Master 2 – Cybersécurité – Année 2024/2025

Sécurité matérielle et protection des firmwares : menaces de bas niveau et architectures de défense

Étude de l'utilisation combinée du TPM 2.0, du Secure Boot et de protections hardware pour garantir l'intégrité du firmware

Ivan KRIVOKUCA

Maître d'apprentissage : Monsieur Luc BOUGANIM

Tuteur enseignant : Professeur Patrice MARTIN

Président du jury : Professeur Osman SALEM

Établissement : Université Paris Cité

Entreprise : INRIA

Remerciement

Je souhaite tout d'abord remercier mon tuteur enseignent Patrice MARTIN, pour sa disponibilité, ses conseils avisés et son accompagnement tout au long de ce mémoire.

Ma reconnaissance va également à mon maître d'apprentissage Luc BOUGANIM, qui m'a accueilli au sein de l'INRIA et plus particulièrement dans l'équipe de recherche PETRUS et qui m'a offert l'opportunité d'appliquer mes connaissances dans un contexte professionnel stimulant.

Je tiens à remercier chaleureusement l'ensemble de l'équipe PETRUS pour leur accueil, leur patience et leur bienveillance. Travailler à leurs côtés a été une expérience aussi enrichissante sur le plan professionnel qu'humain.

Mes remerciements s'adressent également à l'équipe pédagogique de l'Université Paris Cité pour la qualité de leur enseignement et leur accompagnement tout au long de ce master.

Enfin, je ne saurais oublier ma famille et mes amis pour leur soutien inconditionnel et leurs encouragements constants.

À toutes ces personnes qui ont contribué à faire de ces deux années une expérience aussi formatrice qu'épanouissante, je tiens à exprimer ma plus profonde gratitude

Résumé

Ce mémoire examine l'évolution des cyberattaques ciblant les couches basses des systèmes informatiques et analyse comparativement les mécanismes de protection déployés sur les plateformes x86/x64 et les systèmes embarqués (ARM/RISC-V), avec un focus particulier sur le Trusted Platform Module 2.0 (TPM 2.0).

Notre étude établit que la sécurité informatique, historiquement concentrée sur les couches logicielles supérieures, révèle ses limites face à l'émergence d'attaques visant le firmware et le matériel. L'analyse des menaces démontre une évolution vers des techniques d'exploitation persistantes et furtives, telles que les rootkits UEFI, les bootkits contournant le Secure Boot, les attaques par corruption mémoire, les injections de fautes matérielles, et les exploitations d'interfaces. Pour les systèmes embarqués et IoT, nous identifions des vulnérabilités spécifiques liées aux contraintes énergétiques et aux longues durées de vie opérationnelle de ces appareils.

Notre analyse comparative des architectures de protection révèle des différences entre ces dites plateformes. Les systèmes x86/x64 privilégient des solutions comme le Secure Boot, des sécurités intégrées en plus dans les processeurs, et l'intégration du TPM sous diverses formes (dTPM, fTPM, vTPM). Les systèmes embarqués adoptent des approches adaptées à leurs contraintes, avec des architectures spécifiques. Sans oublier l’ajout des Secure Elements et enclaves sécurisées. Ces mécanismes s'articulent autour de principes fondamentaux : racines de confiance matérielles et isolation des environnements d'exécution.

L'étude approfondie du TPM 2.0 met en lumière son rôle central dans la sécurisation du processus de démarrage et la protection des données sensibles via ses fonctionnalités de mesure d'intégrité, d'attestation et de scellement cryptographique. Cependant, notre analyse critique identifie des vulnérabilités significatives : faiblesses d'implémentation, attaques par canaux auxiliaires et contournements pratiques. Ces limitations remettent en question l'efficacité du TPM comme solution unique de protection.

Ce travail conclut que malgré l'importance du TPM 2.0 dans l'établissement d'une chaîne de confiance, aucun mécanisme isolé ne peut garantir une sécurité complète face à l'évolution rapide des menaces.

Table des matières

Liste des figures

Introduction

1. Introduction

1.1 Contexte et motivations

La sécurité des systèmes informatiques a longtemps été abordée principalement comme une problématique logicielle, où les mécanismes de protection s'appuient essentiellement sur les couches supérieures (applications, systèmes d'exploitation). Cette approche, bien qu'ayant démontré son efficacité pour contrer certaines catégories de menaces, elle révèle aujourd'hui ses limites face à l'évolution rapide et continue des cyberattaques, qui ciblent désormais les couches les plus profondes des systèmes. L'ENISA (European Union Agency for Cybersecurity) a souligné dans ses rapports annuels sur les menaces du monde de la cybersécurité, cette évolution vers des attaques plus sophistiquées visant les couches basses des systèmes [ENISA 2023].

Longtemps ignorées ou sous-estimées, les firmwares, interfaces critiques situées entre le matériel et le logiciel, constitue un vecteur d'attaque privilégié par les acteurs malveillants. Un firmware compromis offre aux attaquants un contrôle quasi-complet du système, avec une capacité d'attaque où les mécanismes de protection logiciels et les tentatives de suppression standard sont inefficaces. Des cas emblématiques comme celui du rootkit UEFI (Unified Extensible Firmware Interface) *LoJax*, documenté par les chercheurs d'ESET en 2018, ont démontré la faisabilité d'implants malveillants persistants capables de survivre aux réinstallations complètes du système d'exploitation [ESET 2018]. Cette menace s'avère particulièrement critique pour les systèmes embarqués qui, soumis à des contraintes strictes en matière d'énergie et de ressources, se trouvent fréquemment dépourvus de protections contre ces attaques de bas niveau.

Face à ce changement notable du paysage des menaces informatiques, la sécurité matérielle émerge comme un impératif stratégique incontournable. L'approche dite de sécurité par conception (« Security by Design »), intégrant des mécanismes de protections matériels et logiciels, devenant essentielle pour établir une racine de confiance (« Root of Trust ») capable de garantir l'intégrité et la résilience globale du système. Le NIST (National Institute of Standards and Technology) a formalisé cette approche dans sa publication 800-193 « Platform Firmware Resiliency Guidelines » [NIST 2018]. Ce document fournit des recommandations techniques pour renforcer la résilience du firmware contre les attaques potentiellement destructrices. Dans le même esprit, L'Agence Nationale de Sécurité des Systèmes d'Information (ANSSI) a publié ses recommandations relatives à la sécurité matérielle sur plateformes x86 [ANSSI 2019]. Ce

guide présente des exigences de sécurité s'appliquant aux dispositifs matériels, préconisant notamment l'implémentation systématique d'un TPM (Trusted Platform Module) version 2.0, la configuration du BIOS/UEFI en mode Secure Boot, ainsi que le déploiement de mécanismes avancés de journalisation et d'audit du firmware.

L'adoption massive du TPM 2.0 (standardisé par la norme internationale ISO/IEC 11889:2015), désormais imposé par Microsoft comme matériel obligatoire pour installer son dernier système d'exploitation Windows, semble représenter une avancée majeure dans le domaine de la sécurité matérielle.

1.2 Problématique et objectifs

1.2.1 Problématique

La multiplication des attaques de bas niveau visant le firmware, qu'il s'agisse du Basic Input Output System (BIOS) historique ou, plus récemment, de l'UEFI, remet en cause l'hypothèse présupposé d'un matériel implicitement fiable. La vérification d'intégrité de ce code de bas niveau doit pouvoir s'appuyer sur un composant, créant ainsi une première vulnérabilité structurelle.

Cette problématique s'accentue lorsqu'on compare les systèmes généralistes aux architectures embarquées. Les premiers, dominés par l'écosystème x86/64, bénéficient de ressources matérielles conséquentes permettant l'intégration de mécanismes de protection (virtualisation matérielle, environnement d'exécution isolée). À l'inverse, les dispositifs embarqués (IoT, objets connectés) fondés principalement sur des architectures ARM ou RISC-V doivent concilier sécurité et contraintes strictes (consommation énergétique, mémoire limitée, et l'environnement où le système est utilisé). Ces différences imposent des stratégies de protection différentes, adaptées aux spécificités et aux limitations propres à chaque plateforme.

Le TPM 2.0 se présente comme un composant pivot pour ancrer la confiance, assurant une attestation de l'état système actuelle. Cependant, ses différentes formes, TPM discret (dTPM), TPM firmware (fTPM), TPM virtuel (vTPM), introduisent chacune des hypothèses de menace différentes : le dTPM, bien qu'isolé physiquement, expose une surface d'attaque matérielle via ses bus de communication (ex. interception de signaux sur puce), tandis que les implémentations logicielles (vTPM) soulèvent des questions fondamentales quant à leur isolation. Ces différentes implémentations et leurs implications sécuritaires seront analysées en détail dans la section 3.2.2.

1.2.2 Objectifs

Compte tenu de la complexité croissante des attaques bas niveau sur les firmwares et les architectures matérielles, ce mémoire se concentre sur une analyse approfondie des vulnérabilités associées et des mécanismes de défense qui en découlent.

Le premier objectif vise à dresser un panorama détaillé des vulnérabilités spécifiques aux firmwares et aux composants matériels des plateformes conventionnelles (x86/x64) et des systèmes embarqués (ARM, RISC-V). Cette démarche analytique permettra

d'identifier précisément les vecteurs d'attaque privilégiés selon les spécificités architecturales. Seront particulièrement étudiées les attaques ciblant l'intégrité du firmware (notamment via l'UEFI), les injections de fautes matérielles, les exploitations des interfaces critiques (DMA, JTAG, SPI) et les attaques par canaux auxiliaires (side-channel).

Le second objectif, central à notre analyse, concerne l'évaluation critique du TPM 2.0 en tant qu'élément fondamental de protection des systèmes modernes. Nous comparerons les différentes variantes d'implémentation du TPM (dTPM, fTPM, vTPM) en examinant leur efficacité face aux vecteurs d'attaques identifiés précédemment. Cette comparaison reposera sur une analyse détaillée des spécifications techniques publiées par le Trusted Computing Group et des vulnérabilités récentes documentées, notamment les failles cryptographiques, les attaques temporelles et les faiblesses d'implémentation spécifiques révélées par la communauté scientifique. Une attention particulière sera portée aux contraintes techniques et opérationnelles propres à chaque type de plateforme, permettant ainsi d'établir un cadre comparatif, visant à déterminer la pertinence des différentes implémentations du TPM.

Enfin, à titre exploratoire, ce mémoire propose une réflexion sur les architectures de sécurité matérielle. En s'appuyant sur les principes fondamentaux des racines de confiance matérielles et des enclaves sécurisées, nous examinerons les possibilités offertes par l'intégration synergique de ces technologies au sein d'architectures hybrides.

État de l’art des menaces matérielles et firmware

2. État de l’art des menaces matérielles et firmware

2.1 Attaques sur le firmware

Le firmware désigne un programme intégré directement dans un composant électronique (processeur, microcontrôleur, puce dédiée, périphérique) qui contrôle son fonctionnement fondamental et persiste généralement pendant toute la durée de vie du matériel. Contrairement aux logiciels traditionnels, il n'est pas destiné à être modifié fréquemment et fonctionne à l'interface directe entre matériel et logiciel. Cette section présente les principales menaces ciblant les couches basses des systèmes informatique, qui peuvent être classé en fonction des cibles qu’elles vissent.

Dans le contexte spécifique du démarrage système que nous analysons ici, le firmware UEFI/BIOS constitue la première séquence de code exécutée lors de l'initialisation d'un ordinateur.

2.1.1 Attaques sur l'UEFI/BIOS

L'UEFI et le BIOS constituent l'interface fondamentale entre le matériel informatique et le système d'exploitation. Bien que l'UEFI soit souvent présenté comme le successeur du BIOS, il est important de noter que l'UEFI moderne intègre généralement un mode de compatibilité (mode legacy) permettant d'émuler le fonctionnement d'un BIOS traditionnel pour assurer la rétrocompatibilité avec les systèmes d'exploitation plus anciens. Leurs rôles dans l’initialisation du matériel et le transfert du contrôle au noyau OS en fait des composants critique dans la chaîne de confiance du système.

Le BIOS (Basic Input/Output System), développé au début des années 1980, reposait sur une architecture limitée en mode réel 16 bits. Le mode réel 16 bits, dans lequel opérait initialement le BIOS, offrait un accès direct à la mémoire et aux périphériques. La limitation à 1 Mo d'espace adressable n'était pas intrinsèque au BIOS lui-même, mais résultait de l'architecture des premiers processeurs x86 et des contraintes de rétrocompatibilité maintenues au fil des évolutions. En réalité, même en environnement BIOS, le processeur pouvait basculer en mode ou en mode long (à partir des architectures

x86-64), permettant l'accès à davantage de mémoire et l'activation de mécanismes de protection. Cependant, la structure unifiée et rigide du BIOS traditionnel limitait effectivement sa modularité comparée à l'UEFI.

Le démarrage BIOS, commence par un Power-On Reset (POR) puis un POST (Power-On Self-Test) qui vérifie et initialise le processeur, la RAM et les périphériques (contrôleurs de clavier, affichage, …).

Le processus POST comprend plusieurs étapes techniques :

- Vérification : Test diagnostique des composants critiques (CPU, mémoire, contrôleurs), détection d'erreurs matérielles via des routines de test standardisées, et validation de l'intégrité des ressources système (sommes de contrôle).
- Initialisation : Configuration des registres CPU aux valeurs par défaut, établissement des tables d'interruption, configuration initiale des contrôleurs (chipset, DMA, PIC), et amorçage des sous-systèmes mémoire avec leurs paramètres fondamentaux.

Ce processus établit l'environnement de base nécessaire au chargement et à l'exécution des composants logiciels de plus haut niveau.

Le BIOS configure ensuite le matériel, construit sa table de périphériques et expose des services via des interruptions. Il recherche enfin le Master Boot Record sur le premier périphérique de démarrage configuré, charge ce secteur en mémoire et lui transfère l’exécution.

À partir de 2005, le standard UEFI (Unified Extensible Firmware Interface) a introduit une refonte complète du micrologiciel d’amorçage, structurée autour d’une architecture modulaire et avec l’ajout de mécanismes de sécurité :

- Un environnement d'exécution en mode protégé (32 bits) ou long (64 bits), permettant l'accès à toute la mémoire, ainsi que de la protection mémoire
- Une architecture modulaire basée sur des pilotes et applications indépendants
- Une partition système dédiée (ESP - EFI System Partition) pour stocker les chargeurs d’amorçage, garantissant une séparation entre firmware et système d’exploitation
- Prise en charge native de protocoles réseau
- Des mécanismes de sécurité comme le Secure Boot

Le processus de démarrage UEFI suit plusieurs phases séquentielles distinctes :

1. SEC (Security) : Phase initiale vérification de l'authenticité du firmware
2. PEI (Pre-EFI Initialization) : Initialisation minimale du matériel
3. DXE (Driver Execution Environment) : Chargement des pilotes principaux
4. BDS (Boot Device Selection) : Sélection du périphérique de démarrage
5. TSL (Transient System Load) : Chargement du système d'exploitation

Ces nouvelles caractéristiques architecturales, tout en apportant des améliorations fonctionnelles significatives, modifient indirectement la surface d'attaque. La Figure 1

illustre le flux du processus de démarrage et les points d'intervention potentiels pour les attaquants.

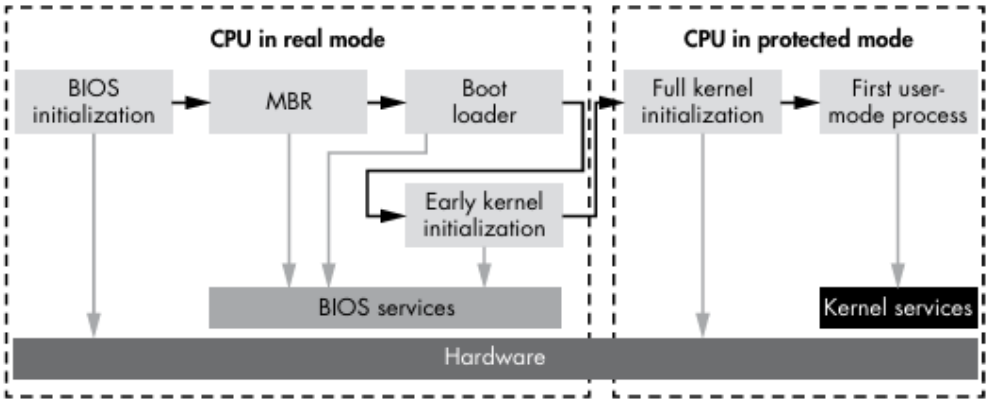

Figure 1 - Flux du processus de démarrage système et cibles potentielles d'attaques (source : Rootkits and Bootkits – p58)

Ainsi, cette modernisation a paradoxalement introduit de nouveaux vecteurs d'attaque, qui peuvent être catégorisé selon plusieurs approches.

La modification directe de la mémoire flash SPI (Serial Peripheral Interface) constitue l'approche la plus fondamentale pour compromettre le firmware. La puce flash SPI, généralement soudée directement sur la carte mère à proximité du chipset, stocke l'intégralité du code UEFI/BIOS et constitue donc une cible privilégiée. Cette puce SPI contient généralement 8-16 Mo de mémoire flash organisée en régions distinctes (descripteur, ME, BIOS) avec différents niveaux de protection.

L'attaquant doit [Xeno] :

1. Élever ses privilèges pour obtenir un accès kernel/ring-0
2. Désactiver le bit BIOSWE (BIOS Write Enable) dans le registre BIOS_CNTL
3. Neutraliser la protection BLE (BIOS Lock Enable)
4. Manipuler les registres Protected Range (PR0-PR4) pour autoriser l'écriture
5. Utiliser des opérations d'E/S directes pour écrire le code malveillant

Comme on peut le voir sur le chemin d’attaque, des protections sont implémentées par les fabricants, que l'attaquant doit franchir successivement [Xeno] :

- Le verrouillage logiciel du BIOS via son interface de configuration
- Les bits de protection du registre BIOS_CNTL, contrôlés par le chipset, qui empêchent les écritures non autorisées sur la puce flash
- Les mécanismes de protection en écriture du SPI lui-même, notamment le registre de statut FLOCKDN (Flash Configuration Lock-Down) qui, une fois activé, bloque toute modification des registres de configuration jusqu'au prochain redémarrage matériel
- La restriction d'accès aux plages d'adresses SPI via les registres Protected Range (PR0-PR4) qui définissent des régions en lecture seule

L'UEFI, contrairement au BIOS, adopte une conception modulaire où différents composants fonctionnels sont implémentés sous forme de pilotes et modules distincts. Cette architecture s'apparente à un mini-système d'exploitation temps réel, avec ses propres protocoles de communication inter-modules, ses services système et son modèle de pilotes extensible. Alors que le BIOS traditionnel se présente comme un unique bloc de code enchaînant toutes ses instructions de manière strictement linéaire, l'UEFI opère plus comme un environnement d'exécution complet, capable de charger dynamiquement des modules, d'exposer des interfaces de programmation (API) standardisées, et de maintenir un état cohérent entre ses différents composants.

Cette modularité, bien qu'avantageuse pour la maintenance et l'évolutivité, élargit la surface d'attaque en multipliant les points d'intervention potentiels. Les attaquants peuvent cibler des modules spécifiques, particulièrement ceux exécutés en phase DXE, qui offrent un contexte d'exécution privilégié et constituent un point d'entrée vers les couches inférieures du système.

Une troisième catégorie d'attaques cible spécifiquement plus largement la manipulation de l'ensemble des variables d'environnement de l'UEFI. Ces variables, stockées dans une mémoire non volatile (NVRAM) accessible au firmware, configurent divers aspects du comportement du système durant et après la séquence d'amorçage. Leur modification peut altérer fondamentalement la trajectoire d'exécution du système sans nécessiter la modification directe du code firmware, en modifiant par exemple les variables contrôlant l'ordre de démarrage ou en désactivant sélectivement des mécanismes de sécurité.

Le cas LoJax, documenté par ESET en 2018 [ESET 2018], représente la première documentation publique d'un rootkit UEFI déployé dans des opérations offensives réelles. Cette opération, attribuée au groupe APT28 (également connu sous les noms Fancy Bear ou Sednit), illustre parfaitement le commencement des attaques ciblant le firmware UEFI.

L'attaque exploitait une fonctionnalité présente dans l'UEFI : LoJack, un logiciel antivol préinstallé par de nombreux fabricants d'ordinateur portable.

Le processus d'infection se déroulait de cette manière :

1. Déploiement initial via des documents malveillants (typiquement des fichiers Word avec macros malveillantes) ciblant le système d'exploitation Windows de la victime, permettant l'installation un logiciel malveillant initial avec des privilèges utilisateur standard, servant de point d'entrée pour les étapes suivantes de l'attaque
2. Élévation des privilèges via l'exploitation de vulnérabilités système (comme CVE-2018-8120) pour obtenir des droits SYSTEM nécessaires aux opérations de bas niveau
3. Utilisation d'un pilote signé légitime mais détourné, nommé « RwDrv.sys » (issu de l'outil RWEverything), pour accéder directement aux registres matériels et à la mémoire SPI
4. Contournement des protections en écriture du firmware en manipulant les registres de contrôle du SPI, notamment en désactivant le bit BIOSWE (BIOS Write Enable) et en neutralisant la protection BIOS Lock Enable (BLE)

5. Installation d'un pilote UEFI malveillant directement dans la mémoire flash SPI, en ciblant spécifiquement la phase DXE (Driver Execution Environment) de l'UEFI pour garantir son chargement à chaque démarrage
6. Configuration d'un mécanisme garantissant le chargement d'un agent malveillant lors du démarrage du système d'exploitation via l'ajout d'entrées dans les variables UEFI persistantes

La persistance de cet implant s'explique par sa localisation dans la mémoire flash SPI physique, distincte et indépendante des supports de stockage de masse (disques durs, SSD) où réside le système d'exploitation. Contrairement aux malwares traditionnels stockés sur le disque système, un rootkit UEFI comme LoJax réside dans une puce dédiée sur la carte mère elle-même, expliquant pourquoi même le remplacement du disque dur ne permet pas d'éliminer l'infection. Seule une reprogrammation complète de la mémoire flash SPI (reflashage) peut éradiquer ce type d'implant.

2.1.2 Bootkits

Les bootkits représentent une classe de logiciels malveillants qui infectent les premières phases du processus de démarrage système, avant même que le système d'exploitation ne soit complètement chargé. Comme l'explique Matrosov, ces attaques ont connu une évolution significative passant des premiers virus de secteur d'amorçage (Boot Sector Infectors - BSI) aux attaques ciblant aujourd'hui les environnements UEFI. [Matrosov 2019]

Les origines de ces attaques remontent au développement des systèmes informatiques pré-IBM PC. Le programme *Creeper* (1971), considéré comme le premier logiciel autoréplicatif opérant en mode noyau, est souvent cité comme l'ancêtre des bootkits modernes. Les premières générations, telles que le PoC *eEye BootRoot,* présenté au Black Hat en 2005, se concentraient sur la compromission du Master Boot Record (MBR), exploitant sa position centrale dans l'architecture x86 pour persister hors de portée des mécanismes de détection du système d'exploitation.

Les bootkits ont connu une évolution parallèlement à l'architecture de démarrage des systèmes. Les bootkits MBR, première génération, modifient les 512 premiers octets du disque et détournent le flux de contrôle vers un code malveillant stocké dans des secteurs cachés. Les bootkits VBR (Volume Boot Record) constituent la deuxième génération, particulièrement efficaces contre les systèmes multi-volumes en ciblant le secteur d'amorçage de chaque partition. Avec l'avènement de l'UEFI, les bootkits comme ESPecter manipulent désormais la partition système EFI (ESP) en remplaçant ou en interceptant le bootloader légitime. La dernière génération, représentée par les bootkits anti-Secure Boot, exploite des vulnérabilités spécifiques comme CVE-2022-21894 pour contourner les vérifications cryptographiques de signature, fondement même de la protection Secure Boot.

Ces stratégies d'attaques peuvent être classifiées en quatre groupes distincts :

- Instrumentalisation des mécanismes légitimes : utilisation des fonctionnalités intégrées au système d'exploitation (ex. désactivation temporaire de la

vérification de signature via des commandes spécifiques), souvent en exploitant des vulnérabilités dans les outils de diagnostic ou de test.

- Exploitation de failles système : ciblage de vulnérabilités critiques dans le noyau ou dans des pilotes signés, permettant l'exécution arbitraire de code non authentifié.
- Attaques sur le chargeur d'amorçage : modification du bootloader pour modifier le noyau et désactiver les vérifications de sécurité avant leur initialisation.
- Infection du firmware

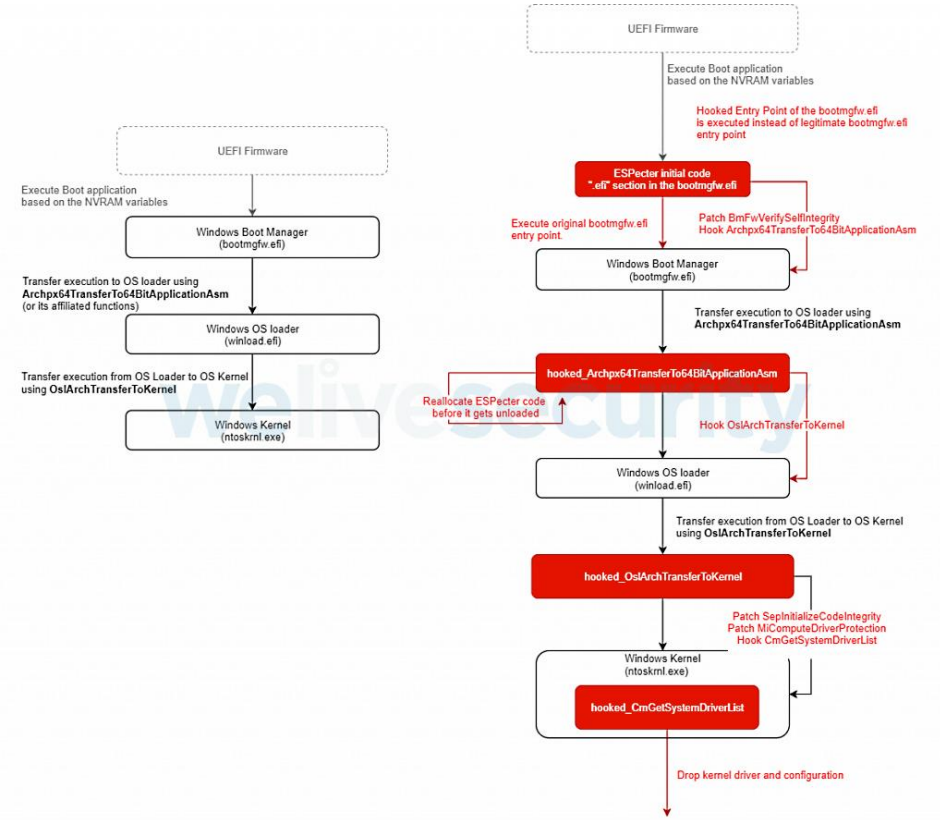

Figure 2 - Démarrage UEFI standard de Windows vs séquence de démarrage modifiée par ESPecter (https://www.bleepingcomputer.com/news/security/new-uefi-bootkit-used-to-backdoor-windows-devices-since-2012/)

En 2023, le bootkit BlackLotus est apparu comme le premier bootkit UEFI capable de contourner la protection Secure Boot sur des systèmes Windows entièrement à jour, exploitant une vulnérabilité connue (CVE-2022-21894) pour s'installer. BlackLotus fonctionne en exploitant une vulnérabilité dans le processus de démarrage Windows pour charger des fichiers DLLs et EXEs non signés malgré Secure Boot. Il utilise une technique de "bootkit remapping" qui intercepte les appels systèmes au niveau du bootloader avant que les mécanismes de protection de l'OS ne soient actifs.

Plus récemment, en 2024, la découverte de Bootkitty, le premier bootkit UEFI ciblant spécifiquement les systèmes Linux, marquant ainsi l'extension de cette menace au-delà de l'écosystème Windows.

2.2 Attaques sur les composants matériels

Les composants matériels des systèmes informatiques constituent un vecteur d'attaque fondamental pour les acteurs malveillants cherchant à compromettre la sécurité des systèmes. Contrairement aux vulnérabilités purement logicielles qui peuvent être corrigées par des mises à jour de firmware ou de système d'exploitation, les failles fondamentales dans l'architecture matérielle elle-même (comme les défauts de conception des circuits intégrés) demeurent généralement exploitables pendant toute la durée de vie du composant. Ces vulnérabilités, telles que Spectre, Meltdown ou Rowhammer, ne peuvent être véritablement éliminées que par une refonte du silicium et un remplacement physique des composants affectés. Les correctifs logiciels déployés pour ces problèmes architecturaux offrent généralement des atténuations qui réduisent l'exploitabilité mais entraînent souvent des compromis significatifs en termes de performance et ne suppriment pas la vulnérabilité.

2.2.1 Attaque sur la mémoire

Les attaques ciblant les sous-systèmes mémoire exploitent les caractéristiques physiques et architecturales des différents types de mémoire pour compromettre la confidentialité, l'intégrité ou la disponibilité des données. Ces attaques peuvent être classifiées en deux catégories principales : les attaques par perturbation physique et les attaques spéculatives.

L'attaque Rowhammer, documentée par [Kim 2014], constitue l'exemple emblématique des attaques par perturbation physique. Cette technique exploite une vulnérabilité fondamentale des mémoires DRAM (Dynamic Random Access Memory). L'attaquant identifie d'abord des paires de lignes mémoire physiquement adjacentes, appelées "aggressor rows". Il accède ensuite répétitivement et alternativement à ces lignes, souvent plus de 100 000 fois par seconde, créant des perturbations électriques par interférence. Ces perturbations déchargent prématurément les condensateurs des cellules de la ligne victime située entre les deux lignes agresseurs. Les bits basculent lorsque la charge des condensateurs tombe sous le seuil de détection, modifiant ainsi les données stockées, tout cela sans avoir accès à ces cellules. L'impact peut être dévastateur : modification des tables de pages mémoire, corruption des structures de contrôle du noyau ou altération des bits de privilège dans les descripteurs de sécurité.

Face à cette menace, les fabricants de puces mémoire ont introduit des mécanismes d'atténuation, notamment le Target Row Refresh (TRR), conçu pour détecter et prévenir les modèles d'accès caractéristiques d'une attaque Rowhammer. Cependant, l'évolution de ces contre-mesures a été suivie par le développement de techniques d'attaque plus sophistiquées, comme TRRespass, Frigo a démontré la capacité à contourner ces protections en employant des modèles d'accès plus complexes et distribués, établissant que les implémentations de TRR étaient vulnérables à des « many-sided hammering patterns » (schémas d'accès multidirectionnels qui ciblent simultanément plusieurs lignes de mémoire adjacentes [Frigo 2020]. Jattke a affiné cette approche en développant un algorithme capable de découvrir automatiquement des modèles d'accès mémoire

efficaces pour déclencher des bit flips, contournant ainsi pratiquement toutes les implémentations TRR existantes [Jattke 2022].

Les attaques spéculatives exploitent les optimisations architecturales des processeurs modernes pour extraire des informations sensibles.

Meltdown et Spectre ont révélé en 2018 et 2019 comment l'exécution spéculative et le réordonnancement des instructions pouvaient être exploités pour contourner les frontières de sécurité et accéder à des données privilégiées. Ces attaques exploitent les mécanismes d'optimisation des processeurs en ciblant l'exécution spéculative, c'est quand le processeur exécute par anticipation des instructions qui pourraient être nécessaires (par ex : réordonnancement des instructions). Meltdown exploite spécifiquement le fait que la vérification des privilèges est effectuée après l'exécution spéculative des instructions. Cette fenêtre temporelle, bien que minuscule (quelques nanosecondes), suffit pour créer des effets secondaires mesurables sur le cache, notamment des variations de temps d'accès. Ces variations permettent l'extraction d'informations sensibles à travers un canal auxiliaire basé sur les timings d'accès au cache, contournant ainsi les protections d'isolation mémoire les plus fondamentales du système. [Meltdown 2018] [Spectre 2019]

En 2024, l'attaque micro-architecturale Indirector a affecté les processeurs Intel de 13e et 14e générations, révélant comment des entiers spéculatifs pouvaient être exploités pour extraire des secrets à travers le cache [Bitdefender 2024].

2.2.2 Attaques par injection de fautes

L'injection de fautes consiste à perturber, de manière contrôlée, la tension, l'horloge ou l'environnement physique du composant pour forcer des dérives de calcul.

La manipulation de la tension d'alimentation représente une première approche, où l'introduction de variations rapides ou de « glitches » peut perturber le fonctionnement normal des circuits, induisant des erreurs de calcul ce qui peut aboutir à des contournements de vérification de sécurité. Cette technique exploite la sensibilité des semi-conducteurs aux fluctuations de tension, particulièrement durant les opérations critiques comme l'exécution d'algorithmes cryptographiques ou de vérifications d'authentification.

Un exemple particulièrement significatif est la technique VGlitch documentée par Jerinsunny [Jerinsunny 2024], démontrant la vulnérabilité des microcontrôleurs STM32 aux attaques par tension. Cette recherche a révélé qu'en appliquant des impulsions de tension précis et synchronisées, il était possible de contourner les mécanismes de protection de la mémoire (PMP) et d'exécuter du code non autorisé, compromettant ainsi l'intégrité du système. Ces perturbations provoquent des erreurs dans l'exécution des instructions, comme la transformation d'une instruction de branchement conditionnel (BEQ - Branch if EQual, qui n'exécute le saut que si la condition d'égalité est remplie) en branchement inconditionnel (comme JMP - Jump, qui exécute toujours le saut sans vérifier aucune condition), ce qui permet de sauter des vérifications cruciales. Pour réaliser cette attaque, un générateur d'impulsions programmable et une sonde de tension

haute précision sont utilisés pour injecter des transitoires de tension de quelques nanosecondes à des moments précis du cycle d'horloge. La précision temporelle et l'amplitude de ces glitches sont calibrées pour affecter spécifiquement certaines portes logiques sans déclencher les détecteurs de sous-tension ou provoquer une réinitialisation complète du système.

Les perturbations électromagnétiques constituent une seconde approche, où l'application ciblée de champs électromagnétiques localisés peut induire des courants parasites dans les circuits, perturbant ainsi leur fonctionnement normal. Cette méthode présente l'avantage significatif de pouvoir être implémentée sans contact physique direct avec le composant ciblé, augmentant ainsi sa discrétion et réduisant les traces forensiques. La technique EMFI (Electromagnetic Fault Injection) a évolué en conséquence, avec le développement d'injecteurs de fautes électromagnétiques de haute précision capables de cibler des zones spécifiques d'un circuit intégré. Les courants induits peuvent modifier l'état des transistors pendant quelques nanosecondes, suffisamment pour transformer un « 0 » en « 1 » dans un registre critique ou inverser une condition d'authentification. Une étude a démontré l'efficacité de l'EMFI contre des implémentations matérielles d'algorithmes cryptographiques : Dehbaoui a montré qu'une impulsion EMFI pouvait provoquer des fautes exploitables par analyse différentielle pour extraire la clé d'un AES. [Dehbaoui 2012]

Les attaques par injection de fautes peuvent avoir différents effets, tel que le « saut instruction », c'est-à-dire qu'on va forcer le processeur à sauter des instructions qu'il devait normalement exécuter, ce qui provoquera une possible corruption des données ou alors de changer le flux d'exécution pour le détourner vers un code malveillant.

2.2.3 Attaques sur les périphérique et interfaces

Les périphériques et interfaces matérielles constituent des vecteurs d'attaque dans l'architecture système, exploitant les privilèges élevés accordés à certaines interfaces permettant de contourner les mécanismes de protection du système d'exploitation car opérant souvent à un niveau de privilège supérieur aux défenses logicielles.

Dans les systèmes x86, l'interface SPI est utilisée pour stocker le firmware UEFI/BIOS. Comme nous l'avons vu précédemment, celles-ci peuvent être utilisé pour l'installation de bootkits persistants et ainsi compromettre le système au plus bas niveau.

Les attaques DMA (Direct Memory Access), exploitant les interfaces offrant un accès direct à la mémoire physique (comme Thunderbolt, PCIe ou FireWire). Ces interfaces sont conçues pour optimiser les performances en permettant aux périphériques de communiquer directement avec la mémoire système sans intervention du processeur. Un périphérique peut théoriquement utiliser ces capacités DMA pour lire ou écrire arbitrairement dans la mémoire système, contournant ainsi les mécanismes de protection implémentés au niveau du système d'exploitation.

Des attaques comme Thunderspy ont démontré comment les interfaces Thunderbolt pouvaient être exploitées pour contourner complètement les protections du système d'exploitation et accéder aux données chiffrées, même sur un système verrouillé ou en

veille. Comme il est conclu dans l'article, « par un accès physique de seulement cinq minutes au dispositif, un attaquant peut extraire les données de périphériques Windows ou Linux équipés de ports Thunderbolt » [Thunderbolt 2020]. L'interface Thunderbolt offre un accès DMA avec une bande passante de 40 Gbps, permettant théoriquement aux périphériques connectés de lire et d'écrire directement dans la mémoire système. Bien que les contrôleurs IOMMU (Input–output memory management unit) (ex : Intel VT-d/AMD-Vi) soient censés restreindre ces accès DMA, l'attaque Thunderspy contourne ces protections en reprogrammant le firmware du contrôleur Thunderbolt. Cette reprogrammation s'effectue par extraction et modification du firmware, désactivant les restrictions de sécurité au niveau matériel. L'attaque exploite également les fenêtres de vulnérabilité qui apparaissent pendant la phase d'initialisation du système, avant que toutes les protections ne soient actives. Une fois ces barrières contournées, l'attaquant peut lire et écrire directement dans la mémoire système, contournant toutes les protections logicielles du système d'exploitation, y compris le chiffrement de disque, puisque les clés déchiffrées résident en mémoire pendant l'utilisation.

L'exploitation des microcontrôleurs périphériques constitue un second vecteur. Les périphériques (cartes graphiques, carte réseau, disques SSD, …) intègrent leurs propres processeurs exécutant un firmware dédié, opérant très souvent avec des privilèges élevés et un accès direct aux ressources système. Les contrôleurs BMC (Baseboard Management Controller) présents dans les serveurs d'entreprise illustrent particulièrement cette menace. Le BMC est un microcontrôleur spécialisé intégré à la carte mère des serveurs, qui fonctionne indépendamment du système d'exploitation principal et du processeur hôte, il permet aux administrateurs de gérer à distance l'ensemble des fonctions du serveur. En 2023, Eclypsium a révélé deux vulnérabilités critiques (CVE-2023-34329 et CVE-2023-34330) dans le firmware BMC MegaRAC, utilisé par de nombreux fournisseurs (HP, Dell, …), ces vulnérabilités permettaient l'exécution de code arbitraire avec des privilèges root matériel, sans authentification préalable. [Eclypsium 2023]

Les attaques ciblant les bus de communication système, comme I2C, SPI ou JTAG, constituent un autre vecteur. Insuffisamment protégées lors de la production matérielle, ces interfaces, initialement conçues pour le débogage, la configuration ou la programmation des composants matériels, peuvent être exploitées pour accéder à des fonctionnalités privilégiées. Accessibles via des points de test sur le circuit imprimé, elles offrent un accès direct à la mémoire et aux registres processeur permettant :

- La lecture et l'écriture du firmware en clair
- Le contournement de toute authentification logicielle
- L'injection de fautes ciblées (*glitching*) pour sauter une étape de vérification.

Après avoir abordé les attaques ciblant directement les composants matériels, il est essentiel d'examiner une autre catégorie d'attaques qui, bien que ne nécessitant pas un accès physique direct au matériel, exploitent néanmoins les caractéristiques physiques ou temporelles des composants pour compromettre leur sécurité : les attaques par canaux auxiliaires.

2.3 Attaques par canaux auxiliaires

Les attaques par canaux auxiliaires exploitent des fuites physiques ou temporelles émises par un dispositif lors de ses opérations cryptographiques afin de récupérer des clés ou des données sensibles. Plutôt que de s'appuyer sur une vulnérabilité logique du firmware ou du système d'exploitation, ces attaques mesurent et analysent des caractéristiques telles que le temps d'exécution, la consommation électrique ou encore les émissions électromagnétiques. Cette menace, peut être menées à distance (via des sondes) ou localement.

Les attaques par canaux auxiliaires peuvent être classifiées selon plusieurs dimensions : le canal exploité, la méthode d'analyse, et la proximité requise avec le dispositif cible.

L'attaque basées sur la consommation d'énergie est basée sur deux principales techniques

1. SPA (Simple Power Analysis) : cette technique examine directement les variations de consommation électrique pendant l'exécution d'opérations cryptographiques pour identifier des motifs correspondant à des opérations spécifiques. Par exemple, dans une implémentation naïve de RSA, la différence de consommation entre une opération de multiplication et une opération de carré peut révéler directement les bits de la clé privée. La SPA nécessite généralement peu d'échantillons mais requiert une connaissance approfondie de l'algorithme ciblé.
2. DPA (Differential Power Analysis) représente une approche plus sophistiquée que la SPA. Cette technique exploite des méthodes statistiques appliquées à de multiples traces de consommation électrique pour extraire les informations relatives aux clés cryptographiques. Sa robustesse lui permet de rester efficace même en présence d'un bruit de mesure significatif ou lorsque des contre-mesures élémentaires ont été implémentées dans le dispositif ciblé.

Au-delà de la consommation électrique, les circuits intégrés émettent des ondes électromagnétiques proportionnelles à l'activité des transistors. En plaçant une antenne ou une sonde près du composant, l'attaquant enregistre ces émissions et, via des traitements spectrogrammes, identifie des patterns corrélés aux opérations cryptographiques. Les attaques SEMA (Simple Electromagnetic Analysis) et DEMA (Differential Electromagnetic Analysis) suivent des principes similaires à leurs homologues basées sur la consommation d'énergie, mais présentent un avantage significatif, elles peuvent être réalisées à distance, sans contact direct avec le circuit.

Les attaques temporelles consistent à mesurer la durée d'exécution d'opérations cryptographiques et à en déduire des informations sur la clé secrète. Chaque instruction ou branche conditionnelle peut présenter un temps d'exécution variable selon les bits traités, en accumulant suffisamment de mesures, un attaquant peut reconstituer l'ensemble de la clé [Kocher 1996]. Ces attaques restent d'actualité, notamment contre les implémentations TLS sur serveurs cloud, avec le "cache timing" [Bitdefender 2024]. D'autres variantes comme les attaques par cache (Cache-timing, Flush+Reload, Prime+Probe) exploitent le partage des caches entre processus pour inférer des informations sensibles à partir des schémas d'accès mémoire.

D'autres canaux physiques peuvent également être exploités :

- Attaques acoustiques : Analyser le son émis par un clavier pour déterminer les touches pressées, ou les variations sonores subtiles de certains composants électroniques pendant des opérations cryptographiques.
- Analyse thermique : Exploiter les variations de température des composants pour déduire des informations sur les calculs effectués.

2.4 Menaces systèmes embarqués et IOT

Les systèmes embarqués ainsi que les objets de l'internet des objets (IoT), qu'ils motorisent un véhicule, un dispositif médical ou un capteur industriel, partagent trois contraintes importantes : des ressources matérielles limitées (CPU, mémoire, énergie), une connectivité quasi permanente et des cycles de vie variables : relativement courts (3-5 ans) pour les dispositifs IoT grand public, mais potentiellement très longs (10-15 ans) pour certains systèmes embarqués industriels ou critiques. Dans les deux cas, ces cycles de vie sont souvent caractérisés par des mises à jour irrégulières qui créent des fenêtres de vulnérabilité prolongées. Avec une projection de plus de 75 milliards d'objets connectes selon le NIST, ces objets deviennent un acteur majeur au niveau de la sécurisation : 67,3 % des firmwares IoT exploitent aujourd'hui encore des bibliothèques obsolètes, exposant ainsi de nombreuses applications à des vulnérabilités connues [AutoFirm 2024]

2.4.1 Chaîne d'approvisionnement : du silicium au firmware

La production d'un objet connecté mobilise souvent un écosystème mondial de sous-traitants, rendant particulièrement complexe la vérification de l'authenticité et de l'intégrité des composants (SoC, microcontrôleurs, modules radio). Les rapports d'ENISA signalent une augmentation préoccupante de la contrefaçon matérielle et de l'insertion de portes dérobées dès l'usine sous forme de blocs IP malveillants [ENISA 2023]. Avec une possibilité d'implants matériels (« hardware implants ») dû un contexte géopolitique et d'espionnage, ces dispositifs microscopiques, détectables uniquement par radiographie ou microscope électronique, peuvent être insérés lors de la fabrication et de la distribution.

Comme évoqué § 2.2.3, les ports JTAG demeurent fréquemment actifs sur le produit final, faute de locking définitif ou d'e-fuses dû à des contrainte de réduction des coûts. Ils deviennent donc la première cible lors d'une attaque hardware, avec le faible coût des sondes et des injecteurs de glitch, cela rend cette menace accessible à un très large éventail d'attaquants. Une étude en 2018 par Vishwakarma recense de nombreux cas d'exploitation réussie de JTAG sur des dispositifs IoT grand public, notamment pour extraire des clés cryptographiques ou injecter du code malveillant [Vishwakarma 2018].

2.4.2 Vulnérabilités liées au cycle de vie et à la maintenance

La gestion des mises à jour constitue un autre défi sécuritaire majeur pour ces systèmes. Si de nombreux dispositifs IoT intègrent des mécanismes de mise à jour over-the-air

(OTA), leur implémentation souffre souvent de failles critiques. L'absence de vérification cryptographique robuste expose ces systèmes à des attaques de type « firmware poisoning », où un attaquant peut injecter un firmware malveillant via un proxy ou un serveur de mise à jour compromis.

Cette vulnérabilité est amplifiée par l'abandon des constructeurs, à partir d'un cycle, d'un support de ses appareils, ainsi les vulnérabilités découvertes demeurent sans correctif. Cette problématique est particulièrement prononcée dans les contextes industriels, où des systèmes embarqués critiques qui peuvent rester opérationnels pendant des décennies, accumulant progressivement une dette de sécurité considérable.

2.4.3 Failles protocolaires, configuration et attaques sur les ressources

Outre les vulnérabilités matérielles et de firmware, les protocoles de communication IoT (MQTT, CoAP, HTTP/REST, WebSocket) et les différentes APIs constituent un vecteur d'attaque privilégié. Par exemple, le protocole MQTT, largement déployé pour sa légèreté, a révélé 33 vulnérabilités critiques affectant des millions de dispositifs, parmi lesquelles 18 jugées « critical » par Kaspersky [Mitchell 2022].

Dans le contexte des dispositifs IoT grand public, la fragilité sécuritaire se manifeste également par l'utilisation de configurations par défaut. De nombreux produits sont déployés avec des identifiants d'accès inchangé, des interfaces de gestion insuffisamment protégées, ou des services superflus activés par défaut. Les botnets Mirai et ses dérivés en sont des exemples frappants, ayant exploité ces vulnérabilités pour orchestrer des attaques distribuées massives.

Les dispositifs modernes implémentent souvent plusieurs protocoles simultanément (WiFi, Bluetooth, LoRa, ZigBee), chacun présentant son propre modèle de menace. Cette multiplicité protocolaire crée un environnement propice aux attaques transitives, où la compromission d'un sous-système peut compromettre l'ensemble du dispositif.

Les attaques par épuisement de batterie constituent une menace particulière aux systèmes alimentés par pile. Un attaquant peut forcer des transmissions radio répétées ou des calculs intensifs pour vider prématurément la batterie, une forme de déni de service particulièrement efficace contre les capteurs déployés dans des zones difficiles d'accès.

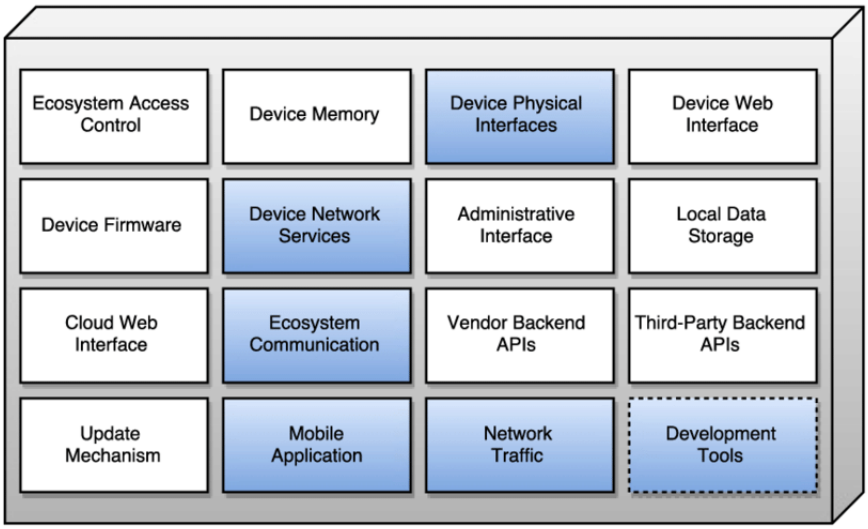

Figure 3 - Surface d'attaque IOT (source : https://www.researchgate.net/figure/oT-Attack-Surface-Areas-Based-on-Miessler-2015_fig2_286440570)

Face à l'ensemble de menaces qui ont été abordés, les approches traditionnelles de sécurisation s'avèrent souvent inadaptées, nécessitant l'élaboration de défense spécifiquement conçus pour les contraintes et vulnérabilités uniques des systèmes embarqués, IoT et x86. Ces architectures de protection, qui seront explorées dans le chapitre suivant, doivent intégrer simultanément les contraintes matérielles inhérentes à ces systèmes tout en établissant des fondations sécuritaires robustes adaptées à leur déploiement dans des environnements potentiellement hostiles.

Architectures de protection matérielle

3. Architectures de protection matérielle

3.1 Principes fondamentaux de défense

Dans un contexte où les attaquent ciblant les couches basses (firmware, pilotes, chargeurs d'amorçage) deviennent de plus en plus sophistiquées, la conception d'architectures de protection matérielle s'impose comme une exigence critique. Ces architectures reposent sur quatre piliers fondamentaux :

- Les racines de confiance
- Les chaînes de confiance
- Les mécanismes d'attestation
- Les principes d'isolation et de cloisonnement

Leur combinaison permet d'établir un continuum de sécurité depuis l'amorçage du système jusqu'aux applications de haut niveau.

- Racine de confiance (Root of Trust) : Selon la norme ISO/IEC 11889:2015, une racine de confiance désigne « un ensemble de fonctions au sein d'un système de confiance qui sont toujours implicitement fiables et qui forment la base permettant d'établir la confiance dans l'ensemble du système ». Le NIST SP 800-193 précise qu'une RoT doit posséder trois propriétés essentielles : elle doit être « inaltérable » (immutable), « mesurable de manière fiable » (reliably measured) et « minimale » pour réduire la surface d'attaque.
- Chaîne de confiance (Chain of Trust) : Définie par le standard GlobalPlatform TEE System Architecture v1.3 comme une séquence de transferts d'exécution où chaque étape vérifie cryptographiquement l'intégrité et l'authenticité de l'étape suivante avant de lui transférer le contrôle, créant ainsi une propagation transitive de la confiance depuis la racine initiale.
- Mécanismes d'attestation : Ils complètent la chaîne de confiance en fournissant, à un tiers ou à un hyperviseur, la preuve cryptographique de l'état exact de la plateforme. En utilisant des valeurs aléatoires uniques (nonces) comme défis cryptographiques et des signatures générées par des clés d'attestation dédiées, ils garantissent que le système n'a pas été compromis depuis sa mesure initiale.

- Isolation/Cloisonnement : Le NIST IR 8320 caractérise l'isolation comme « la séparation des domaines d'exécution ou de stockage pour prévenir toute influence non autorisée entre eux », en distinguant l'isolation spatiale (séparation des ressources mémoire), temporelle (séparation des cycles d'exécution) et logique (séparation des privilèges d'accès).

3.1.1 Racines de confiance matérielles

La racine de confiance matérielle (Hardware Root of Trust (HRoT)) constitue le fondement de toute architecture de sécurité moderne. Selon les recommandations du NIST [NIST 2018], comme dit précédemment, toute RoT doit répondre à au moins trois critères essentiels : immutabilité, capacité cryptographique, et résistance aux attaques physiques et logiques.

L'immutabilité s'exprime par la non-modifiabilité des données sensibles après la fabrication, souvent assurée par des mémoires ROM (Read Only Memory). La capacité cryptographique inclut la génération de clés asymétriques (RSA, ECC) et le calcul de fonctions de hachage sécurisées (SHA-256 au moins), tandis que la résistance aux attaques repose sur la protection des attaques physiques ou des contre-mesures contre les attaques par canaux auxiliaires.

La littérature distingue plusieurs services de RoT, notamment :

- Root of Trust for Measurement (RTM) : initialise le processus de mesure du code
- Root of Trust for Verification (RTV) : vérifie la validité des blobs mesurés.
- Root of Trust for Storage (RTS) : fournit un stockage sécurisé pour les données sensibles.

Ces services collaborent en chaîne pour établir une fondation de confiance dès l'allumage du système.

3.1.2 Chaînes de confiance et attestation

Le concept de chaîne de confiance constitue un fondement architectural critique dans la sécurité moderne des systèmes informatiques. Il repose sur un principe fondamental : établir une séquence ininterrompue de validations cryptographiques depuis un ancrage de confiance initial jusqu'aux couches applicatives.

Cette approche s'articule autour de trois mécanismes, la mesure séquentielle, la validation cryptographique et le transfert du contrôle après que les précédents mécanismes ont réussi. C'est la procédure qui est utilisé dans Secure Boot et qui sera discuté plus en détail dans une prochaine partie.
Dans la mesure séquentielle, chaque composant de la chaîne produit un hash du composant suivant avant de lui transférer le contrôle d'exécution, stocké dans des registres protégés contre les modifications.

Deux modèles principaux d'établissement de chaîne de confiance coexistent : La Static Root of Trust for Measurement (SRTM) initialise la chaîne dès l'allumage du système et maintient une séquence continue de validations tout au long du processus de démarrage

et le Dynamic Root of Trust for Measurement (DRTM) qui permet d'établir une nouvelle racine de confiance à tout moment pendant l'exécution du système, utile dans les environnements dynamiques.

L'attestation matérielle permet ensuite de fournir à un tiers (hyperviseur, gestionnaire de réseau) des preuves cryptographiques de l'état du système, avec l'utilisation d'une clé d'attestation (Attestation Identity Key), consolidant la confiance tout au long du cycle

3.1.3 Isolation et cloisonnement

Le principe d'isolation vise à compartimenter les ressources du système afin de contenir les éventuelles compromissions et limiter leur propagation

L'isolation peut être mise en œuvre à différents niveaux :

- Isolation physique : séparation matérielle complète des composants critiques (ex : Secure Element).
- Isolation par virtualisation : utilisation d'hyperviseurs pour séparer les environnements d'exécution.
- Isolation par contrôle d'accès mémoire : restriction des accès à certaines zones mémoire par des mécanismes matériels.
- Isolation temporelle : séparation dans le temps des opérations critiques et non critiques.

Les architectures ARM TrustZone illustrent cette isolation via deux mondes distincts, « Secure World » et « Normal World », avec une barrière matérielle empêchant tout accès non autorisé (sans une authentification préalable) du monde normal au monde sécurisé

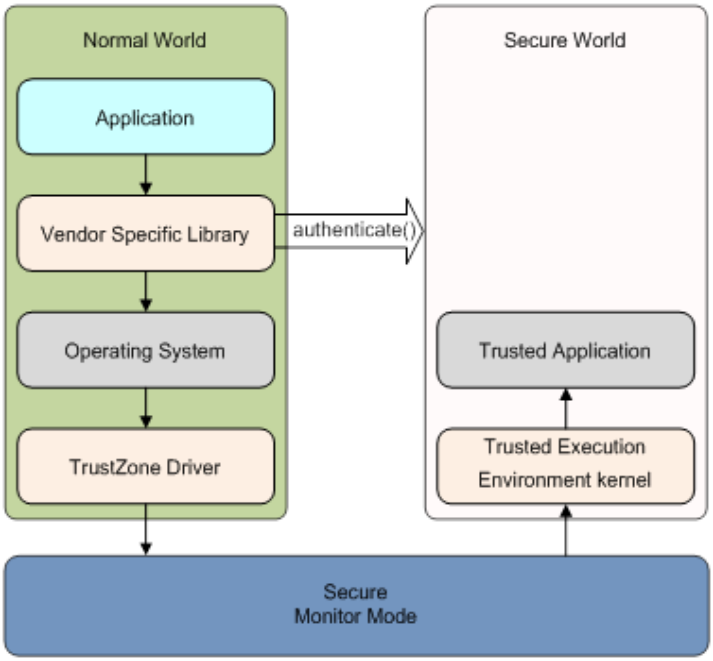

Figure 4 - La vision de ARM sur l'isolation et le cloisonnement (source : ARM)

Le cloisonnement des privilèges s'appuie sur le principe de moindre privilège (Principle of Least Privilege), selon lequel chaque composant ne doit disposer que des droits strictement nécessaires à son fonctionnement, réduisant ainsi la surface d'attaque.

3.2 Technologies de sécurité matérielle pour systèmes x86/x64

Les architectures de protection matérielle pour plateformes x86/x64 ont considérablement évolué ces dernières années. Cette section analyse les mécanismes fondamentaux déployés dans les systèmes modernes pour garantir un niveau de sécurité adéquat jusqu'au démarrage du système d'exploitation.

3.2.1 Secure Boot et UEFI protégé

Le Secure Boot constitue un mécanisme de protection essentiel établissant une chaîne de confiance cryptographique durant la séquence d'amorçage. Son implémentation repose sur une validation systématique des signatures de chaque composant depuis le firmware initial jusqu'au noyau du système d'exploitation.

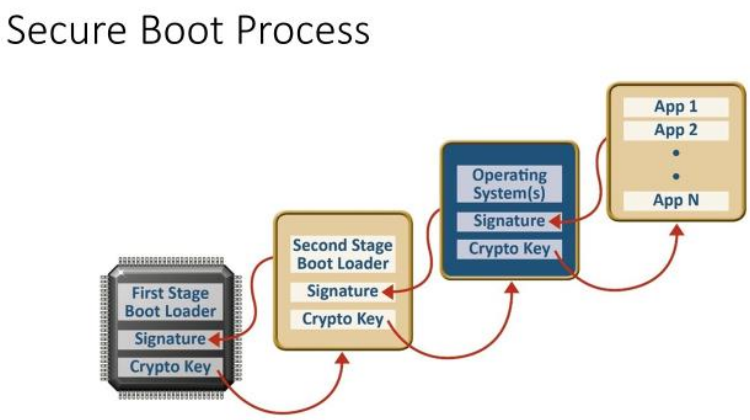

Figure 4 - Processus du Secure Boot (source : https://ealtili.medium.com/secure-boot-process-8b5fa87903f4)

L'architecture de validation s'articule autour d'une hiérarchie de clés :

- Platform Key (PK) : Racine de confiance contrôlant l'accès aux variables UEFI protégées
- Key Exchange Keys (KEK) : Clés intermédiaires permettant la signature des certificats d'autorisation
- Bases de données db/dbx : Contenant respectivement les signatures autorisées et révoquées

Cette hiérarchie de confiance implique plusieurs acteurs exerçant des responsabilités, les fabricants de matériel (OEMs), établissent initialement la Platform Key (PK) lors de la fabrication du système et l'inscrivent dans la mémoire non volatile. Cette clé représente l'autorité ultime sur la configuration de sécurité UEFI. Les OEMs préconfigurent également les KEKs initiales et les entrées des bases db/dbx. Les OS peuvent aussi fournisent leurs KEKs

Les processeurs modernes intègrent des racines de confiance matérielles complémentaires au Secure Boot UEFI. Intel Boot Guard, introduit avec les processeurs de 4ème génération, implémente une racine de confiance matérielle vérifiant l'authenticité du premier code exécuté lors du démarrage avec deux modes opérationnels distincts : le mode Verified Boot, qui vérifie l'authenticité du firmware sans bloquer nécessairement l'exécution en cas d'échec, et le mode Measured Boot, qui calcule une empreinte cryptographique du firmware et la stocke dans le TPM pour permettre une attestation ultérieure. La clé publique de vérification est stockée dans des fusibles électroniques non reprogrammables, établissant un ancrage de confiance résistant à la subversion.

Parallèlement, AMD a développé sa propre solution avec le Platform Security Processor (PSP), un coprocesseur de sécurité intégré qui vérifie l'intégrité du BIOS avant son exécution et implémente une racine de confiance matérielle pour le système. Son architecture repose sur un cœur ARM dédié, isolé du processeur principal, garantissant ainsi une séparation physique entre l'environnement d'exécution sécurisé et le reste du système. Le PSP implémente également des capacités de chiffrement autonomes et prend en charge l'implémentation du firmware TPM (fTPM).

Ces technologies constituent une défense significative contre les attaques ciblant la phase du démarrage, comme le souligne l'ANSSI dans ses recommandations pour les plateformes x86 [ANSSI 2019]. Cependant, ces protections ont montré leurs limites avec la découverte de vulnérabilités critiques. La faille CVE-2024-7344, identifiée par les chercheurs d'ESET en janvier 2025, permet le contournement du Secure Boot. [ESET 2024]. Cette vulnérabilité critique exploite une faiblesse dans le processus de validation des signatures du bootloader, permettant à un attaquant disposant de privilèges administratifs d'injecter du code non signé dans la chaîne de démarrage.

Dans l'architecture ARM, il existe ce qu’on appelle Trusted Firmware (ATF), ce processus, similaire au fonctionnement de Secure Boot implique plusieurs étapes : BL1 (ROM) → BL2 (Trusted Boot Firmware) → BL3 (Runtime Firmware) → OS

3.2.2 Trusted Platform Module : variantes et vulnérabilités

Le TPM représente une composante fondamentale dans l'architecture de sécurité modernes. Spécifié par le Trusted Computing Group (TCG), le TPM 2.0 constitue aujourd'hui la norme dominante dans ce domaine, succédant à sa version 1.2 avec des améliorations en termes de fonctionnalités cryptographiques et de résistance aux attaques [ISO 2015].

Le TPM assure plusieurs fonctions primordiales dans l'architecture de sécurité globale. Il permet la génération et le stockage sécurisé de clés cryptographiques, offre des

capacités de mesure et d'attestation de l'intégrité du système via ses registres de configuration de plateforme (PCR), le scellement cryptographique de données, et fournit une source fiable de nombres aléatoires [Arthur 2015].

L'écosystème TPM présente plusieurs implémentations, chacune avec ses compromis entre sécurité, coût et intégration.

Le dTPM constitue l'implémentation la plus traditionnelle et, théoriquement, la plus sécurisée. Cette forme de TPM est un composant matériel physiquement, généralement sous forme de puce dédiée, connecté à la carte mère via un bus LPC (Low Pin Count) ou SPI (Serial Peripheral Interface). Cette séparation physique confère au dTPM un niveau d'isolation supérieur face aux attaques logicielles. L'ANSSI recommande explicitement cette implémentation pour les environnements à haute sensibilité [ANSSI 2019].

Le fTPM représente une évolution plus récente, consistant en une implémentation logicielle exécutée dans un environnement privilégié du processeur. C'est ce qu'utilise AMD avec le Platform Security Processor et Intel avec son Platform Trust Technology (PTT). Cette approche présente l'avantage de réduire les coûts et la complexité de conception. Toutefois, le fTPM partage partiellement son environnement d'exécution avec d'autres composants du système, réduisant son isolation face à certaines catégories d'attaques.

Ces implémentations ne sont pas exemptes de vulnérabilités, la vulnérabilité TPM-FAIL a démontré la possibilité d'extraire des clés privées via des attaques temporelles contre des TPM. Sur le fTPM d'Intel, les chercheurs ont récupéré une clé ECDSA après seulement 1 300 observations en moins de deux minutes et sur un TPM matériel de STMicroelectronics cette clé a été extraite après 40 000 observations en 80 minutes. [TPM-FAIL 2020]

Le vTPM constitue une implémentation entièrement logicielle, généralement déployée dans des environnements virtualisés pour fournir des fonctionnalités TPM aux machines virtuelles. Google a été le premier fournisseur majeur de cloud à offrir des TPM virtualisés dans le cadre de leur produit. L'hyperviseur gère généralement ces instances de vTPM dont le niveau de sécurité dépend fondamentalement de la robustesse de l'hyperviseur et de l'environnement d'exécution hôte.

L'efficacité de ces différentes variantes de TPM face aux menaces dépend non seulement de leur mode d'implémentation, mais également de leur intégration cohérente dans l'architecture de sécurité globale du système. Les évolutions récentes dans le domaine des TPM tendent vers une intégration du fTPM.

3.3 Solutions pour systèmes embarqués

3.3.1 ARM TrustZone / RISC-V PMP

ARM TrustZone représente la technologie de sécurité dominante dans l'écosystème des systèmes embarqués modernes. Introduite par ARM, en 2004, cette technologie implémente un concept de séparation matérielle entre deux mondes d'exécution : un monde sécurisé (Secure World), pour les opérations critique et un monde normal (Normal

World), dans lequel l'OS tournera. Initialement conçue pour les processeurs haut de gamme, cette technologie a été étendu sur un grand ensemble de microcontrôleurs et s'est imposée comme un élément clé dans la sécurisation des applications IoT sensibles, notamment pour le paiement mobile, l'authentification biométrique et la protection des clés cryptographiques.

Cette isolation s'étend à tous les niveaux de l'architecture système. Au niveau du processeur, TrustZone implémente un bit d'état Non-Secure (NS) qui détermine si le CPU fonctionne dans le monde sécurisé (NS=0) ou non-sécurisé (NS=1). Pour la mémoire, le Security Attribution Unit (SAU) partitionne l'espace d'adressage, attribuant des régions spécifiques à chaque monde. Les périphériques sont contrôlés par le Security Configuration Controller (SCC) qui définit leur accessibilité depuis chaque monde. Enfin, les bus système propagent le bit NS à travers toutes les transactions, garantissant que la séparation des mondes est maintenue jusqu'aux périphériques externes.

Une étude de 2019 sur un SoC CortexA53 (Raspberry Pi 3) montre que le basculement entre mondes sécurisé et non sécurisé prend ≈ 1520 µs et que les calculs réalisés dans le *Secure World* n'encaissent que < 5 % de perte de performance, seule l'écriture dans le secteur de stockage chiffré souffre d'un ralentissement (débit ÷ 7) [Amacher 2019].

L'architecture RISC-V propose une approche différente mais complémentaire avec son mécanisme Physical Memory Protection (PMP).

Le mécanisme PMP définit 64 régions mémoire qui peuvent être individuellement configurées pour appliquer des permissions d'accès. Les caractéristiques architecturales du PMP sont :

- Hiérarchie des privilèges : Machine-mode (M-mode), Supervisor-mode (S-mode) et User-mode (U-mode)
    - Configuration exclusive M-mode : Seul ce mode peut programmer les registres PMP
- Verrouillage irrévocable : Le bit L empêche toute modification jusqu'au reset du matériel (même par un logiciel qui peut être en M-mode)
- Les permissions de lecture (R), d'écriture (W) et d'exécution (X) par région

Cette architecture de sécurité trouve une application particulièrement pertinente dans le contexte des environnements d'exécution de confiance (Trusted Execution Environment, TEE), où l'isolation totale entre composants sécurisés et non sécurisés constitue un impératif absolu.

3.3.2 Secure Elements et enclaves sécurisées

Un Secure Element (SE) représente l'approche hardware la plus robuste pour la protection cryptographique, généralement sous la forme d'un microprocesseur sécurisé dédié (forme de puce distincte ou intégrée), qui offre un environnement hautement sécurisé.

L'architecture typique d'un Secure Element :

- Protection physique : Blindage métallique, maillage actif

- Capteurs anti-intrusion : Détection tension, température, lumière, fréquence
- Moteur cryptographique : Accélérateurs RSA/ECC, AES, SHA
- Mémoire non-volatile : EEPROM/Flash sécurisée pour clés et certificats
- Interface limitée : SPI/I2C avec authentification des commandes

Les SE se déclinent en plusieurs formats : eSE (embedded Secure Element) soudé à la carte mère, eSIM, module discret SPI ou carte micro-SD sécurisée. Les SE atteignent typiquement Common Criteria EAL5+ ou FIPS 140-3 niveau 2, les rendant adaptés aux applications critiques comme paiements mobiles ou passeports électroniques.

Les enclaves sécurisées (TEE) représentent une évolution du concept de Secure Element, en offrant un environnement d'exécution isolé directement intégré au sein des processeurs principaux. Elles créent une zone protégée tout en partageant certaines ressources avec le processeur hôte. Celle-ci sont utilisé par des technologies comme Intel SGX ou Arm Confidential Compute Architecture.

Parmi les avantages clés des enclaves sécurisées, on trouve :

- La protection des données en cours d'utilisation, chiffrées au sein de l'enclave
- Isolation des algorithmes utilisées
- La résistance aux tentatives de falsification matérielles et logicielles
- Le support du démarrage sécurisé et des mises à jour de firmware authentifiées

3.3.3 Synergie entre Secure Element et enclave

L'intégration synergique des Secure Elements et des enclaves sécurisées représente une proposition d’architecture intéressante pour les systèmes embarqués, combinant les forces respectives de chaque technologie dans une approche défense en profondeur.

Cette architecture hybride s'articule autour :

1. Secure Element comme racine de confiance matérielle
    - Stockage des clés racines dans une mémoire EEPROM protégée
    - Authentification cryptographique de l’enclave via signatures ECDSA
    - Validation de chaque mise à jour de firmware par le SE avant chargement dans l’enclave (« comme un Secure Boot »).
2. Enclave pour l’exécution isolée
    - Exécution des opérations cryptographiques sensibles (chiffrement, signature, génération d’aléas) dans un environnement isolé, « sans impact » sur l’OS hôte.
    - Protection contre les attaques par DMA grâce à l’usage d’un contrôleur d’accès dédié, bloquant tout accès direct à la mémoire de l’enclave.
3. Persistance et attestation via le SE avec des scellement des clés et stockage des certificats
4. Protection d'exécution continue par l'enclave
    - Surveillance active de l’intégrité du code et des données chargés dans l’enclave (measurement & runtime integrity checks).

- Mise en place de mécanismes de remédiation automatique (reset partiel, rollback) en cas de détection d'anomalie.

En combinant ces deux composants, l'enclave déleste le CPU principal pour les tâches de sécurité en temps réel, tandis que le Secure Element assure en arrière-plan la persistance et l'attestation, permettant ainsi une gestion optimisée de l'énergie et une réduction de la complexité logicielle.

Malgré leurs différences fondamentales, toutes ces approches partagent le même objectif : fournir une racine de confiance robuste, tout en tenant compte des contraintes spécifiques des plateformes qu'elles protègent.

3.4 Analyse comparative des solutions

Les architectures de protection matérielle présentées révèlent des compromis fondamentaux entre sécurité, performance et contraintes d'implémentation. Cette section propose une analyse comparative systématique des différentes approches.

Les solutions pour plateformes x86/x64 offrent généralement le niveau de protection le plus élevé. Le TPM discret représente l'approche la plus robuste, avec une isolation matérielle complète et une forte résistance aux attaques physiques, bien que son coût soit significativement plus élevé. Les implémentations firmware (fTPM) d'Intel et AMD constituent un compromis attractif : intégrées directement dans le processeur, elles réduisent les coûts tout en maintenant un niveau de sécurité acceptable pour la majorité des cas d'usage, malgré une résistance moindre aux attaques physiques.

Dans l'écosystème embarqué, les contraintes de ressources imposent des approches différentes. ARM TrustZone s'est imposée comme la solution dominante pour les processeurs de moyenne et haute gamme, offrant une isolation matérielle efficace avec un surcoût de performance minimal (moins de 5% selon [Amacher 2019]). Cette technologie bénéficie d'une intégration native dans l'architecture ARM, éliminant les coûts additionnels tout en maintenant une résistance raisonnable aux attaques. Pour les architectures RISC-V, le mécanisme PMP (Physical Memory Protection) propose une alternative libre.

Les Secure Elements occupent une position particulière dans cet écosystème. Offrant le niveau de protection physique le plus élevé grâce à leur conception dédiée et leurs contre-mesures hardware. Leur coût élevé et leur bande passante limitée par les interfaces de communication (SPI) restreignent cependant leur adoption généralisée.

En synthèse, le choix de la solution « optimale » dépend étroitement du contexte d'application et des menaces spécifiques. Les environnements critiques justifient l'investissement dans des solutions matérielles dédiées (TPM discret, Secure Elements), tandis que les déploiements à grande échelle privilégient souvent les approches intégrées (fTPM, TrustZone). La tendance actuelle vers des architectures hybrides, combinant plusieurs mécanismes de protection, reflète la nécessité d'adapter les défenses à la sophistication croissante des attaques.

Solution	Forces	Faiblesses
TPM discret (dTPM)	Isolation matérielle complète Résistance élevée aux attaques logicielles	Coût plus élevé Vulnérabilité aux attaques sur bus de communication Bande passante limitée
TPM firmware (fTPM)	Coût réduit Performance supérieure Intégration native	Isolation réduite Dépendance à la sécurité du CPU Vulnérabilité aux attaques du processeur
TPM virtuel (vTPM)	Flexibilité maximale Mise à jour simplifiée	Sécurité dépendante de l'hyperviseur Risques d'attaques inter-VM Pas de protection matérielle
ARM TrustZone	Intégration native dans SoC Faible impact énergétique Isolation matérielle légère	Surface d'attaque au niveau du moniteur (logiciel gérant les transitions entre les mondes) Isolation binaire (seulement deux mondes)
RISC-V PMP	Architecture ouverte et flexible Faible surcoût en silicium	Maturité limitée et écosystème en développement
Secure Elements	Protection physique maximale Résistance aux attaques matérielles	Coût important Interface limitée Performance restreinte

Table 1 - Tableau récapitulatif des solutions de sécurité

Le TPM 2.0 comme élément central de protection

4. Le TPM 2.0 comme élément central de protection

À la suite de l'analyse des architectures de protection matérielle présentées dans le chapitre précédent, cette section se concentre sur le Trusted Platform Module 2.0, qui s'est progressivement imposé comme la pierre angulaire des chaînes de confiance modernes. Déployé aussi bien dans les ordinateurs personnels que dans les infrastructures cloud et les systèmes embarqués, le TPM constitue aujourd'hui l'ancre matérielle de référence pour sécuriser les plateformes informatiques. La version 2.0, standardisée par le Trusted Computing Group (TCG) en 2015 et révisée en 2019, représente une évolution majeure par rapport à son prédécesseur. Elle introduit notamment un modèle cryptographique permettant l'évolution des algorithmes (dont le support des algorithmes quantiques), ainsi que le support natif de la cryptographie à courbes elliptiques et des politiques d'accès conditionnelle. Cette section examine successivement l'architecture interne et les fonctionnalités du TPM 2.0 (4.1), ses principaux cas d'usage pour la protection des systèmes (4.2), ainsi que ses limites et vulnérabilités connues (4.3), permettant ainsi d'évaluer son rôle effectif dans les architectures de sécurité.

4.1 Architecture et fonctionnalités du TPM 2.0

4.1.1 Composants et opérations fondamentales

Le TPM 2.0 se présente comme un cryptoprocesseur sécurisé, conçu pour protéger les informations sensibles et garantir l'intégrité des plateformes. En tant que racine de confiance matérielle, il constitue le fondement sur lequel repose l'ensemble de la chaîne de sécurité d'un système.

Au cœur du module de ce TPM 2.0 se trouve un processeur cryptographique qui prend en charge diverses opérations cryptographiques essentielles, la génération de nombres aléatoires, la création et la gestion de clés cryptographiques, ainsi que les opérations de chiffrement, déchiffrement et signature. Ce processeur est complété par plusieurs

générateurs d'algorithmes cryptographiques, parmi lesquels figurent obligatoirement RSA, SHA-1, SHA-256 et HMAC, auxquels s'ajoutent maintenant des algorithmes comme l'ECC (Elliptic Curve Cryptography) et AES [Arthur 2015].

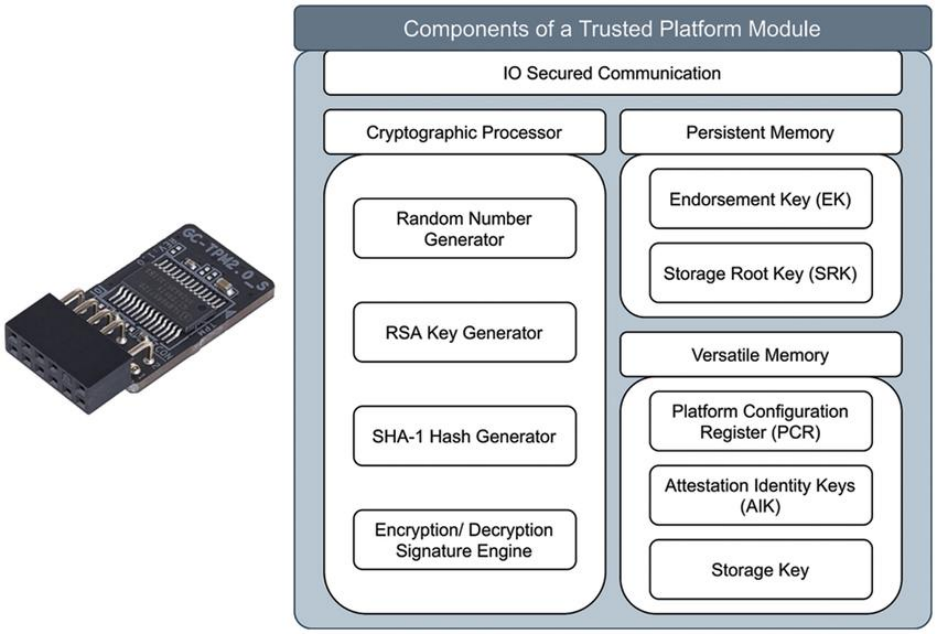

Figure 5 - Architecture interne d'un TPM 2.0 (source : https://www.researchgate.net/figure/Main-components-of-Trusted-Platform-Module-TPM_fig1_363027155)

La mémoire non volatile du TPM constitue un élément critique de son architecture, permettant de stocker de manière sécurisée différents types de données persistantes :

- Les clés d'endossement (Endorsement Keys), générées lors de la fabrication du TPM et uniques à chaque module
- Les clés de stockage (Storage Root Keys), utilisées pour protéger d'autres clés et données sensibles
- Les mesures d'intégrité du système enregistrées dans les registres PCR
- Les politiques de sécurité qui définissent les conditions d'accès aux ressources protégées

Les registres PCR constituent le cœur du mécanisme de mesure d'intégrité. Ces registres ne sont pas modifiables directement, mais uniquement à travers une opération appelée "extension". L'opération d'extension PCR suit la formule : *PCR[i] = Hash(PCR[i] || data_to_extend)*, où || représente la concaténation. Cette propriété mathématique garantit qu'une fois une valeur étendue, il est cryptographiquement impossible de manipuler le registre pour revenir à un état précédent sans réinitialiser entièrement le TPM.

Le TPM 2.0 dispose typiquement de 24 registres PCR (contre 16 pour le TPM 1.2), numérotés de 0 à 23, chacun ayant une fonction spécifique définie par la spécification TCG :

- PCR 0-7 : Réservés pour les mesures BIOS/UEFI et firmware

- PCR 8-15 : Attribués aux composants du système d’exploitation (boot loader, OS kernel, modules)
- PCR 16 : Destiné aux tests et débogage
- PCR 17-22 : Réservés pour le DRTM
- PCR 23 : Support d’application, librement exploitable par l’OS ou les applications utilisateur

Comme le précise la norme ISO/IEC 11889:2015 [ISO 2015], le TPM 2.0 implémente également une hiérarchie de clés, structurée autour de quatre domaines principaux : la hiérarchie d'endossement, utilisée pour les fonctions d'attestation et l'identification unique du TPM, la hiérarchie de plateforme, réservée aux constructeurs et aux administrateurs système, la hiérarchie de stockage, dédiée à la protection des données utilisateurs, et la hiérarchie Null, qui fournit un mécanisme pour les opérations temporaires sans persistance. Cette architecture hiérarchisée permet une séparation claire des responsabilités et des privilèges.

Les opérations fondamentales du TPM 2.0 s'articulent autour de plusieurs fonctions cryptographiques essentielles :

- La génération et la protection de clés cryptographiques, avec la possibilité de créer des clés qui ne peuvent jamais quitter le périmètre sécurisé du TPM ("non-migratable keys").
- L'attestation, permettant de prouver de manière cryptographique l'état d'intégrité d'un système à un tiers vérificateur, basé sur les valeurs PCR signées par une clé d'attestation.
- Le scellement (sealing) et le descellement (unsealing) de données, assurant que les informations sensibles ne peuvent être déchiffrées que si la plateforme se trouve dans un état d'intégrité prédéfini.
- La mesure et l'enregistrement sécurisés de l'état du système via des opérations d'extension des registres PCR

4.1.2 Modèle de sécurité

Le modèle de sécurité du TPM 2.0 repose sur plusieurs mécanismes de protection complémentaires qui garantissent la robustesse de l'ensemble du système.

Le TPM 2.0 implémente un mécanisme de protection contre les attaques par force brute. Ce système verrouille automatiquement le TPM après un nombre défini de tentatives d'authentification échouées (typiquement 32), puis impose un délai croissant entre chaque nouvelle tentative, celui-ci est rénitialisé après un certain temps.

Au-delà de cette protection, le TPM 2.0 introduit un système de sessions d’autorisation démarrées via TPM2_StartAuthSession(), divisées en sessions HMAC et sessions Policy. Les sessions HMAC reposent sur une clé secrète partagée pour authentifier l’utilisateur, tandis que les sessions Policy permettent de composer des règles complexes combinant l’état des PCR, des contraintes temporelles, des signatures externes ou même des localités.

Chaque requête TPM est transmise dans un tampon de commande structuré : un préambule de dix octets qui inclut notamment les champs tag (type de session), commandSize et commandCode. Ce découpage permet une hiérarchisation des commandes.

- Commandes non restreintes : Accessibles sans autorisation préalable, principalement utilisées pour l'interrogation des capacités du TPM (ex. TPM2_GetCapability, TPM2_GetRandom)
- Commandes authentifiées : Requièrent une session d'autorisation valide et sont utilisées pour les opérations cryptographiques courantes (ex. TPM2_Create, TPM2_Sign)
- Commandes privilégiées : Réservées aux propriétaires des hiérarchies TPM, permettent la modification de l'état global du module (ex. TPM2_Clear, TPM2_HierarchyControl)
- Commandes de maintenance : Utilisables uniquement en mode Field Upgrade Mode (FUM) pour les mises à jour du firmware

Enfin, la sécurité physique et logique du TPM 2.0 est garantie par une isolation matérielle renforcée (puce dédiée, stockage non volatile protégé), des mécanismes anti-tampering et des anti-canaux auxiliaires exigeant des implémentations « constant time » avec en plus des commandes contrôlées de verrouillage et de réinitialisation de l'état interne. Cet empilement de protections fait du TPM 2.0 une racine de confiance robuste, capable de répondre aux exigences des environnements PC, cloud et IoT tout en restant extensible face aux nouvelles menaces.

4.2 Cas d'usage de protection avec TPM

Le TPM 2.0 offre un ensemble de primitives cryptographiques qui peuvent être combinées pour répondre à différents besoins de sécurité. Cette section examine les principaux cas d'usage où le TPM apporte une valeur ajoutée significative en termes de protection matérielle, depuis la sécurisation du processus de démarrage jusqu'à la protection des données sensibles en passant par les mécanismes d'attestation.

4.2.1 Protection de l'intégrité du firmware

La protection de l'intégrité du firmware constitue l'un des cas d'usage les plus critiques du TPM 2.0, le TPM offre des mécanismes permettant de détecter toute altération malveillante du firmware et d'établir une chaîne de confiance dès le démarrage.

Le processus de mesure d'intégrité s'inscrit dans le cadre du Secure Boot, où chaque composant du firmware est mesuré cryptographiquement avant son exécution. Ces mesures sont enregistrées dans les registres PCR via des opérations d'extension selon la formule présentée en section 4.1.1, créant ainsi une chaîne de mesures inaltérable qui reflète fidèlement la séquence de démarrage.

La séquence typique de protection comprend :

- L'exécution du code d'initialisation immuable (Root of Trust for Measurement)

- Le calcul d'un hachage cryptographique du firmware UEFI/BIOS
- L'extension de cette mesure dans les PCR appropriés
- La mesure récursive de chaque composant suivant dans la chaîne

Cette approche, permet de détecter toute modification non autorisée des composants firmware. En effet, une altération du firmware entraînerait inévitablement une modification des valeurs enregistrées dans les PCR.

Les standards récents, notamment le RFC 9683 publié par l'IETF en décembre 2024 (« Remote Integrity Verification of Network Devices Containing Trusted Platform Modules »), soulignent l'importance cruciale de cette première mesure effectuée par le RTM, qui constitue le fondement de toute la chaîne de confiance ultérieure.

Au-delà de la simple détection, le TPM permet d'implémenter des mécanismes de réaction aux compromissions via le « scellement conditionnel ». Cette technique garantit que les données sensibles, comme les clés de chiffrement de disque, restent inaccessibles si le firmware a été altéré. Dans le contexte des réseaux d'entreprise, cette capacité permet de mettre en œuvre des politiques d'accès, où seuls les dispositifs présentant un état firmware validé sont autorisés à accéder aux ressources souhaitées.

Il convient toutefois de noter que la protection TPM reste limitée face à certaines attaques matérielles. Une compromission au niveau du circuit intégré ou des bus de communication peut potentiellement contourner ces mécanismes, rappelant l'importance d'une approche défense en profondeur.

4.2.2 Attestation de l'état système

L'attestation représente l'une des fonctionnalités les plus distinctives du TPM 2.0, permettant à une plateforme de prouver cryptographiquement son état d'intégrité à un vérificateur distant. Cette capacité prend une importance dans les architectures Zero Trust où la confiance ne peut être présupposée et doit être continuellement vérifiée.

Le TPM supporte plusieurs formes d'attestation, dont la plus fondamentale est l'attestation des valeurs PCR. Dans ce processus, le vérificateur émet un défi cryptographique (nonce) que le TPM doit signer conjointement avec les valeurs actuelles de ses registres PCR, en utilisant une clé d'attestation (Attestation Identity Key). La signature produite prouve non seulement l'authenticité du TPM, mais également l'état exact du système au moment de l'attestation.

L'attestation peut également s'étendre aux objets protégés par le TPM, tels que les clés cryptographiques. Cela permet de certifier qu'une clé particulière possède certaines propriétés (par exemple, qu'elle a été générée au sein du TPM), renforçant ainsi la confiance dans les opérations cryptographiques réalisées avec cette clé.

Le dernier type d’attestation est l’attestation directe anonyme (Direct Anonymous Attestation). Cette technique permet à un TPM de prouver qu'il est authentique et non compromis, sans révéler son identité unique, préservant ainsi la confidentialité de l'utilisateur.

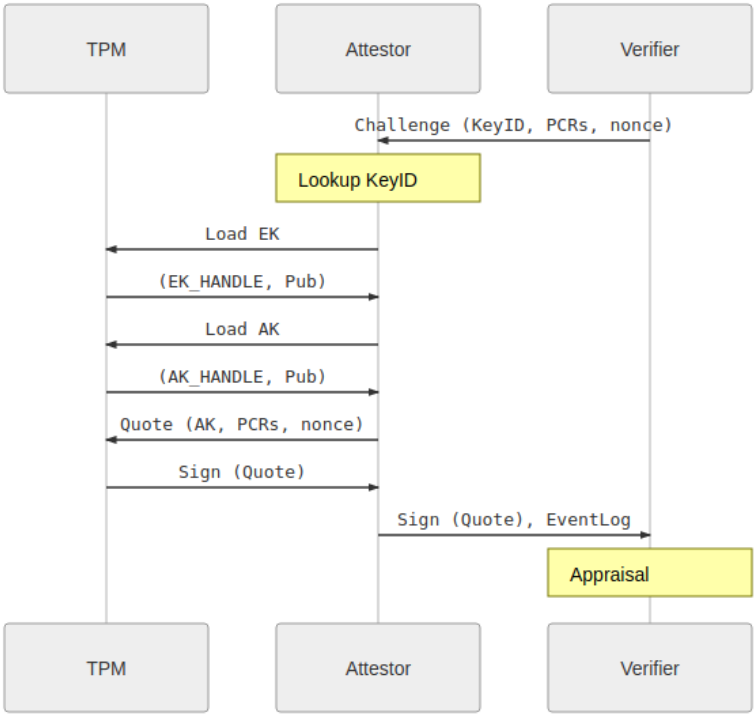

Figure 6 - Diagramme d'attestation avec le TPM : Flux de communication entre le système attesté (Attestor) et le vérificateur (Verifier) montrant les étapes de challenge, signature et vérification (source : https://tpm2-software.github.io/tpm2-tss/getting-started/2019/12/18/Remote-Attestation.html)

Dans le contexte des infrastructures cloud, l'attestation TPM joue un rôle dans la sécurisation des environnements virtualisés. Comme le révèlent les documentations des différents fournisseurs cloud, les vTPM sont désormais largement déployés pour fournir des garanties d'intégrité aux machines virtuelles, permettant ainsi d'étendre les bénéfices de l'attestation aux environnements « multi-locataires ». L'attestation à distance permet aux parties tierces de vérifier l'intégrité de la chaîne de démarrage complète. Celle-ci est aussi utilisé dans l'architecture Zero Trust comme sur Microsoft Azure qui utilise l'attestation TPM pour valider l'intégrité des nœuds de calcul avant d'autoriser l'exécution de charges de travail jugés sensibles.

4.2.3 Scellement de données sensibles

Le scellement de données (sealing) constitue l'un des mécanismes de protection les plus puissants offerts par le TPM 2.0. Cette fonctionnalité permet de chiffrer des données de telle sorte qu'elles ne puissent être déchiffrées que si le système se trouve dans un état d'intégrité spécifique, offrant ainsi une protection contre les attaques visant à extraire des informations sensibles d'un système compromis.

Le processus de scellement associe cryptographiquement les données protégées à un ensemble de valeurs PCR cibles, représentant l'état d'intégrité du système dans lequel le descellement (unsealing) sera autorisé. Techniquement, cette association est réalisée en chiffrant les données avec une clé dérivée des valeurs PCR spécifiées, garantissant ainsi

que seul un système présentant ces mêmes valeurs PCR pourra réaliser l'opération de descellement.

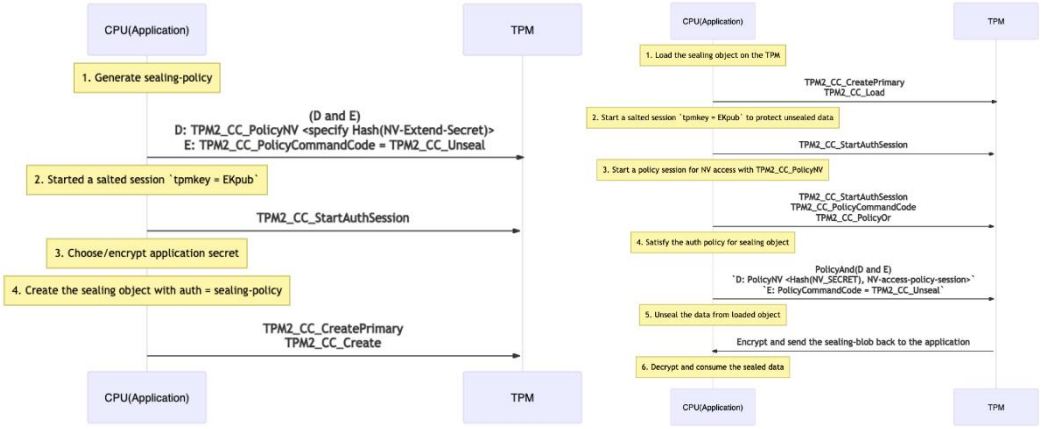

Figure 7 - Processus de scellement/descellement TPM - (a) Création d'un objet scellé avec une politique d'autorisation - (b) Descellement conditionnel des données après vérification de la politique et de l'état du système (source : https://tpm2-software.github.io/2021/02/17/Protecting-secrets-at-TPM-interface.html)

L'utilisation la plus répandue concerne les solutions Full Disk Encryption (FDE) comme BitLocker de Microsoft ou LUKS sous Linux. La clé principale est scellée par le TPM et ne peut être récupérée que si les composants critiques du système n'ont pas été altérés, protégeant ainsi contre les attaques de démarrage.

Dans les environnements cloud, le scellement TPM joue un rôle crucial pour protéger les clés utilisées dans les enclaves sécurisées et les conteneurs confidentiels, garantissant que les données sensibles restent inaccessibles même à l'infrastructure d'hébergement.

Le TPM 2.0 introduit des capacités étendues via les Enhanced Authorization Policies, permettant de définir des conditions de descellement combinant : Des valeurs PCR spécifiques (reflétant l'intégrité du système), l'authentification utilisateur (PIN, mot de passe, biométrie), des signatures cryptographiques externes…

Cette flexibilité permet d'implémenter des modèles de sécurité multicouches adaptés aux exigences spécifiques de chaque cas d'usage. Par exemple, dans une configuration d'entreprise, le descellement d'une clé peut nécessiter à la fois un système intègre (vérification PCR), une authentification forte de l'utilisateur (carte d'accès + PIN), et une validation temporelle (accès uniquement pendant les heures ouvrables). Dans un environnement industriel déployé sur des dispositifs IoT, le scellement TPM peut garantir que les clés cryptographiques utilisées pour la communication réseau ne soient accessibles que si le firmware et les composants critiques n'ont subi aucune altération, limitant ainsi fortement les possibilités d'une intrusion par des modifications matérielles ou logicielles non autorisées.

Un défi opérationnel majeur concerne la gestion des mises à jour légitimes. Les modifications du firmware ou des composants système altèrent inévitablement les valeurs PCR, rendant impossible le descellement des données. Des stratégies appropriées doivent être mises en place :

- Descellement temporaire et re-scellement avec les nouvelles valeurs PCR
- Utilisation de politiques flexibles autorisant plusieurs ensembles de PCR valides
- Mécanismes de récupération d'urgence (recovery keys) pour les situations exceptionnelles

4.3 Limites et vulnérabilités connues

Bien que le modèle de sécurité du TPM 2.0 soit robuste en théorie, de nombreuses faiblesses d'implémentation existent en pratique. Diverses études récentes mettent en évidence ces vulnérabilités pratiques, notamment des failles cryptographiques ou physiques, démontrant ainsi que la sécurité offerte par le TPM n’est efficace que sous réserve d’une mise en œuvre rigoureuse et continue.

4.3.1 Faiblesses d'implémentation

Malgré la robustesse théorique de son modèle de sécurité, le TPM 2.0 n'est pas exempt de faiblesses d'implémentation qui peuvent compromettre significativement les garanties qu'il est censé fournir.

Au-delà des attaques par canaux auxiliaires comme TPM-FAIL décrites précédemment (voir section 3.2.2), l'écosystème TPM 2.0 a révélé une multiplicité de vulnérabilités d'implémentation affectant différentes couches du système. Une vaste étude de Svenda [TPMScan 2024] a révélé une grande variabilité dans la qualité des implémentations disponibles sur le marché. L'étude a notamment identifié des déficiences dans la génération de nombres aléatoires de certains TPM, rendant potentiellement vulnérables l'ensemble des opérations cryptographiques qui en dépendent.

En 2023, des chercheurs de Quarkslab ont découvert deux vulnérabilités majeures (CVE-2023-1017 et CVE-2023-1018) dans l'implémentation de référence du TPM 2.0 fournie par le Trusted Computing Group. Ces vulnérabilités, respectivement de type dépassement de tampon en écriture et en lecture, peuvent être déclenchées par des applications en mode utilisateur envoyant des commandes TPM 2.0, plus particulièrement la fonction CryptParameterDecryption() qui est utilisé pour traiter les paramètres chiffrés des commandes TPM. Selon l'analyse de SecurityWeek, cette faille permettait à un attaquant authentifié disposant d'un accès local d'accéder en lecture à des données sensibles ou de remplacer des données normalement protégées par le TPM, comme les clés cryptographiques. L'impact potentiel inclut la divulgation d'informations sensibles, l'élévation de privilèges, et dans certains cas, l'exécution arbitraire de code au sein du TPM. [Falcon 2023]

Comme l'a révélé la publication détaillée de Quarkslab, ces failles affectent potentiellement des milliards d'appareils, y compris des TPM matériels et des implémentations logicielles utilisées dans les solutions de virtualisation majeures comme VMware, Microsoft Hyper-V et QEMU. [Falcon 2023]

Au-delà des vulnérabilités purement cryptographiques, des faiblesses ont également été identifiées dans la mise en œuvre des mécanismes de protection physique des TPM. Bien que conçus pour résister aux tentatives d'extraction physique d'informations, certains

TPM se sont révélés vulnérables à des techniques avancées d'analyse invasive, telles que l'analyse par sonde électromagnétique ou la microscopie à faisceau d'ions focalisés [Forgette 2022]. Ces vulnérabilités remettent en question l'hypothèse fondamentale selon laquelle les secrets stockés dans le TPM demeurent inaccessibles même face à un attaquant disposant d'un accès physique au dispositif.

Les implémentations firmware du TPM (fTPM), qui exécutent les fonctionnalités TPM au sein d'environnements d'exécution sécurisés comme Intel SGX ou ARM TrustZone plutôt que dans un composant matériel dédié, présentent leurs propres vulnérabilités spécifiques. Ces implémentations héritent potentiellement des vulnérabilités de leur environnement d'exécution sous-jacent, comme l'ont démontré diverses attaques contre les technologies d'enclaves sécurisées [Raj 2016]. En 2022, AMD a d'ailleurs annoncé que leur implémentation fTPM pouvait, causer des problèmes de performance, nécessitant une mise à jour du BIOS pour y remédier.

4.3.2 Contournements pratiques

Au-delà des faiblesses d'implémentation intrinsèques au TPM lui-même, diverses techniques de contournement pratique ont été développées pour neutraliser les protections offertes par le TPM 2.0, notamment dans le contexte de la sécurisation du processus de démarrage et de la protection des données. Comme l'ont souligné plusieurs chercheurs [Svenda 2024], le modèle de sécurité du TPM repose sur l'hypothèse fondamentale que tous les composants de la chaîne de démarrage jusqu'au point de mesure sont exempts de vulnérabilités, une hypothèse irréaliste dans les systèmes.

Les TPM sont généralement connectés au système principal sur des bus standardisés (SPI, I2C ou LPC). Des attaquants peuvent intercepter ou modifier les communications sur ces bus, potentiellement en injectant des commandes malveillantes ou en capturant des informations sensibles. Bien que ces attaques nécessitent un accès physique, elles peuvent compromettre fondamentalement la sécurité du système TPM [Svenda 2024].

Une approche de contournement concerne les attaques de réinitialisation des PCR. Dans certaines configurations, un attaquant disposant de privilèges administratifs peut forcer la réinitialisation du TPM sans redémarrer le système, effaçant ainsi les mesures d'intégrité enregistrées dans les PCR. Cette manipulation peut permettre de contourner les mécanismes de scellement conditionnés aux valeurs PCR, comme l'a démontré Forgette [Forgette 2022] dans sa présentation « TPM is not the holy way ».

Les attaques par démarrage à froid (Cold Boot Attacks), attaques nécessitant un refroidissement physique de la mémoire (typiquement avec de l'azote liquide) permettent de prolonger la persistance des données et permettre leur extraction. Cette technique permet de récupérer les clés de chiffrement une fois qu'elles ont été déchiffrées par le TPM et quand elles sont chargées en mémoire principale.

Les implémentations de TPM virtuel (vTPM) présentent des vecteurs de contournement spécifiques. Si l'hyperviseur qui héberge le vTPM est compromis, toutes les garanties de sécurité offertes par le vTPM peuvent être invalidées. Cette vulnérabilité est particulièrement préoccupante dans les environnements cloud où les TPM virtuels sont

fréquemment utilisés pour fournir des garanties d'intégrité aux machines virtuelles [Arthur 2015]. Les chercheurs de Quarkslab ont d'ailleurs démontré que les vulnérabilités qu'ils ont découvertes dans l'implémentation de référence du TPM 2.0 affectaient les principales solutions de virtualisation, révélant ainsi un risque d'évasion de machine virtuelle. Dans les environnements cloud, les vTPM introduisent des défis de sécurité supplémentaires liés au partage des ressources physiques. Les attaques de type "cross-VM" peuvent potentiellement exploiter les canaux cachés entre machines virtuelles partageant le même matériel physique pour compromettre l'isolation de ses machines. Cette problématique est particulièrement critique dans les offres de cloud public où l'utilisateur n'a aucun contrôle sur l'infrastructure dont il dépend.

Cette limitation est particulièrement problématique dans le contexte des attaques "Time-of-Check to Time-of-Use" (TOCTOU), où un attaquant peut compromettre le système entre le moment de la mesure d'intégrité et l'utilisation effective des ressources protégées. Le TPM ne peut garantir l'intégrité que jusqu'au moment de la mesure, sans aucune protection contre les compromissions ultérieures. Même un TPM parfaitement sécurisé ne peut garantir la sécurité globale d'un système si ce dernier présente des vulnérabilités au niveau du firmware UEFI/BIOS, du chargeur d'amorçage ou du système d'exploitation.

L'analyse des vulnérabilités du TPM 2.0 révèle un paradoxe fondamental : ce composant censé sécuriser l'ensemble du système introduit lui-même de nouvelles surfaces d'attaque. Les vulnérabilités identifiées, allant des faiblesses cryptographiques aux erreurs d'implémentation, illustrent la difficulté à créer un composant de sécurité véritablement infaillible. Microsoft ayant imposé l'utilisation du TPM en 2021, comme composant de sécurité obligatoire pour Windows 11, quand ce composant lui-même a été affecté par des vulnérabilités critiques.

Ces contournements pratiques illustrent une réalité fondamentale de la sécurité informatique : aucun mécanisme de protection isolé, ne peut garantir une sécurité absolue. Une approche de défense en profondeur, combinant différentes technologies de protection et pratiques de sécurité, reste indispensable pour établir un niveau de sécurité robuste face à des adversaires déterminés.

4.4 Synthèse critique des forces et faiblesses du TPM 2.0

Avant de conclure ce mémoire, il est essentiel de réaliser une synthèse structurée des capacités et limites du TPM 2.0, permettant d'identifier clairement ses domaines d'efficacité et ses points de vulnérabilité.

4.4.1 Forces du TPM 2.0: contextes d'efficacité

Le TPM 2.0 offre une protection significative dans plusieurs scénarios d'attaque, bien que cette protection soit soumise à certaines conditions :

Protection contre les attaques logicielles conventionnelles : Le TPM offre une protection significative pour les clés cryptographiques et secrets contre les attaques purement logicielles, même avec des privilèges élevés dans le système d'exploitation. Les opérations cryptographiques critiques peuvent s'exécuter entièrement dans

l'environnement isolé du TPM, sans jamais exposer les clés privées à la mémoire principale. Toutefois, l'efficacité réelle dépend fortement de la qualité d'implémentation des applications qui interagissent avec le TPM, certaines pouvant inadvertamment exposer des données sensibles en mémoire après utilisation.

Préservation de l'intégrité du démarrage : La capacité de mesure et d'attestation du TPM permet de détecter efficacement les modifications non autorisées du firmware et des composants de démarrage. Cette vérification d'intégrité établit une première ligne de défense contre les bootkits et les rootkits. Il convient cependant de noter que le TPM détecte mais n'empêche pas l'exécution de code malveillant, il conditionne simplement l'accès aux données protégées à l'intégrité du système.

Protection conditionnelle des données : Le mécanisme de scellement garantit que les données sensibles (comme les clés de chiffrement de disque) restent inaccessibles si le système a été altéré, offrant une protection même en cas de vol physique du dispositif. Cette protection demeure efficace contre les attaquants disposant de compétences et de ressources limitées, mais présente des vulnérabilités face aux attaques logiques exploitant la fenêtre temporelle entre le descellement et l'utilisation des données.

Attestation à distance fiable : Le TPM permet de prouver cryptographiquement l'état d'intégrité d'un système à un vérificateur distant, facilitant la mise en œuvre de politiques de sécurité basées sur l'état réel du système plutôt que sur des présomptions de confiance. Cette capacité reste particulièrement précieuse dans les architectures Zero Trust, bien qu'elle ne reflète que l'état du système au moment précis de l'attestation.

4.4.2 Faiblesses du TPM 2.0: scénarios de vulnérabilité

Malgré ses capacités, le TPM présente plusieurs limitations fondamentales :

Le TPM offre une résistance limitée face à un attaquant disposant d'un accès physique prolongé et d'équipements spécialisés. Les attaques par canaux auxiliaires (analyse de consommation, émissions électromagnétiques), les attaques par injection de fautes, et l'interception des bus de communication (SPI, LPC) peuvent compromettre son isolation.

Hypothèse d'intégrité initiale non garantie : Le modèle de sécurité du TPM repose sur l'intégrité du premier code exécuté (CRTM - Core Root of Trust for Measurement). Si ce composant est compromis avant la première mesure, toute la chaîne de confiance s'effondre sans possibilité de détection. Des mécanismes comme le DRTM tentent d'atténuer ce problème en établissant une racine de confiance après le démarrage initial, mais présentent leurs propres limitations et peuvent être contournés par des attaques sophistiquées.

Vulnérabilités d’implémentation : Comme l'ont démontré les failles TPM-FAIL et les vulnérabilités CVE-2023-1017/1018, même un composant de sécurité critique peut contenir des défauts d'implémentation significatifs qui compromettent son modèle de sécurité théorique. Ces vulnérabilités, souvent découvertes bien après le déploiement massif, affectent potentiellement des millions de systèmes et compliquent la mise en place de correctifs à grande échelle.

Protection temporelle limitée : Le TPM ne peut garantir qu'un instantané d'intégrité au moment de la mesure, créant une fenêtre de vulnérabilité TOCTOU. Un système vérifié comme intègre peut être compromis immédiatement après l'attestation.

Isolation imparfaite des implémentations non discrètes : Les fTPM et vTPM héritent des vulnérabilités de leur environnement d'exécution sous-jacent (processeur, hyperviseur), compromettant potentiellement leur isolation.

Le TPM se révèle particulièrement inefficace dans les scénarios suivants :

1. Attaques avec accès physique : Un attaquant disposant d'équipements spécialisés (microscopes électroniques, stations de micro-sondage, générateurs d'impulsions électromagnétiques) peut contourner la plupart des protections du TPM.
2. Compromission précoce de la chaîne de démarrage : Une modification du firmware avant la première mesure ou une corruption du CRTM annule l'efficacité de toute la chaîne de confiance, un vecteur particulièrement exploité par les attaques voulant cibler des infrastructures critiques.
3. Attaques transitives via des périphériques connectés : Les contrôleurs DMA (cartes réseau, GPU) peuvent contourner les protections logicielles et accéder directement à la mémoire, y compris aux zones contenant temporairement des clés descellées par le TPM, même sur des systèmes correctement configurés si l'IOMMU présente des vulnérabilités.
4. Environnements virtualisés partagés : Dans les infrastructures cloud utilisant des vTPM, les attaques inter-VM ou les compromissions de l'hyperviseur peuvent neutraliser l'isolation du TPM virtuel.
5. Gestion de mises à jour : Les modifications normales du système (mises à jour firmware ou OS) altèrent les valeurs PCR, nécessitant des mécanismes de migration des données scellées qui créent souvent de nouvelles vulnérabilités.

Cette analyse confirme que le TPM 2.0, malgré ses capacités cryptographiques robustes, doit être considéré comme un élément nécessaire mais non suffisant d'une architecture de sécurité défensive en profondeur. Sa valeur réside dans sa contribution à élever considérablement le niveau de difficulté des attaques, particulièrement celles d'origine logicielle. Pour certains modèles de menace bien définis et limités, notamment face à des attaquants sans ressources significatives ou sans accès physique, le TPM peut fournir des garanties adéquates. Cependant, il ne représente pas une solution ultime face aux attaquants déterminés disposant de ressources significatives ou d'un accès physique, même relativement bref.

Conclusion

5. Conclusion

Dans le cadre de ce mémoire, nous avons exploré en profondeur les menaces émergentes ciblant les couches matérielles et firmwares des systèmes informatiques modernes, avec une analyse comparative spécifique des mécanismes de protection déployés sur les plateformes x86/x64 et les systèmes embarqués (ARM/RISC-V). Cette étude s'est particulièrement concentrée sur le Trusted Platform Module 2.0 (TPM 2.0), en examinant son rôle critique dans la protection des systèmes à bas niveau.

Nous avons d'abord dressé une cartographie des attaques matérielles et firmwares, soulignant la sophistication croissante des menaces comme les bootkits UEFI, les injections de fautes, les attaques par canaux auxiliaires et les vulnérabilités liées aux interfaces matérielles telles que DMA et JTAG. Cette analyse a révélé que les menaces évoluent constamment et exploitent souvent les limites structurelles des mécanismes de défense traditionnels.

Notre étude comparative a ensuite permis d'établir les spécificités et les contraintes inhérentes à chaque catégorie de systèmes. Tandis que les plateformes conventionnelles bénéficient de ressources matérielles et énergétiques significatives permettant l'intégration de mécanismes de sécurité avancés comme Secure Boot et TPM discret (dTPM), les systèmes embarqués doivent composer avec des contraintes strictes, nécessitant des solutions optimisées telles qu'ARM TrustZone, RISC-V PMP, l'utilisation des Secure Elements et des enclaves sécurisées.

Le TPM 2.0 s'est avéré être un élément central dans la création d'une chaîne de confiance robuste, capable de garantir l'intégrité du firmware à travers des mécanismes cryptographiques solides tels que la mesure d'intégrité et le scellement des données sensibles. Cependant, nous avons également mis en évidence des vulnérabilités notables, tant dans les implémentations matérielles que dans les variantes logicielles du TPM. Des faiblesses d'implémentation aux contournements pratiques via les bus de communication, ces limitations rappellent l'importance d'une stratégie de défense en profondeur plutôt que d'une dépendance exclusive à un composant de sécurité unique.

L'évolution rapide des architectures matérielles et l'émergence de nouvelles classes d'attaques suggèrent plusieurs axes de recherche prometteurs : L'impact de l'informatique quantique sur les mécanismes de protection actuels, notamment les primitives cryptographiques du TPM, l'intégration de mécanismes d'intelligence artificielle pour la détection proactive des attaques matérielles.

En conclusion, bien que le TPM 2.0 constitue une avancée significative dans la sécurisation des systèmes, il ne saurait suffire seul face à la complexité actuelle des menaces matérielles et firmware. Seule une approche intégrée, adaptative et multicouche pourra répondre efficacement aux défis sécuritaires de demain, ouvrant ainsi de nombreuses perspectives pour les recherches futures en cybersécurité matérielle.

Glossaire

6. Glossaire

ARM	Advanced RISC Machine
BIOS	Basic Input/Output System
DMA	Direct Memory Access
dTPM	discrete Trusted Platform Module
DRTM	Dynamic Root of Trust for Measurement
EMFI	Electromagnetic Fault Injection
fTPM	firmware Trusted Platform Module
I2C	Inter-Integrated Circuit
IoT	Internet of Things
JTAG	Joint Test Action Group
KEK	Key Exchange Key
LPC	Low Pin Count
MQTT	Message Queuing Telemetry Transport
PCR	Platform Configuration Register
PK	Platform Key
RISC-V	Reduced Instruction Set Computer - Five
RoT	Root of Trust
SE	Secure Element
SoC	System On a Chip
SPI	Serial Peripheral Interface
TEE	Trusted Execution Environment
TOCTOU	Time-of-Check to Time-of-Use
TPM	Trusted Platform Module

UEFI	Unified Extensible Firmware Interface
vTPM	virtual Trusted Platform Module

Références

7. Référence

1. [AutoFirm 2024] YongLe Chen AutoFirm: Automatically Identifying Reused Libraries inside IoT Firmware at Large-Scale https://arxiv.org/abs/2406.12947
2. [Amacher 2019] Julien Amacher, Valerio Schiavoni. On the Performance of ARM TrustZone. 19th IFIP Interna tional Conference on Distributed Applications and Interoperable Systems (DAIS), Jun 2019, Kongens Lyngby, Denmark. pp.133-151, 10.1007/978-3-030-22496-7_9. hal-02319569 https://inria.hal.science/hal-02319569v1/document
3. [ANSSI 2019] Agence nationale de la sécurité des systèmes d'information, Exigences de sécurité matérielle pour plate-forme x86, Version 1.0, 2019. https://cyber.gouv.fr/sites/default/files/2019/11/anssi-guide-exigences_securite_materielle.pdf
4. [Arthur 2015] Arthur, W., Challener, D., & Goldman, K., A Practical Guide to TPM 2.0: Using the Trusted Platform Module in the New Age of Security, Apress, 2015. https://link.springer.com/book/10.1007/978-1-4302-6584-9
5. [Bakhshi 2024] Bakhshi, T., Ghita, B., & Kuzminykh, I., A review of IoT firmware vulnerabilities and auditing techniques, Sensors 2024, 24, 708 https://www.mdpi.com/1424-8220/24/2/708
6. [Bitdefender 2024] Bitdefender, New side-channel attack targets Intel 13th and 14th gen CPUs https://www.bitdefender.com/en-us/blog/hotforsecurity/new-side-channel-attack-targets-intel-13th-and-14th-gen
7. [Dehbaoui 2012] Amine Dehbaoui, Jean-Max Dutertre, Bruno Robisson, Assia Tria. Electromagnetic Transient Faults Injection on a hardware and software implementations of AES. FDTC 2012, Sep 2012 https://hal-emse.ccsd.cnrs.fr/emse-00742639v1/file/HAL_FDTC2012_Electromagnetic_Transient_Faults_Injection_on_a_hardware_and_software_implementations_of_AES.pdf
8. Protecting System Firmware Storage https://eclypsium.com/blog/protecting-system-firmware-storage/, consulté en 2025
9. [Eclypsium 2023] BMC&C: Lights Out Forever https://eclypsium.com/research/bmcc-lights-out-forever/
10. [ENISA 2023] European Union Agency for Cybersecurity, ENISA Threat Landscape 2023, 2023.

https://www.enisa.europa.eu/sites/default/files/publications/ENISA%20Threat%20Landscape%202023.pdf
11. [ESET 2018] ESET Research, LoJax: First UEFI rootkit found in the wild, courtesy of the Sednit group, 2018. https://web-assets.esetstatic.com/wls/2018/09/Eset-LoJax.pdf
12. [ESET 2024] ESET, UEFI Secure Boot bypass vulnerability, 2024. https://www.techtarget.com/searchsecurity/news/366618102/ESET-details-UEFI-Secure-Boot-bypass-vulnerability
13. [Falcon 2023] Vulnerabilities in the TPM 2.0 reference implementation code, 2023 https://blog.quarkslab.com/vulnerabilities-in-the-tpm-20-reference-implementation-code.html
14. [Forgette 2022] Forgette, B., TPM is not the holy way, 2022. https://www.sstic.org/media/SSTIC2022/SSTIC-actes/tpm_is_not_the_holy_way/SSTIC2022-Article-tpm_is_not_the_holy_way-forgette_7RUa27n.pdf
15. [Frigo 2020] Frigo, P, TRRespass: Exploiting the Many Sides of Target Row Refresh, 2020. https://download.vusec.net/papers/trrespass_sp20.pdf
16. [ISO 2015] ISO/IEC 11889:2015, Information technology — Trusted platform module library. https://trustedcomputinggroup.org/resource/tpm-library-specification/
17. [Jattke 2022] Jattke, P., Van Der Veen, V., Frigo, P., Gunter, S., & Razavi, K., BLACKSMITH: Scalable Rowhammering in the Frequency Domain, 2022. https://doi.org/10.1109/SP46214.2022.9833772
18. [Jerinsunny 2024] Jerin Sunny, STM32 VGlitch – Voltage Fault Injection on STM32, https://jerinsunny.github.io/stm32_vglitch/
19. [Kocher 1996] Kocher P, Timing Attacks on Implementat ions of Diffie-Hellman, RSA, DSS, and Other Systems, 1996 https://link.springer.com/chapter/10.1007/3-540-68697-5_9
20. [Kim 2014] Kim, Y., Daly, R., Kim, J., et al., Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors, ISCA, Juin 2014. https://users.ece.cmu.edu/~yoonguk/papers/kim-isca14.pdf
21. [Matrosov 2019] Matrosov, A., Rodionov, E., & Bratus, S., Rootkits and Bootkits: Reversing Modern Malware and Next Generation Threats, No Starch Press, 2019. https://nostarch.com/rootkits
22. [Meltdown 2018] Moritz Lipp, Michael Schwarz, Daniel Gruss Meltdown: Reading Kernel Memory from User Space https://meltdownattack.com/meltdown.pdf
23. [Mitchell 2022] Robin Mitchell, 33 Critical Vulnerabilities Found in Popular IoT Protocol MQTT https://www.electropages.com/blog/2022/02/researchers-find-mqtt-have-33-vulnerabilities
24. [MITRE 2025] MITRE Corporation, Technique T1542: Pre-OS Boot, ATT&CK Framework. https://attack.mitre.org/techniques/T1542/
25. [NIST 2018] NIST, SP 800-193 - Platform Firmware Resiliency Guidelines, 2018. https://csrc.nist.gov/pubs/sp/800/193/final

26. [Raj 2016] Raj, H., Saroiu, S., Wolman, A., et al., fTPM: A software-only implementation of a TPM chip, USENIX Security Symposium, Août 2016. https://www.usenix.org/system/files/conference/usenixsecurity16/sec16_paper_raj.pdf
27. [RISC-V 2025], RISC-V Physical Memory Protection (PMP) documentation https://sifive.github.io/freedom-metal-docs/devguide/pmps.html
28. [Spectre 2019] Paul Kocher, Spectre Attacks: Exploiting Speculative Execution https://spectreattack.com/spectre.pdf
29. [Chifflier 2019] SSTIC, UEFI et bootkits PCI : étude de cas, 2019. https://cyber.gouv.fr/sites/default/files/IMG/pdf/uefi-pci-bootkits_sstic_article_fr.pdf
30. [Thunderbolt 2020] Thunderbolt Flaws Expose Millions of PCs to Hands-On Hacking, consulté en 2025 https://www.wired.com/story/thunderspy-thunderbolt-evil-maid-hacking/
31. [TPM-FAIL 2020] TPM-FAIL : TPM meets Timing and Lattice Attacks, Daniel Moghimi and Berk Sunar and Thomas Eisenbarth and Nadia Heninger, 2020 https://tpm.fail/
32. [TPMScan 2024] Petr Svenda, Antonin Dufka, Milan Broz, Roman Lacko, Tomas Jaros, Daniel Zatovicand Josef Pospisil TPMScan: A wide-scale study of security-relevant properties of TPM 2.0 chips https://www.researchgate.net/publication/378944595_TPMScan_A_wide-scale_study_of_security-relevant_properties_of_TPM_20_chips
33. [Xeno] Xeno Kovah, BIOS and SMM Internals – SPI Flash Protection Mechanisms, OpenSecurityTraining. https://opensecuritytraining.info/IntroBIOS_files/Day2_03_Advanced%20x86%20-%20BIOS%20and%20SMM%20Internals%20-%20SPI%20Flash%20Protection%20Mechanisms.pdf
34. Wikipedia, Élément sécurisé, consulté en 2025 https://fr.wikipedia.org/wiki/%C3%89l%C3%A9ment_s%C3%A9curis%C3%A9
35. [Vishwakarma 2018] Vishwakarma, G., Exploiting JTAG and Its Mitigation in IOT: A Survey https://www.mdpi.com/1999-5903/10/12/121