espace mémoire TSEG couvre complètement SMRAM et que le bit `LOCK` de `TSEGMB`, permettant de verrouiller cette configuration, est positionné à 1.

```
[*] running module: chipsec.modules.smm_dma
[x][ ================================================================================
[x][ Module: SMM TSEG Range Configuration Check
[x][ ================================================================================
[*] TSEG      : 0x00000000DA000000 - 0x00000000DBFFFFFF (size = 0x02000000)
[*] SMRR range: 0x00000000DA000000 - 0x00000000DBFFFFFF (size = 0x02000000)

[*] checking TSEG range configuration..
[+] TSEG range covers entire SMRAM
[+] TSEG range is locked
[+] PASSED: TSEG is properly configured. SMRAM is protected from DMA attacks
```

Fig. 18. Vérification de la configuration de TSEG

Un autre élément important contrôlé par Chipsec et abordé plus haut est la valeur du bit `D_LCK` de `SMRAMC` qui doit être positionnée à 1 afin de verrouiller une partie de la configuration de l'espace mémoire physique.

La deuxième vulnérabilité liée au *S3 suspend/resume* et mentionnée précédemment exploite le fait que la table soit sauvegardée en RAM et non en SMRAM mais également le fait que le registre `TSEGMB` soit déverrouillé (comme `BIOS_CNTL`) suite à une sortie de veille. Ainsi, depuis un accès au noyau, l'accès à la RAM n'étant pas spécifiquement protégée, il est possible de compromettre la structure de la table en lui indiquant par exemple de charger une configuration spécifique et donc un code arbitraire en SMM. Des mesures palliatives ont été poussées dans le kit de développement EFI (EDK2), tel que le concept « LockBox » qui propose de sauvegarder la table en SMRAM, donc non accessible depuis le mode noyau.

Là encore, les différents registres et bits vérifiés pour la protection de la SMRAM ont été consolidés dans un mindmap (Fig. 19) et restent disponibles sur le github de l'ANSSI [5].

Détection des vulnérabilités propres aux options CPU Certaines vulnérabilités proviennent de l'implémentation matérielle du CPU lui-même, ou bien de fonctionnalités dont il dispose et qui pourraient être détournées à des fins malveillantes (par exemple des fonctions de *debug*).

Chipsec est en mesure de détecter certaines de ces vulnérabilités à l'aide de plusieurs modules :

`chipsec.modules.common.cpu.spectre_v2` Ce module est dédié aux vulnérabilités *side channel* découvertes à partir de 2017 dans les CPU

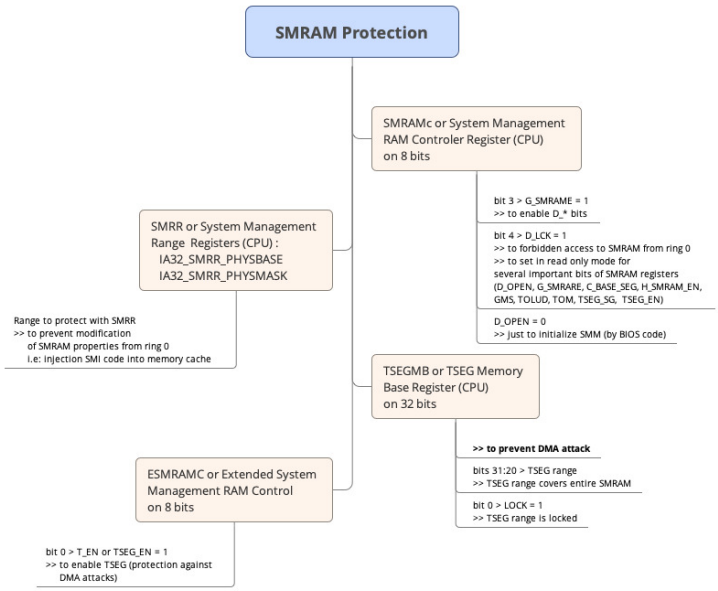


Fig. 19. Registres et bits protégeant la SMRAM

Intel, à commencer par *Spectre variant 2* (CVE-2017-5715, aussi appelée *Branch Target Injection*). Sans rentrer dans le détail des vulnérabilités *Spectre*, Intel a introduit des correctifs contre diverses vulnérabilités dans des microcodes (et dans le matériel pour certaines générations suivantes de CPU), sous la forme de plusieurs technologies : *Indirect Branch Restricted Speculation* (IBRS), *Indirect Branch Predictor Barrier* (IBPB), *Single Thread Indirect Branch Predictors* (STIBP), *Enhanced IBRS* (IBRS_ALL). Ces fonctionnalités sont exposées par le processeur au système d'exploitation (ou à l'hyperviseur) qui doit ensuite les prendre en compte, par exemple pour nettoyer des buffers lors de changements de contexte ou de niveau de privilège.

Le processeur expose ces informations via `cpuid` pour IBRS, IBPB et STIBP tandis que IBRS_ALL est exposé dans le MSR `IA32_ARCH_CAPABILITIES` (*Enumeration of Architectural Features*).

Ce MSR accessible uniquement en lecture a été introduit par Intel dans les processeurs récents afin d'exposer au système d'exploitation les fonctionnalités de défense contre les diverses attaques matérielles. Sa présence doit donc être d'abord confirmée (à l'aide de la réponse à `cpuid`) avant de consulter son contenu.

chipsec.modules.debugenabled Les plate-formes Intel disposent de fonctionnalités permettant, sous certaines conditions, d'accéder à des fonctionnalités de debug au sein même du processeur ou d'autres composants comme le chipset. Ces fonctionnalités de debug étaient initialement réservées à des systèmes en cours de développement, mais il est devenu possible d'utiliser ces fonctionnalités sur des machines de série (afin de pouvoir identifier et corriger des problèmes une fois sortie des lignes d'assemblage d'Intel).

Cette technologie reposait avant 2015 sur le *Closed Chassis Adapter* (CCA), un outil matériel propriétaire Intel permettant de s'interfacer avec une machine à debugger. À partir de 2015, les systèmes *Skylake* et ultérieurs disposent de la technologie *Direct Connect Interface* (DCI) qui s'appuie sur le bus USB3 et ne nécessite pas un composant matériel spécifique.

Il est évident qu'un accès bas niveau tel que le JTAG à des machines en production est extrêmement dangereux, puisque ce genre d'accès permet de contourner tout type de contrôle que le système d'exploitation pourrait mettre en place. Le debug matériel est donc désactivé par défaut, et doit être explicitement activé, ce qui peut se faire de trois façons :

— via le positionnement d'une variable du firmware UEFI (`HDCIEN`, *DCI Enable*) qui nécessite de reflasher le BIOS (et donc d'avoir un accès en écriture à la flash SPI) et invalide le *SecureBoot* ;
— via le positionnement d'un PCH strap, une variable située dans la région de configuration de la flash SPI, qui nécessite donc l'accès en écriture à la flash SPI mais n'invalide pas le *SecureBoot* ;
— après le démarrage, via le *DCI Control Register* (`ECTRL`), accessible via les registres privés de configuration du PCH : le bit 4, `HDCIEN` (*Host DCI Enable*) permet d'activer l'interface et donc le debug.

Ces moyens ont été identifiés par des chercheurs de Positive Technologies [11] et présentés au 33ème Chaos Computer Congress en 2016 [10].

Ces fonctionnalités de debug peuvent être verrouillées au travers d'un nouveau MSR, `IA32_DEBUG_INTERFACE` (*Silicon Debug Feature Control*) qui comprend trois bits qui nous intéressent ici :

0 *Enable* permet d'activer (`1`) ou désactiver (`0`) les fonctionnalités de debug ;

30 *Lock* verrouille le registre et en particulier le premier bit, il est automatiquement positionné à la première SMI ;

31 *Debug Occured* est en lecture seule et positionné par le CPU lui-même si un debug a déjà été initié.

Sur des machines standard, le BIOS doit normalement s'assurer que le bit *Enable* est à 0, puis passer le *Lock* à 1 afin d'éviter tout debug subséquent.

Malheureusement, les chercheurs de Positive Technologies ont remarqué que sur un certain nombre de machines testées le bit *Lock* n'était pas positionné (manuellement par le BIOS ou bien à la première SMI), ce qui permettait donc de réactiver la fonctionnalité. Ce défaut de configuration a reçu le CVE-2017-5684 et a touché Intel, Lenovo, Asus et Gigabyte.

Les différents registres et bits vérifiés ont été consolidés dans un mindmap (Fig. 20) et restent disponibles sur le github de l'ANSSI [4].

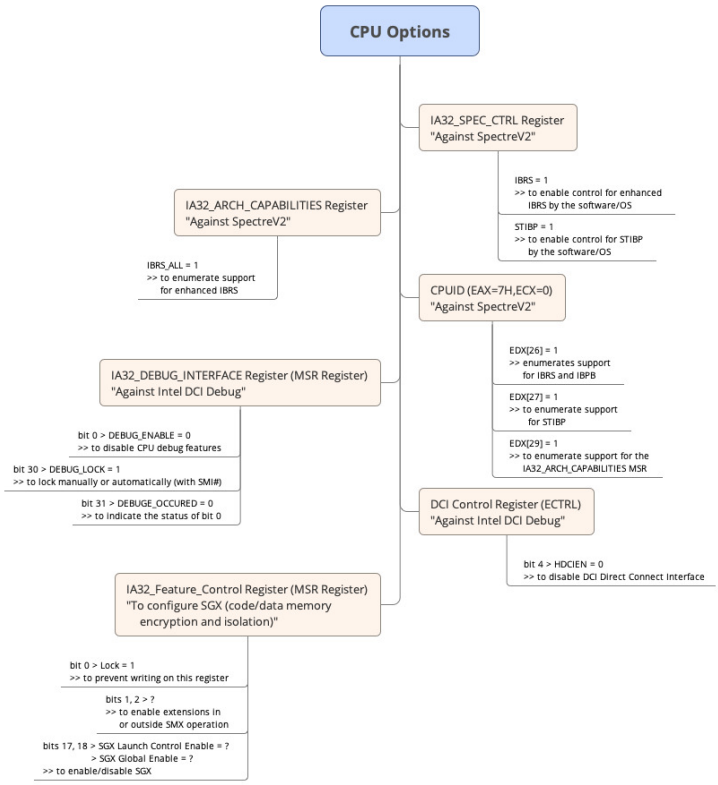


Fig. 20. Registres et bits protégeant le CPU

4 Conclusion

Aujourd'hui, les nombreuses fuites de documents appartenant à des vendeurs d'exploits (Hacking Team, NSO Group, ...) ou à des gouvernements (`IRATEMONK`, `DEITYBOUNCE`, *Vault 7*...) prouvent que les menaces touchant le matériel des plate-formes sont réelles. Des implants BIOS sont très certainement utilisés en pratique mais il est bien rare de les identifier tant leur niveau de complexité est élevé, sans compter que peu de monde tente de les détecter.

Une attention accrue portée aux enjeux des interfaces entre le matériel et le logiciel, et la prise en compte du matériel dans les analyses de risque semblent inéluctables. Ainsi, l'ajout d'exigences de sécurité sur le matériel acquis par le gouvernement français a été effectué et sont proposées à tous.

Ces exigences sont complétées par des tests réels sur les plates-formes soumises, avec des moyens là encore proposés à tous et reposant sur Chipsec. Cet outil pratique est aujourd'hui un incontournable dans l'évaluation de la sécurité des plate-formes Intel, il serait dommage de ne pas l'exploiter.

La complexité des systèmes actuels, illustrée dans cet article, montre le défi à atteindre pour augmenter le niveau de sécurité des utilisateurs. Cependant, le niveau de maturité de l'industrie semble progresser petit à petit, que ce soit les fabricants de processeurs ou les constructeurs avec qui nous avons pu échanger dans le cadre de la rédaction des exigences. Ainsi, suite aux nombreuses vulnérabilités dans ses CPU, Intel aussi fait des efforts concernant la sécurité et le montre en soutenant le projet Chipsec et en communiquant sur les vulnérabilités affectant leurs CPU [14]. Intel propose également sur ses plate-formes récentes des solutions rassurantes telles que Intel Boot Guard et Intel Bios Guard protégeant la manipulation de toute la séquence de démarrage UEFI en vérifiant l'intégrité des codes du firmware UEFI (pendant les phases SEC, PEI, DXE et BDS) fournis par les constructeurs (les signatures de confiance étant stockées de manière sécurisée via des fusibles du CPU ou dans un TPM). Ainsi, avec Secure Boot, la chaîne de démarrage est protégée de bout en bout depuis le chargement du premier code UEFI jusqu'au lancement du chargeur de démarrage UEFI et du noyau du système d'exploitation. Par ailleurs, les constructeurs et fournisseurs de BIOS sont de plus en plus attentifs à fournir un paramétrage sécurisé de leurs systèmes.

En ce qui concerne les moyens de détection pour les BIOS déjà compromis, comme a pu le constater ESET suite à l'analyse de LoJax, ils sont difficiles à mettre en oeuvre. Les premières pistes envisageables pour

répondre à cette problématique de détection pourraient être, comme le propose le service open-source de distribution de mises à jour firmware fwupd [3], de constituer une base de BIOS assez volumineuse pour sortir des statistiques exploitables et identifier des charges malveillantes.

Références

1. ANSSI. Modules chipsec. https://github.com/ANSSI-FR/chipsec-check/blob/master/doc/output/modules_chipsec.pdf.
2. ANSSI. Tools for testing requirements. https://github.com/anssi-fr/chipsec-check.
3. ANSSI. Exigences de sécurité matérielle pour plate-formes x86. https://www.ssi.gouv.fr/guide/exigences-de-securite-materielles/, 2019.
4. ANSSI. Mindmap cpu protection. https://github.com/ANSSI-FR/chipsec-check/blob/master/doc/output/CPU%20Options.pdf, 2019.
5. ANSSI. Mindmap smram protection. https://github.com/ANSSI-FR/chipsec-check/blob/master/doc/output/SRAM%20Protection.pdf, 2019.
6. ANSSI. Mindmap spi protection. https://github.com/ANSSI-FR/chipsec-check/blob/master/doc/output/SPI%20Protection.pdf, 2019.
7. D0nAnd0n BSDaemon, coideloko. System management mode hack using smm for "other purposes. http://phrack.org/issues/65/7.html, 2008.
8. Loic Duflot, Olivier Levillain, Benjamin Morin, and Olivier Grumelard. Getting into the smram : Smm reloaded. 2009.
9. ESET. Lojax // first uefi rootkit found in the wild, courtesy of the sednit group. https://cdn1.esetstatic.com/ESET/US/resources/datasheets/ESETus-datasheet-lojax.pdf, 2018.
10. Maxim Goryachy and Mark Ermolov. Taping into the core. 2016.
11. Maxim Goryachy and Mark Ermolov. Where there's a jtag, there's a way : obtaining full system access via usb. 2017.
12. Intel. Intel® 64 and ia-32 architectures software developer's manuals. Technical report.
13. Intel. Platform security assessment framework. https://github.com/chipsec/chipsec.
14. INTEL. Insights intel. https://software.intel.com/security-software-guidance/insights, 2020.
15. Corey Kallenberg, Xeno Kovah, John Butterworth, and Sam Cornwell. All your boot are belong to us. 2014.
16. Corey Kallenberg, Xeno Kovah, John Butterworth, and Sam Cornwell. SENTER Sandman : Using Intel TXT to Attack BIOSes. 2014.
17. Corey Kallenberg and Rafal Wojtczuk. Speed racer : Exploiting an intel flash protection race condition. 2015.

3. https ://fwupd.org

18. Kaspersky. Absolute computrace revisited. https://securelist.com/absolute-computrace-revisited/58278.
19. Kaspersky. Black hat 2014 : Absolute computrace revisited. https://www.blackhat.com/docs/us-14/materials/us-14-Kamlyuk-Kamluk-Computrace-Backdoor-Revisited.pdf.
20. Lenovo. Unintended computrace activation. https://support.lenovo.com/fr/fr/solutions/ht105220.
21. Daniel Etiemble Loic Duflot and Olivier Grumelard. Using cpu system management mode to circumvent operating system security functions. https://www.researchgate.net/publication/241643659_Using_CPU_System_Management_Mode_to_Circumvent_Operating_System_Security_Functions, 2006.
22. Olivier Levillain Benjamin Morin Loic Duflot and Olivier Grumelard. System management mode design and security issues. https://www.ssi.gouv.fr/uploads/IMG/pdf/IT_Defense_2010_final.pdf, 2010.
23. John Loucaides and Yuriy Bulygin. Platform security assessment with CHIPSEC. 2014.
24. Dmytro Oleksiuk. Exploiting uefi boot script table vulnerability. http://blog.cr4.sh/2015/02/exploiting-uefi-boot-script-table.html, 2015.
25. Dmytro Oleksiuk. Exploiting ami aptio firmware on example of intel nuc. http://blog.cr4.sh/2016/10/exploiting-ami-aptio-firmware.html, 2016.
26. Rafal Wojtczuk and Corey Kallenberg. Attacks on UEFI security. 2015.