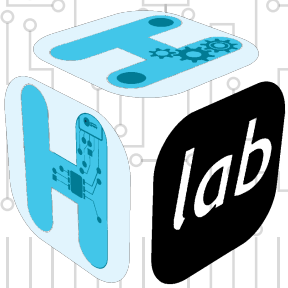
H2LAB-KNOW-TRUSTZONE
08-2026

TrustZone : un guide exhaustif des forces et des faiblesses



Guide H²Lab





Informations

Créée en 2020, H2Lab est une association loi 1901 dont l'objectif est de concevoir, développer et diffuser des solutions open-hardware et open-source dédiées à l'expérimentation autour des systèmes embarqués.
Ses principaux axes de travail sont :

- Conception de plateformes matérielles pour l'analyse hardware et software
- Développement d'alternatives ouvertes aux outils propriétaires, souvent opaques
- Publication de guides techniques et de recommandations pour promouvoir les bonnes pratiques en matière de sécurité, de confidentialité et de durabilité dans l'écosystème des objets connectés et des systèmes embarqués

Soutien aux chercheurs, enseignants et étudiants via la mise à disposition d'outils, de contenus pédagogiques et de formations.
Partenariats académiques pour encourager la mutualisation des ressources et des savoirs.

Historique des révisions

Version	Date	Auteur	Modifications
1.0	Novembre 2026	H2Lab	Version initiale du guide

Table des matières

Informations **i**

1 Glossaire et acronymes **1**
1.1 Glossaire . . . 1
1.2 Liste des acronymes . . . 2

2 But du document **4**
2.1 Objectif . . . 4
2.2 Lectorat . . . 4
2.3 Périmètre et méthode . . . 5
2.4 Modèles d'attaquant . . . 5
2.5 Organisation du guide . . . 5

3 À propos de TrustZone **7**
3.1 Historique et évolutions . . . 7
3.2 L'arrivée de TrustZone-M . . . 8
3.3 TrustZone et le démarrage sécurisé . . . 9
3.4 Une brève mise en perspective . . . 9

4 Principe architectural de TrustZone **11**
4.1 Évolution des cœurs processeurs . . . 11
4.1.1 L'état de sécurité et le bit NS . . . 11
4.1.2 Niveaux d'exception et moniteur . . . 11
4.1.3 Interruptions et routage . . . 12
4.1.4 Bascules de monde et échange de données . . . 12
4.2 TrustZone et la mémoire . . . 13
4.2.1 Un espace d'adressage dédoublé . . . 13
4.2.2 Partitionner les mémoires : le TZASC . . . 14
4.2.3 Points de vigilance . . . 14
4.3 TrustZone et les périphériques . . . 15
4.3.1 Le rôle de l'interconnect . . . 15
4.3.2 Périphériques TZ-aware . . . 15
4.3.3 Le cas du debug . . . 15

5 Les limites architecturales de TrustZone **17**
5.1 L'intégration dans les SoC : une sécurité déléguée . . . 17
5.2 Les maîtres de bus non TrustZone . . . 18
5.3 TrustZone et les architectures multi-cœurs . . . 18
5.4 SMMU et PAMU : étendre le contrôle aux maîtres . . . 19
5.5 Les contrôleurs de domaines matériels . . . 20

5.6 Cartographier une plateforme : la matrice maîtres–ressources 21
5.7 Le besoin d'un démarrage sécurisé 22
5.8 Synthèse des limites 22

6 TrustZone et l'évolution des attaques **23**
6.1 Attaques microarchitecturales 23
6.1.1 Canaux auxiliaires par les caches 23
6.1.2 Exécution spéculative 24
6.1.3 Injection de fautes par la gestion d'énergie 24
6.1.4 Détournement des infrastructures de debug 24
6.1.5 Perturbations de la DRAM 25
6.2 Attaques indirectes par maîtres DMA 25
6.3 La surface logicielle du monde sécurisé 25
6.3.1 Un TCB plus grand que prévu 25
6.3.2 Le fossé sémantique : l'attaque BOOMERANG 26
6.3.3 Cryptographie de confiance mal conçue : le cas Keymaster 26
6.4 Évaluer l'applicabilité d'une attaque publiée 27
6.5 Enseignements 27

7 TrustZone-M **29**
7.1 Présentation de TrustZone-M 29
7.2 Différences avec TrustZone-A 30
7.3 Mémoire et périphériques 31
7.4 Sécurité, isolation et écosystème 31
7.5 Pièges de développement et de portage 32
7.6 Limites et retour d'expérience des attaques 33
7.6.1 Défauts aux frontières : scellement de pile et VLLDM 33
7.6.2 Canaux microarchitecturaux sans caches : BUSted 33
7.6.3 Défauts de plateforme : le cas LPC55S69 34
7.6.4 Attaques physiques 34
7.6.5 Étude de cas : évaluer un produit sur microcontrôleur TrustZone-M . . . 34

8 Conclusion **36**
8.1 Ce que TrustZone garantit, et à quelles conditions 36
8.2 Suffisante, à compléter, ou insuffisante 36
8.3 Grille d'évaluation 37
8.4 Perspectives 38

Licence **42**



1
Glossaire et acronymes

1.1 Glossaire

Attaque microarchitecturale. Attaque exploitant le comportement observable des ressources internes d'un processeur (caches, prédicteurs, files d'exécution, bus) plutôt qu'une faille logicielle, le plus souvent au travers de mesures de temps.

Canal auxiliaire. Vecteur d'information involontaire (temps d'exécution, consommation, rayonnement électromagnétique, contention d'une ressource partagée) permettant de déduire un secret sans violer le contrôle d'accès logique.

Chaîne de confiance. Suite de composants logiciels dont chacun vérifie l'intégrité et l'authenticité du suivant avant de lui transférer l'exécution, depuis une racine de confiance immuable jusqu'à l'application.

Domaine de sécurité. Ensemble de maîtres et de ressources d'un SoC partageant les mêmes droits d'accès matériels. TrustZone définit deux domaines. Les contrôleurs de domaines (xRDC, XMPU, etc.) en définissent davantage.

État de sécurité. État matériel du processeur (*Secure* ou *Non-secure*) qui détermine l'attribut de sécurité des transactions qu'il émet et les ressources auxquelles il peut accéder.

Injection de faute. Perturbation volontaire du fonctionnement d'un circuit (tension, horloge, impulsion électromagnétique, laser) destinée à corrompre une instruction ou une donnée, par exemple un test de sécurité.

Interconnect. Matrice de bus d'un SoC reliant les maîtres (cœurs, DMA, accélérateurs) aux esclaves (mémoires, périphériques) et propageant les attributs des transactions, dont l'attribut de sécurité.

Maître de bus. Composant matériel capable d'émettre des transactions sur l'interconnect de sa propre initiative : cœur de calcul, contrôleur DMA, GPU, contrôleur réseau ou USB, coprocesseur radio.

Moniteur sécurisé. Logiciel s'exécutant au niveau d'exception le plus privilégié d'un cœur Cortex-A (EL3), responsable de la bascule entre monde normal et monde sécurisé et de la préservation des contextes.

Monde normal, monde sécurisé. Les deux environnements d'exécution définis par TrustZone. Le monde normal (*Non-secure*) héberge l'OS généraliste et les applications. Le monde sécurisé (*Secure*) héberge le TEE et les services de confiance.

Pare-feu de bus. Composant matériel filtrant les transactions vers une ressource (plage mémoire, périphérique) selon l'attribut de sécurité, l'identifiant du maître ou le domaine, et rejetant les accès non autorisés.

Périphérique TZ-aware. Périphérique qui échantillonne l'attribut de sécurité des transactions qu'il reçoit, et parfois de celles qu'il émet, et adapte son comportement en conséquence, par opposition à un périphérique protégé en bloc par un filtre externe.

Racine de confiance. Élément matériel ou logiciel immuable (ROM de démarrage, clé publique en fusibles) dont la confiance ne peut être établie par un autre composant et sur lequel repose la chaîne de confiance.

Démarrage sécurisé (secure boot). Processus de démarrage dans lequel chaque étage logiciel est authentifié (signature cryptographique) avant exécution, garantissant l'intégrité de la plateforme à la mise sous tension.

TEE, REE. *Trusted Execution Environment* : environnement d'exécution de confiance hébergé dans le monde sécurisé (OS sécurisé et applications de confiance). *Rich Execution Environment* : environnement généraliste du monde normal (Linux, Android, RTOS).

Application de confiance (TA). Programme s'exécutant dans le TEE, exposant un service (gestion de clés, DRM, paiement) au monde normal au travers d'une interface contrôlée.

Veneer (fonction de jonction). Petit fragment de code placé dans une région de transition (NSC en TrustZone-M) constituant le seul point d'entrée légitime du monde non sécurisé vers une fonction sécurisée.

Bascule de monde (world switch). Transition entre monde normal et monde sécurisé : via le moniteur EL3 sur Cortex-A, via des instructions de branchement dédiées sur Cortex-M.

1.2 Liste des acronymes

AES. Advanced Encryption Standard.

AHB, APB, AXI. Advanced High-performance Bus, Advanced Peripheral Bus, Advanced eXtensible Interface : protocoles de bus de la famille AMBA.

AMBA. Advanced Microcontroller Bus Architecture.

API. Application Programming Interface (interface de programmation).

CCA. (Arm) Confidential Compute Architecture.

CVE. Common Vulnerabilities and Exposures.

DMA. Direct Memory Access (accès direct à la mémoire).

DRM. Digital Rights Management (gestion des droits numériques).

DVFS. Dynamic Voltage and Frequency Scaling (ajustement dynamique de la tension et de la fréquence).

ECDSA. Elliptic Curve Digital Signature Algorithm.

EL. Exception Level (niveau d'exception, EL0 à EL3).

FIQ, IRQ. Fast Interrupt Request, Interrupt Request.

GIC. Generic Interrupt Controller.

GPU. Graphics Processing Unit.

HSE. Hardware Security Engine (enclave de sécurité matérielle).

IDAU. Implementation Defined Attribution Unit.

IOMMU. Input/Output Memory Management Unit.

JTAG, SWD. Joint Test Action Group, Serial Wire Debug : interfaces de debug.

MMU. Memory Management Unit.

MPC, PPC. Memory Protection Controller, Peripheral Protection Controller.

MPSoC. Multi-Processor System-on-Chip.

MPU. Memory Protection Unit.

NS. Non-secure (attribut de sécurité TrustZone).

NSC. Non-Secure Callable (région appelable depuis le monde non sécurisé).

NVIC. Nested Vectored Interrupt Controller.

OTP. One-Time Programmable (mémoire programmable une seule fois, fusibles).

PAMU. Peripheral Access Management Unit (NXP QorIQ).

PSA. Platform Security Architecture.

PSCI. Power State Coordination Interface.

REE. Rich Execution Environment.

ROM. Read-Only Memory.

RTOS. Real-Time Operating System.

SAU. Security Attribution Unit.

SCR. Secure Configuration Register.

SMC. Secure Monitor Call.

SMMU. System Memory Management Unit.

SMP. Symmetric Multi-Processing.

SoC. System-on-Chip (système sur puce).

SPU. System Protection Unit (Nordic Semiconductor).

TA. Trusted Application (application de confiance).

TBBR. Trusted Board Boot Requirements.

TCB. Trusted Computing Base (base de calcul de confiance).

TEE. Trusted Execution Environment.

TF-A, TF-M. Trusted Firmware-A, Trusted Firmware-M.

TLB. Translation Lookaside Buffer.

TZASC. TrustZone Address Space Controller.

TZPC. TrustZone Protection Controller.

xRDC. Extended Resource Domain Controller (NXP).

2
But du document

2.1 Objectif

TrustZone est, depuis vingt ans, la réponse d'Arm au besoin d'isolation forte entre un environnement d'exécution généraliste et des services sensibles partageant le même processeur. La technologie est aujourd'hui présente dans la quasi-totalité des processeurs d'application Cortex-A et, depuis l'architecture Armv8-M, dans une part croissante des microcontrôleurs Cortex-M. Elle est fréquemment mise en avant dans les fiches produit comme un argument de sécurité en soi, formulation qui entretient une confusion : TrustZone est un mécanisme d'isolation, pas une garantie de sécurité. Son efficacité réelle dépend de la qualité de son intégration dans le SoC, des mécanismes complémentaires présents (démarrage sécurisé, pare-feux de bus, contrôleurs de domaines), de la robustesse du logiciel qui s'exécute dans le monde sécurisé et du modèle d'attaquant retenu.

Ce guide a pour objectif de donner au lecteur les moyens d'évaluer ce que TrustZone apporte et ce qu'elle n'apporte pas sur une plateforme donnée. Il décrit le fonctionnement de la technologie au niveau du cœur, de la mémoire et des périphériques, puis analyse ses limites architecturales, les classes d'attaques publiées qui les exploitent et les mécanismes complémentaires qui doivent être présents pour que l'isolation tienne. Les variantes Cortex-A (TrustZone dite *TrustZone-A* dans la suite) et Cortex-M (TrustZone-M) sont traitées séparément lorsque leurs mécanismes diffèrent, ce qui est le cas pour l'essentiel de l'architecture.

Le guide est une référence d'analyse et d'aide à la décision. Il ne constitue ni un tutoriel de développement pour TEE, ni un manuel d'exploitation offensive : les attaques publiées y sont décrites au niveau de leur principe, de leurs préconditions et de leur impact, c'est-à-dire au niveau nécessaire pour décider si elles concernent un produit donné et comment s'en protéger.

2.2 Lectorat

Ce document s'adresse principalement :

- aux architectes système et aux ingénieurs embarqués qui doivent choisir une plateforme ou concevoir le partitionnement de sécurité d'un produit ;
- aux équipes de sécurité chargées d'évaluer une plateforme existante, de construire un modèle de menace ou de préparer une certification ;

- aux responsables produit qui doivent arbitrer entre TrustZone seule, TrustZone complétée par un composant de sécurité dédié (élément sécurisé, HSE), ou une autre approche d'isolation.

La lecture suppose une familiarité générale avec l'architecture des systèmes embarqués (bus, MMU ou MPU, interruptions, chaîne de démarrage). Les notions propres à TrustZone sont définies au fil du texte et reprises dans le glossaire du chapitre 1.

2.3 Périmètre et méthode

Le périmètre couvre TrustZone pour les profils A et M de l'architecture Arm, son intégration dans les SoC et MPSoC, les mécanismes d'isolation complémentaires (TZASC, TZPC, SMMU, contrôleurs de domaines, MPC et PPC) et les classes d'attaques documentées publiquement jusqu'à la rédaction de ce guide. Les technologies voisines (Arm CCA et ses *realms*, éléments sécurisés discrets, enclaves propriétaires) ne sont évoquées que lorsqu'elles éclairent une limite de TrustZone ou une alternative de conception.

Ce guide se concentre sur la solution TrustZone d'Arm. Les notions d'isolation de confiance comparables dans les architectures RISC-V ne sont pas traitées, afin de préserver la lisibilité et la cohérence de l'analyse.

La méthode suivie est constante dans tout le guide : décrire d'abord le mécanisme et ses préconditions, puis son effet observable ou son impact, son applicabilité à un produit réel, et enfin les protections disponibles avec leurs limites. Les affirmations techniques s'appuient sur les spécifications Arm, la documentation des fondeurs et les publications originales des attaques citées. Lorsque la littérature publique est incomplète ou qu'un comportement dépend d'un choix d'intégration non documenté, le guide le signale explicitement plutôt que de généraliser. Une démonstration publiée sur une plateforme donnée n'implique pas qu'un autre produit soit vulnérable : les conditions à vérifier sont indiquées à chaque fois.

2.4 Modèles d'attaquant

Les conclusions de ce guide dépendent systématiquement de la classe d'attaquant considérée : une même plateforme peut être adaptée face à un attaquant logiciel distant et indéfendable face à un laboratoire équipé. Pour éviter les formulations vagues, le guide s'appuie sur la typologie du tableau 2.1, reprise dans l'analyse des attaques (chapitre 6) et dans la grille de décision finale (chapitre 8).

En première approximation, TrustZone vise les classes A1 et A2. Le chapitre 6 montre que même dans ce périmètre nominal, des conditions d'intégration et de qualité logicielle doivent être réunies. Les classes A3 et A4 sont hors du modèle : lorsqu'elles figurent au modèle de menace du produit, TrustZone doit être complétée par des composants conçus pour y résister.

2.5 Organisation du guide

Le chapitre 3 retrace l'historique de TrustZone, son articulation avec le démarrage sécurisé et l'arrivée de TrustZone-M. Le chapitre 4 décrit les principes architecturaux : état de sécurité des cœurs, partitionnement de la mémoire, propagation de l'attribut de sécurité vers les périphériques.

Classe	Attaquant	Capacités typiques
A1	Logiciel, non privilégié	Exécute du code dans le monde normal sans privilège : application malveillante, code atteint par une faille distante. Peut invoquer les interfaces du TEE et mesurer le temps.
A2	Logiciel, privilégié	Contrôle le noyau du monde normal : pilotes, registres de périphériques et de régulateurs, programmation des maîtres DMA accessibles au monde normal. C'est le modèle nominal contre lequel TrustZone prétend protéger.
A3	Physique, non invasif	Accès au produit : interfaces de debug externes, démontage, perturbations bon marché (glitch de tension ou d'horloge, impulsions électromagnétiques), mesures physiques (consommation, rayonnement).
A4	Physique, invasif	Moyens de laboratoire : préparation du composant, microsondage, extraction et modification de circuit, imagerie. Hors de portée de toute isolation sur silicium standard.

TABLE 2.1 – Typologie d'attaquants utilisée dans ce guide.

Le chapitre 5 analyse les limites architecturales : dépendance à l'intégration SoC, maîtres de bus non TrustZone, multi-cœurs, SMMU et contrôleurs de domaines, nécessité d'un démarrage sécurisé. Le chapitre 6 confronte ces limites aux attaques publiées, des canaux microarchitecturaux aux compromissions logicielles de TEE. Le chapitre 7 traite TrustZone-M en détail, de son modèle d'attribution par adresse jusqu'au retour d'expérience des vulnérabilités publiées sur les premiers microcontrôleurs. Le chapitre 8 synthétise l'ensemble sous la forme d'une grille d'évaluation et de critères de décision.

Ce que ce guide n'est pas



Ce guide n'est pas un manuel de programmation de TEE ni un guide d'exploitation. Il ne remplace pas la documentation de référence du SoC évalué : les noms, le périmètre et les valeurs de réinitialisation des contrôleurs de sécurité varient d'un fondeur à l'autre, et seule la documentation du composant (souvent sous accord de confidentialité) fait foi pour un produit donné.

3
À propos de TrustZone

3.1 Historique et évolutions

TrustZone est introduite par Arm en 2003 comme extension de sécurité de l'architecture ARMv6, avec une première implémentation dans le cœur ARM1176JZ-S. La motivation initiale est dictée par le marché de la téléphonie mobile : les opérateurs et les ayants droit exigent des terminaux capables de protéger des contenus (DRM), des secrets d'abonnement et, plus tard, des transactions de paiement, sans supporter le coût d'un processeur de sécurité dédié. Le principe retenu est économe : plutôt que d'ajouter un second processeur, l'architecture dédouble logiquement le processeur existant en deux environnements d'exécution, un *monde normal* pour le système d'exploitation généraliste et un *monde sécurisé* pour les services sensibles, l'isolation étant propagée par le matériel jusqu'aux mémoires et aux périphériques [2].

L'architecture ARMv7-A généralise la technologie à toute la gamme Cortex-A à partir du Cortex-A8. Durant cette période, TrustZone se déploie massivement dans les smartphones, où chaque fabricant développe son propre système d'exploitation sécurisé : QSEE chez Qualcomm, Kinibi chez Trustonic, TEEGRIS chez Samsung, iTrustee chez Huawei. La normalisation des interfaces vient de GlobalPlatform, qui spécifie l'architecture du TEE et les API des applications de confiance [18].

ARMv8-A (2011) restructure le modèle d'exécution autour de quatre niveaux d'exception, EL0 à EL3, et fait du moniteur sécurisé un niveau d'exception à part entière (EL3) [14]. Arm publie en 2013 une implémentation de référence open source du logiciel d'EL3 et de la chaîne de démarrage, Trusted Firmware-A (TF-A) [36], qui devient le standard de fait sur les SoC Cortex-A. Le TEE open source OP-TEE, maintenu par Linaro, joue un rôle comparable côté monde sécurisé. L'extension Armv8.4-A ajoute en 2017 un niveau de virtualisation dans le monde sécurisé (Secure EL2). Il s'agit d'une extension distincte du mécanisme de séparation Secure/Non-secure de TrustZone : lorsqu'il est implémenté et correctement utilisé avec un moniteur d'hyperviseur sécurisé, Secure EL2 peut fournir une isolation supplémentaire entre des partitions sécurisées. TrustZone seule, y compris sur un système multi-cœur, ne fournit pas cette isolation. Le monde sécurisé reste un seul domaine de confiance.

Armv9-A (2021) introduit l'architecture de calcul confidentiel CCA et ses *realms* [8] : l'espace d'adressage physique n'est plus partagé en deux mais en quatre (*Root*, *Secure*, *Realm*, *Non-secure*), avec un contrôle matériel supplémentaire (*Granule Protection Check*) à la sortie de la MMU. TrustZone n'est pas remplacée : elle demeure le socle sur lequel CCA s'appuie, et reste le seul

mécanisme disponible sur l'immense majorité des SoC embarqués en production. Le tableau 3.1 résume ces étapes.

Période	Architecture	Apport pour TrustZone
2003–2004	ARMv6 (ARM1176)	Extensions de sécurité initiales : bit NS, mode moniteur, dédoublement des mondes.
2005–2011	ARMv7-A (Cortex-A8 et suivants)	Généralisation à la gamme Cortex-A, déploiement massif dans les terminaux mobiles.
2011	ARMv8-A	Niveaux d'exception EL0–EL3, moniteur à EL3, AArch64 ; écosystème TF-A et OP-TEE.
2015–2016	Armv8-M (Cortex-M23/M33)	TrustZone-M : déclinaison pour microcontrôleurs, attribution par adresse, sans moniteur logiciel (chapitre 7).
2017	Armv8.4-A	Extension Secure EL2 : virtualisation et isolation supplémentaires entre partitions sécurisées. Ce n'est pas le cloisonnement de base fourni par TrustZone.
2021	Armv9-A (CCA)	Quatre espaces d'adressage physiques, *realms* ; TrustZone conservée comme socle.

Table 3.1 – Principales étapes de l'évolution de TrustZone.

Deux constats guident la lecture de cet historique. D'une part, TrustZone est une technologie mûre, outillée et peu coûteuse : l'isolation est fournie par le cœur et l'interconnect déjà présents, sans silicium additionnel significatif. D'autre part, ses évolutions successives (Secure EL2, CCA, et côté logiciel le durcissement des TEE) répondent chacune à des faiblesses constatées en exploitation réelle, que les chapitres 5 et 6 détaillent : concentration de privilèges dans le monde sécurisé, surface d'attaque des TEE, insuffisance du modèle à deux domaines dans les SoC complexes.

3.2 L'arrivée de TrustZone-M

Annoncée fin 2015 avec l'architecture Armv8-M puis concrétisée par les cœurs Cortex-M23 et Cortex-M33 (2016), TrustZone-M transpose le principe des deux mondes aux microcontrôleurs [11]. La motivation est cette fois celle de l'IoT : des objets connectés à bas coût doivent protéger des clés d'identité, un mécanisme de mise à jour et des fonctions critiques, sur des composants dépourvus de MMU et soumis à des contraintes temps réel et de consommation strictes.

La transposition n'est pas une simple réduction : le mécanisme change de nature. Sur Cortex-A, l'état de sécurité est un état du processeur et la bascule passe par un moniteur logiciel. Sur Cortex-M, la sécurité est un attribut de l'adresse, déterminé par des unités d'attribution matérielles (SAU et IDAU), et la bascule entre mondes s'effectue par de simples branchements contrôlés, en quelques cycles et sans logiciel intermédiaire [4]. Ce choix préserve le déterminisme des latences d'interruption, essentiel sur microcontrôleur, mais déplace la charge de la sécurité vers la configuration statique du partitionnement mémoire, avec des conséquences propres analysées au chapitre 7.

Les premiers microcontrôleurs TrustZone-M arrivent sur le marché entre 2018 et 2020 : Microchip SAM L11 [26], Nordic nRF9160 [28], NXP LPC55S69 [29], STMicroelectronics STM32L5 [34], suivis depuis par la majorité des gammes équipées de cœurs Cortex-M33 et ultérieurs. L'écosystème logiciel s'organise autour de Trusted Firmware-M (TF-M) [37] et du cadre de certification PSA Certified [15].

3.3 TrustZone et le démarrage sécurisé

TrustZone et le démarrage sécurisé sont deux mécanismes distincts mais indissociables, et leur confusion est une source récurrente d'erreurs d'évaluation. TrustZone isole, à l'exécution, deux environnements l'un de l'autre. Elle ne dit rien de l'intégrité du logiciel qui s'exécute dans le monde sécurisé. Or la configuration de toute l'isolation — tables de traduction sécurisées, programmation des pare-feux de bus, attribution des périphériques — est effectuée par du logiciel au démarrage. Si ce logiciel peut être remplacé ou modifié, l'attaquant ne contourne pas TrustZone : il en prend le contrôle et la configure à sa convenance.

Arm formalise la chaîne de démarrage attendue dans les *Trusted Board Boot Requirements* [5], implémentées par TF-A : une ROM immuable (BL1) authentifie un chargeur de second niveau (BL2), qui authentifie à son tour le moniteur d'exécution (BL31), le TEE (BL32) et le chargeur du monde normal (BL33), chaque vérification s'appuyant sur une chaîne de certificats ancrée dans une clé publique dont l'empreinte est stockée en fusibles OTP. La racine de confiance est donc le couple ROM + fusibles, pas TrustZone. TrustZone prolonge à l'exécution une confiance établie au démarrage. La figure 3.1 représente cette chaîne. La section 5.7 revient sur les conséquences pratiques de cette dépendance, et le guide H2Lab consacré à la sécurité des interfaces JTAG/SWD traite le volet debug de la même problématique [1].

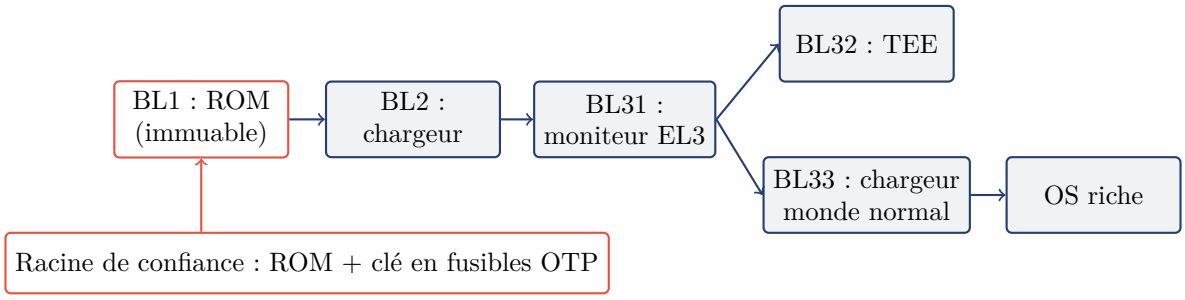


FIGURE 3.1 – Chaîne de démarrage de confiance selon le modèle TBBR implémenté par TF-A. Chaque étage authentifie le suivant. TrustZone prolonge à l'exécution la confiance ainsi établie.

À retenir

TrustZone sans démarrage sécurisé ne protège que contre un attaquant incapable de modifier le logiciel de la plateforme, c'est-à-dire pratiquement personne dans un modèle de menace réaliste. L'évaluation d'une plateforme TrustZone commence donc toujours par celle de sa chaîne de démarrage et de sa racine de confiance.

3.4 Une brève mise en perspective

Avant d'entrer dans l'architecture, il est utile de situer TrustZone parmi les mécanismes d'isolation disponibles dans un système embarqué. La MMU ou la MPU isolent des processus sous le contrôle d'un même noyau : elles ne protègent rien contre ce noyau lui-même. La virtualisation

1. https://blog.h2lab.org/stigs/new-technical-guide-jtag-swd-security/

isole des systèmes invités sous le contrôle d'un hyperviseur. TrustZone isole deux piles logicielles complètes l'une de l'autre, y compris leurs noyaux respectifs, avec une asymétrie assumée : le monde sécurisé peut observer et contrôler le monde normal, jamais l'inverse. Enfin, un élément sécurisé ou un HSE isole physiquement les secrets sur un silicium distinct ou un sous-système dédié, avec des contre-mesures matérielles (capteurs, blindage) que TrustZone ne fournit pas.

Ces mécanismes ne sont pas concurrents mais complémentaires, et le choix dépend du modèle de menace : TrustZone protège le contrôle d'accès logique entre deux mondes sur un même cœur. Elle n'offre par construction aucune garantie contre les canaux auxiliaires microarchitecturaux (les deux mondes partagent les mêmes caches et les mêmes unités d'exécution) ni contre un attaquant physique outillé. Le chapitre 6 quantifie ce constat à partir des attaques publiées, et le guide H2Lab de sélection des composants de sécurité [2] propose une démarche de choix entre ces briques.

2. https://blog.h2lab.org/stigs/new-technical-guide-security-components-selection/

4

Principe architectural de TrustZone

Ce chapitre décrit les trois piliers de TrustZone sur les processeurs d'application : l'état de sécurité du cœur et les mécanismes de bascule entre mondes (section 4.1), le partitionnement de la mémoire (section 4.2) et la propagation de l'isolation vers les périphériques (section 4.3). La déclinaison TrustZone-M, dont les mécanismes diffèrent sensiblement, est traitée au chapitre 7.

4.1 Évolution des cœurs processeurs

4.1.1 L'état de sécurité et le bit NS

Le principe fondateur de TrustZone est de faire d'un même cœur physique deux processeurs virtuels de sécurité : à tout instant, le cœur s'exécute soit dans l'état *Secure*, soit dans l'état *Non-secure*. Cet état est matérialisé par le bit `NS` du registre de configuration sécurisé (`SCR_EL3` sur Armv8-A), accessible uniquement au niveau le plus privilégié [14]. L'état de sécurité ne se limite pas au cœur : chaque transaction émise sur le bus porte un attribut de sécurité dérivé de cet état, si bien que l'isolation est visible de bout en bout, du pipeline jusqu'au contrôleur mémoire. Une transaction non sécurisée qui atteint une ressource réservée au monde sécurisé est rejetée par le matériel, en général avec une erreur de bus.

Pour que les deux mondes puissent coexister sans se corrompre, une partie des registres système est dédoublée (*banked*) : chaque monde possède ses propres registres de configuration de la MMU, ses registres de base des tables de traduction, son vecteur d'exceptions. Le monde sécurisé voit ainsi un processeur complet, avec ses propres traductions d'adresses et sa propre gestion des exceptions, superposé au processeur que voit le monde normal. Les registres généraux, eux, ne sont pas dédoublés : leur sauvegarde et leur nettoyage lors des bascules relèvent du logiciel, point dont l'importance apparaîtra à plusieurs reprises dans la suite.

4.1.2 Niveaux d'exception et moniteur

Sur Armv8-A, le modèle d'exécution comprend quatre niveaux d'exception : EL0 (applications), EL1 (noyaux), EL2 (hyperviseurs) et EL3, niveau le plus privilégié, qui héberge le moniteur sécurisé. EL0 et EL1 existent dans les deux mondes. EL2 n'existait à l'origine que dans le monde

normal, jusqu'à l'introduction de Secure EL2 par Armv8.4-A. EL3 appartient structurellement au monde sécurisé et contrôle le bit `NS`. La figure 4.1 représente cette organisation.

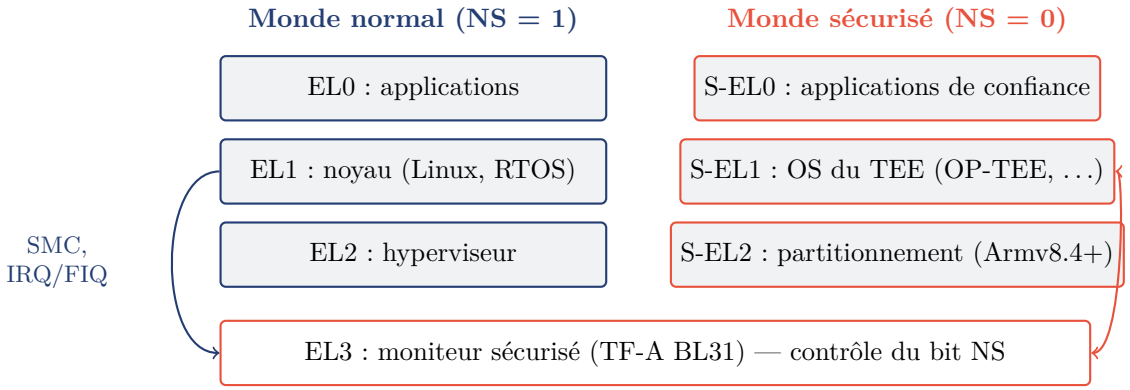


FIGURE 4.1 – Les deux mondes et les niveaux d'exception sur Armv8-A. Le moniteur EL3 est l'unique point de passage entre les mondes.

Le moniteur est le pivot de l'architecture. Toute bascule de monde passe par lui : il est atteint par l'instruction `SMC` (*Secure Monitor Call*) émise depuis EL1 ou EL2, ou par une interruption configurée pour être routée vers EL3. Son rôle est minimal par conception : sauvegarder le contexte du monde sortant (registres généraux, registres système non dédoublés), basculer le bit `NS`, restaurer le contexte du monde entrant et lui rendre la main. Cette minimalité est une exigence de sécurité autant que de performance : le moniteur s'exécute au privilège maximal, et chaque ligne de code qu'il contient appartient à la base de confiance de toute la plateforme.

L'interface entre mondes est normalisée par la convention d'appel SMC (SMCCC) [12] : chaque service est identifié par un numéro de fonction, les arguments et retours passent par les registres généraux. Cette convention supporte aussi bien les appels rapides, non préemptibles, que les appels longs pouvant être interrompus. Les services standard d'Arm l'utilisent, au premier rang desquels PSCI [9], qui confie au moniteur la gestion de l'alimentation des cœurs — choix cohérent, puisque l'extinction et le réveil d'un cœur touchent directement l'état de sécurité de la plateforme (section 5.3).

4.1.3 Interruptions et routage

L'arrivée d'une interruption peut nécessiter une bascule de monde : un périphérique sécurisé doit être servi par le monde sécurisé, même si le cœur exécutait le monde normal, et réciproquement. Le contrôleur d'interruptions générique (GIC) est pour cela TZ-aware : chaque ligne d'interruption est assignée à un groupe, sécurisé (Group 0, historiquement signalé par FIQ) ou non sécurisé (Group 1, signalé par IRQ). Le monde normal ne peut ni observer ni reconfigurer les interruptions du groupe sécurisé, et la priorité des interruptions sécurisées peut être rendue supérieure par construction, garantissant au monde sécurisé de reprendre la main. Le schéma de routage classique — FIQ vers le monde sécurisé, IRQ vers le monde normal — est une convention d'intégration, pas une obligation architecturale, la configuration effective fait partie des points à vérifier lors d'une évaluation.

4.1.4 Bascules de monde et échange de données

Un appel de service typique (par exemple une signature par le TEE) suit le chemin suivant : l'application du monde normal invoque un pilote du noyau, qui prépare les arguments et émet un `SMC`. Le moniteur bascule vers le monde sécurisé. L'OS du TEE ordonnance l'application

de confiance concernée. Le résultat revient par le chemin inverse. Chaque bascule a un coût, de l'ordre de quelques centaines de cycles à quelques microsecondes selon le SoC et ce que le moniteur doit sauvegarder, coût qui borne la granularité raisonnable des services sécurisés : TrustZone se prête aux services appelés ponctuellement (gestion de clés, attestation, vérification de mise à jour), mal au chiffrement de flux à haut débit si chaque bloc exige une bascule. Côté monde normal, l'appel se réduit à charger l'identifiant de fonction et les arguments dans les registres puis à exécuter l'instruction `SMC`, le moniteur faisant le reste :

```
    movw r0, #0x0004    /* identifiant de fonction */
    movt r0, #0x3200    /* plage du service TEE */
    ldr  r1, =shm_base  /* tampon partagé (NS) */
    smc  #0             /* vers le moniteur EL3 */
    /* retour : r0 = statut, r1-r3 = résultats */
```

Les données volumineuses ne transitent pas par les registres : elles sont placées dans une mémoire partagée, c'est-à-dire une région non sécurisée accessible aux deux mondes — le monde sécurisé pouvant, lui, accéder à la mémoire non sécurisée. Cette dissymétrie est le fondement du modèle, mais la mémoire partagée est aussi l'un des points les plus délicats de l'interface : le contenu reste sous contrôle de l'attaquant potentiel pendant que le monde sécurisé le consulte. Le code sécurisé doit donc copier les données avant toute validation (sous peine de *time-of-check/time-of-use*), valider que les pointeurs reçus désignent bien de la mémoire non sécurisée (sous peine d'être transformé en oracle d'écriture dans sa propre mémoire ou dans celle du noyau normal, mécanisme exploité par l'attaque BOOMERANG, section 6.3), et nettoyer les registres et caches de tout résidu sensible avant de rendre la main.

4.2 TrustZone et la mémoire

4.2.1 Un espace d'adressage dédoublé

Du point de vue du système mémoire, l'adresse physique d'une transaction est complétée par son attribut de sécurité : tout se passe comme si le SoC disposait de deux espaces d'adressage physiques parallèles, l'un sécurisé, l'autre non. Les caches et les TLB mémorisent cet attribut avec chaque ligne ou entrée : une ligne de cache chargée par une transaction sécurisée n'est pas servie à une requête non sécurisée, et aucune purge des caches n'est nécessaire lors d'une bascule de monde. Ce marquage, indispensable aux performances, a une conséquence directe sur la sécurité : les deux mondes partagent physiquement les mêmes caches, et s'y disputent les mêmes emplacements. L'isolation est fonctionnelle, pas temporelle — le chapitre 6 montre ce que les attaques par canaux auxiliaires en tirent.

Chaque monde traduit ses adresses au travers de son propre jeu de tables : la MMU utilise les registres de traduction du monde courant. Les tables du monde sécurisé comportent, pour chaque entrée, un bit `NS` indiquant si l'accès résultant est émis comme sécurisé ou non : le monde sécurisé peut ainsi cartographier et accéder à la mémoire non sécurisée (nécessaire aux échanges de données), tandis que le monde normal émet toujours des transactions non sécurisées, quelles que soient ses tables. L'asymétrie est donc structurelle et ne dépend d'aucune configuration logicielle du monde normal.

4.2.2 Partitionner les mémoires : le TZASC

Le cœur émet des attributs. Encore faut-il que les mémoires les fassent respecter. Une mémoire DDR standard ignore tout de TrustZone : le filtrage est réalisé en amont, par un contrôleur dédié placé sur le chemin du contrôleur mémoire, le TZASC (*TrustZone Address Space Controller*), dont l'implémentation Arm de référence est le TZC-400 [3]. Le TZASC découpe l'espace DDR en régions dont il définit, pour chacune, les droits de lecture et d'écriture des transactions sécurisées et non sécurisées. Il peut de plus affiner le filtrage par identifiant de maître (*NSAID*), ce qui permet d'autoriser par exemple un décodeur vidéo à lire une région protégée sans l'ouvrir à tout le monde normal. Toute transaction en violation est rejetée et peut lever une interruption — qu'il convient de router vers le monde sécurisé et de traiter, un rejet silencieux privant la plateforme d'un signal de compromission précieux.

Les mémoires internes suivent le même principe avec d'autres moyens : les SRAM sur puce sont partitionnées par des contrôleurs propres au SoC ou par un découpage fixe, les ROM et les flash internes par des unités comparables. Les registres de configuration de ces contrôleurs sont eux-mêmes des esclaves sécurisés, et disposent en général de bits de verrouillage interdisant toute modification ultérieure jusqu'au prochain reset : les programmer tôt dans le démarrage puis les verrouiller fait partie des attendus d'une intégration correcte.

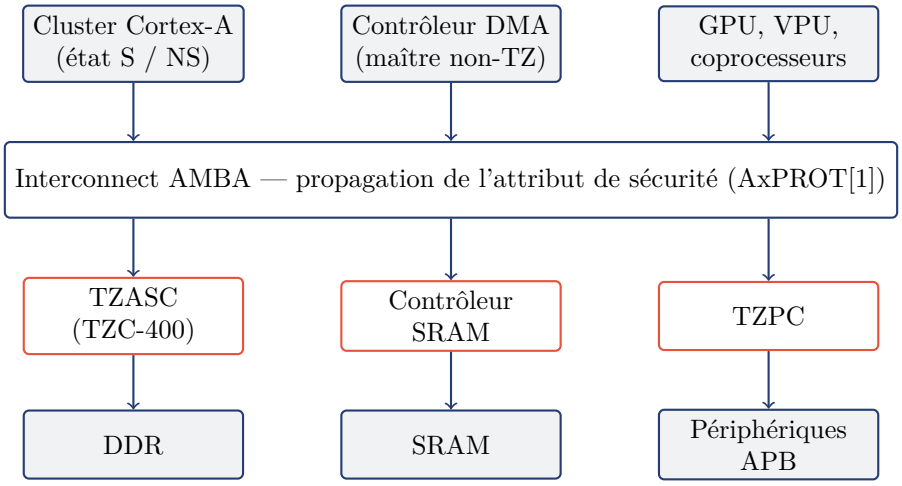


FIGURE 4.2 – Propagation de l'attribut de sécurité dans un SoC. Les pare-feux (en rouge) filtrent les transactions au plus près des ressources. Les maîtres non-TZ posent la question de l'attribut qu'ils émettent (chapitre 5).

4.2.3 Points de vigilance

Le partitionnement mémoire est l'endroit où la distance entre l'architecture et son intégration se mesure le mieux. Trois points concentrent les erreurs observées en évaluation. Le premier est la couverture : chaque chemin d'accès physique à une mémoire doit passer par un filtre. Si la même DDR est accessible par un second port (un accélérateur, un contrôleur d'affichage) non filtré, le TZASC du port principal ne protège rien. Le deuxième est la cohérence des configurations : régions qui se recouvrent, granularité trop grossière forçant à élargir une région sécurisée, alias d'adresses présentant la même mémoire physique sous deux adresses dont une seule est filtrée. Le troisième est le verrouillage : une configuration correcte mais modifiable par un monde normal compromis ne vaut rien. Les bits de verrouillage doivent être positionnés avant le premier transfert de contrôle hors de la chaîne de démarrage de confiance. Des travaux sur SoC programmables

ont montré qu'une seule erreur de configuration d'interconnect suffit à faire s'effondrer l'isolation de l'ensemble [16].

4.3 TrustZone et les périphériques

4.3.1 Le rôle de l'interconnect

L'attribut de sécurité d'une transaction est transporté par l'interconnect d'un bout à l'autre du SoC : sur AXI, il s'agit du signal `AxPROT[1]` [7], sur AHB du signal `HNONSEC`. L'interconnect lui-même participe au contrôle : une matrice comme le NIC-400 peut être synthétisée avec un filtrage de sécurité par port esclave, rejetant en amont les transactions non sécurisées vers les esclaves déclarés sécurisés. La protection des périphériques est donc distribuée entre l'interconnect, des contrôleurs de protection dédiés et les périphériques eux-mêmes — distribution qui varie d'un SoC à l'autre et que la documentation du composant est seule à décrire précisément.

Le TZPC (*TrustZone Protection Controller*) complète ce dispositif pour les périphériques simples : il fournit des bits de configuration, programmables par le monde sécurisé, qui assignent dynamiquement chaque périphérique (ou groupe de périphériques) d'un bus APB à l'un des deux mondes. Un périphérique assigné au monde sécurisé devient invisible au monde normal : toute lecture ou écriture non sécurisée est rejetée. Cette assignation peut évoluer à l'exécution — un contrôleur d'écran tactile peut être prêté au monde sécurisé le temps d'une saisie de code secret, puis rendu au monde normal.

4.3.2 Périphériques TZ-aware

Un périphérique est dit *TZ-aware* lorsqu'il échantillonne lui-même l'attribut de sécurité des transactions qu'il reçoit et adapte son comportement, au lieu d'être simplement masqué dans un monde. Le GIC en est l'exemple canonique : il expose aux deux mondes des vues différentes du même matériel, chacune limitée à son groupe d'interruptions. Les accélérateurs cryptographiques en sont un autre : un moteur AES TZ-aware peut détenir des clés chargées par le monde sécurisé dans des emplacements inaccessibles aux requêtes non sécurisées, tout en restant utilisable par le monde normal avec d'autres clés. Les horloges temps réel, les compteurs système et les contrôleurs de fusibles relèvent de la même logique : une partie de leurs registres n'est visible que du monde sécurisé.

Un maître de bus peut également être TZ-aware : un contrôleur DMA conçu pour TrustZone émet, pour chaque canal, des transactions portant l'attribut du monde qui a programmé le canal. Cette propriété, qui paraît un détail, est en réalité l'une des lignes de partage majeures entre une intégration TrustZone solide et une intégration de façade : un maître qui ne la possède pas doit être contraint par d'autres moyens, et c'est précisément là que commencent les limites analysées au chapitre suivant.

4.3.3 Le cas du debug

Les interfaces de debug et de trace traversent les mondes par construction : un débogueur capable d'arrêter le cœur et de lire ses registres dans l'état sécurisé annule toute l'isolation. L'architecture prévoit pour cela des signaux d'autorisation distincts pour le debug invasif et non invasif du monde sécurisé (`SPIDEN`, `SPNIDEN`), contrôlés par le cycle de vie du composant (fusibles, authentification de debug). Leur gestion correcte — debug sécurisé désactivé en production, réactivation soumise à authentification — relève de l'intégrateur et du fabricant du produit. La

section 6.1 montre ce qu'il advient lorsqu'elle est négligée, et le guide H2Lab JTAG/SWD détaille les architectures d'authentification de debug.

5

Les limites architecturales de TrustZone

TrustZone définit un modèle, elle ne garantit pas son application. Ce chapitre analyse les limites qui tiennent à l'architecture elle-même et à son intégration dans les SoC réels : la dépendance au travail de l'intégrateur (section 5.1), les maîtres de bus étrangers au modèle (section 5.2), les architectures multi-cœurs (section 5.3), les mécanismes qui étendent le contrôle aux maîtres — SMMU et contrôleurs de domaines (sections 5.4 et 5.5) — et la dépendance au démarrage sécurisé (section 5.7). Ces limites ne sont pas des défauts cachés : elles délimitent le périmètre dans lequel TrustZone tient ses promesses, et chacune correspond à des attaques concrètes présentées au chapitre 6.

5.1 L'intégration dans les SoC : une sécurité déléguée

Arm ne vend pas de puces : TrustZone est livrée sous forme de propriétés architecturales des cœurs et d'IP d'infrastructure (interconnect, TZC, GIC) que chaque fondeur assemble, complète et configure. La sécurité effective d'une plateforme dépend donc de décisions qui échappent entièrement à l'architecture : quels esclaves sont filtrés et avec quelle granularité, quels maîtres peuvent émettre des transactions sécurisées, quelles sont les valeurs de réinitialisation des contrôleurs de protection, quels registres disposent de bits de verrouillage, quelles violations remontent une interruption. Deux SoC équipés des mêmes cœurs Cortex-A peuvent offrir des niveaux d'isolation sans rapport l'un avec l'autre.

Cette délégation a trois conséquences pratiques pour l'évaluateur. D'abord, l'évaluation ne peut pas être menée sur la seule foi de l'architecture Arm : elle exige la documentation de sécurité du SoC, souvent incomplète dans les manuels publics et réservée pour partie à des annexes sous accord de confidentialité — l'absence de documentation vérifiable est en soi un résultat d'évaluation. Ensuite, la surface d'erreur de configuration est vaste : la littérature sur les SoC programmables a montré qu'un signal de sécurité laissé à une valeur par défaut dans la configuration de l'interconnect, ou un périphérique omis du périmètre d'un contrôleur, suffit à ouvrir un contournement complet [16]. Enfin, le logiciel de référence du fondeur (chaîne de démarrage, moniteur adapté, pilotes) fait partie du périmètre évalué : une plateforme correctement

câblée mais livrée avec une configuration logicielle permissive des pare-feux présente la même exposition qu'une plateforme mal câblée.

Point d'évaluation



Pour chaque ressource sensible du SoC (mémoire, périphérique, fusibles, debug), identifier le composant qui fait respecter l'attribut de sécurité, sa valeur à la réinitialisation, le logiciel qui le configure, et l'existence d'un verrouillage. Toute ressource dont la protection ne peut être tracée de cette manière doit être considérée comme non protégée.

5.2 Les maîtres de bus non TrustZone

Un SoC moderne compte des dizaines de maîtres de bus autres que les cœurs : contrôleurs DMA généralistes, GPU, processeurs vidéo et d'affichage, contrôleurs USB, Ethernet, stockage, modems et coprocesseurs radio, moteurs de traitement d'image. Le modèle TrustZone suppose que chaque transaction porte un attribut de sécurité pertinent. Or la plupart de ces maîtres sont des IP conçues indépendamment de TrustZone, incapables de gérer deux contextes de sécurité. L'intégrateur doit alors trancher : câbler l'attribut du maître à une valeur fixe, ou le rendre configurable par le monde sécurisé.

Le cas d'un maître câblé non sécurisé est le plus sain : le maître est confiné au monde normal, les pare-feux le tiennent à distance des ressources sécurisées, et la seule conséquence est fonctionnelle — le monde sécurisé ne peut pas déléguer de transferts à ce maître. Le cas dangereux est celui du maître dont l'attribut est configurable, ou pire, câblé sécurisé pour des raisons de commodité fonctionnelle : tout logiciel capable de programmer ce maître hérite de fait de ses privilèges de bus. Un contrôleur DMA émettant des transactions sécurisées mais programmable depuis le monde normal est un oracle de lecture et d'écriture dans la mémoire sécurisée, c'est-à-dire une annulation pure et simple de l'isolation (section 6.2). La règle d'intégration correspondante est simple à énoncer : le registre de programmation d'un maître doit être au moins aussi protégé que les transactions que ce maître peut émettre. Sa vérification exhaustive, maître par maître, est l'un des exercices les plus rentables d'une évaluation de plateforme.

Le périmètre du problème dépasse les contrôleurs DMA explicites : tout coprocesseur exécutant son propre micrologiciel (modem, GPU, contrôleur de gestion d'énergie) est un maître programmable dont le micrologiciel, rarement audité et parfois mis à jour par des canaux distincts, appartient de fait à la base de confiance si ses transactions peuvent être sécurisées. Les contrôleurs de domaines (section 5.5) et les SMMU (section 5.4) sont les réponses structurelles à ce problème.

5.3 TrustZone et les architectures multi-cœurs

Sur une plateforme SMP, chaque cœur possède son propre état de sécurité : à un instant donné, certains cœurs exécutent le monde normal pendant que d'autres exécutent le monde sécurisé. L'architecture le prévoit — les attributs de transactions et le marquage des caches

restent corrects, y compris dans le trafic de cohérence [7] — mais la concurrence élargit la surface d'erreur logicielle. Le moniteur et le TEE doivent être réentrants ou sérialiser correctement leurs chemins critiques. Les fenêtres de bascule, pendant lesquelles un cœur manipule les deux contextes, doivent rester insensibles à ce que font les autres cœurs. L'état sécurisé global (configuration des pare-feux, tables du TEE) doit également être protégé contre les accès concurrents. Les implémentations de référence (TF-A, OP-TEE) traitent ces points, mais les portages et les moniteurs propriétaires doivent être évalués sur ce critère.

Limite fondamentale : un seul domaine sécurisé

⚠ TrustZone distingue deux états, Secure et Non-secure, et non plusieurs domaines sécurisés indépendants. Sur deux cœurs distincts exécutant chacun un OS TEE en état Secure, le fait que les cœurs soient séparés ne crée aucune frontière de sécurité entre eux : ils appartiennent au même monde sécurisé et peuvent accéder aux ressources Secure que l'intégration matérielle leur rend accessibles. TrustZone ne permet donc, en aucun cas à elle seule, de cloisonner deux OS TEE l'un de l'autre, même si chacun est affecté à un cœur physique différent. Présenter cette configuration comme une isolation fournie par TrustZone est une erreur de compréhension de son modèle. Il faut un mécanisme d'isolation distinct et explicitement configuré (par exemple Secure EL2 avec un moniteur de partitionnement, ou des contrôleurs matériels de domaines couvrant les cœurs, mémoires et périphériques concernés), puis évaluer ce mécanisme pour lui-même.

La gestion d'énergie est le second point de friction : éteindre, réveiller ou faire migrer un cœur touche l'état de sécurité de la plateforme, puisqu'un cœur qui se réveille exécute du code depuis une adresse de reprise qu'il faut protéger, et que ses caches et registres doivent être remis dans un état sûr. C'est la raison d'être de PSCI [9] : le monde normal demande, le moniteur exécute. Toute interface de gestion d'énergie qui contourne EL3 (registre de contrôle d'horloge ou de tension accessible au monde normal) est une anomalie d'intégration — et l'attaque CLKSCREW (section 6.1.3) montre qu'elle peut devenir une arme d'injection de fautes.

Enfin, les MPSoC hétérogènes — clusters Cortex-A accompagnés de cœurs Cortex-M ou R, de DSP et d'enclaves de sécurité — posent une question de périmètre souvent mal comprise : le TrustZone du cluster A ne gouverne que le cluster A. Un Cortex-M compagnon est, du point de vue de l'interconnect, un maître comme un autre : son appartenance à un domaine de sécurité relève des contrôleurs du SoC (section 5.5), pas de l'état de sécurité des cœurs A. Évaluer « la sécurité TrustZone » d'un MPSoC sans cartographier l'ensemble des maîtres et de leurs domaines revient à évaluer la serrure d'une porte dans un mur percé.

5.4 SMMU et PAMU : étendre le contrôle aux maîtres

La SMMU (*System MMU*) [10] est la réponse d'Arm au problème des maîtres DMA : placée sur le chemin d'un ou plusieurs maîtres, elle traduit leurs adresses et filtre leurs accès comme une MMU le fait pour un processus. Chaque flux de transactions est identifié (*StreamID*) et associé à un contexte de traduction. L'interface de configuration de la SMMU est elle-même partitionnée

entre monde sécurisé et monde normal, de sorte que le monde sécurisé peut imposer à un maître des fenêtres d'accès que le monde normal ne peut pas élargir. Chez NXP, la PAMU joue un rôle comparable sur les plateformes QorIQ, avec une logique de fenêtres d'accès par maître.

La SMMU transforme le problème des maîtres non TrustZone en problème de configuration, ce qui est un progrès réel mais pas une solution automatique. Son efficacité suppose trois conditions rarement toutes réunies. La couverture, d'abord : seuls les maîtres effectivement placés derrière une SMMU sont contraints ; sur beaucoup de SoC, une partie seulement des maîtres l'est, et la documentation ne le précise pas toujours. La finesse de configuration, ensuite : des fenêtres trop larges, ouvertes en permanence pour simplifier les pilotes ou préserver les performances, laissent subsister l'essentiel du risque — la famille d'attaques Thunderclap l'a démontré sur les IOMMU des systèmes généralistes, où des fenêtres de transfert surdimensionnées exposaient des structures du noyau à des périphériques malveillants [25]. Le coût, enfin : chaque traduction manquée dans le TLB de la SMMU ajoute une latence de parcours de tables sur le chemin DMA, ce qui pousse en pratique à des configurations statiques et larges — précisément celles qui protègent le moins.

5.5 Les contrôleurs de domaines matériels

Le modèle à deux mondes de TrustZone est trop pauvre pour les MPSoC : entre « le monde normal » et « le monde sécurisé », où placer le cœur temps réel qui pilote la machine, le modem dont le micrologiciel vient d'un tiers, le sous-système multimédia ? Les fondeurs ont répondu par des contrôleurs de domaines matériels qui généralisent le mécanisme : chaque maître du SoC est assigné à un domaine (parmi 4, 8, 16 selon les implémentations), chaque transaction porte l'identifiant de domaine de son maître, et des contrôleurs de ressources (mémoire, périphériques) accordent ou refusent l'accès par domaine, en plus de l'attribut TrustZone. NXP les nomme RDC puis xRDC sur les familles i.MX et S32, Xilinx XMPU et XPPU sur Zynq UltraScale+, Texas Instruments *firewalls* sur les familles Sitara et Jacinto. Le TZC-400 lui-même, avec son filtrage par NSAID, en est une forme embryonnaire [3]. La figure 5.1 illustre le principe sur un MPSoC simplifié.

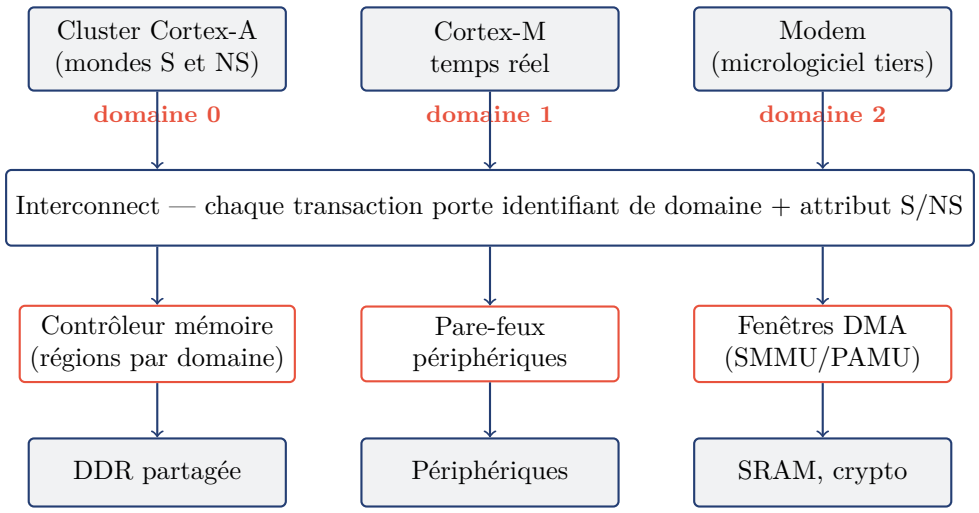


FIGURE 5.1 – Contrôleurs de domaines dans un MPSoC hétérogène. L'assignation des maîtres aux domaines et les droits de chaque domaine sur chaque ressource sont fixés par le logiciel sécurisé au démarrage, puis verrouillés. L'attribut TrustZone reste une dimension de contrôle distincte, qui s'ajoute au domaine.

Ces contrôleurs sont indispensables à une isolation sérieuse dans un MPSoC — ce sont eux qui confinent le modem hors du domaine des cœurs d'application, ou qui protègent le cœur de sûreté des cœurs applicatifs. Mais ils déplacent une nouvelle fois la sécurité vers l'intégration et la configuration : leurs modèles, granularités et valeurs par défaut sont propriétaires et sans portabilité d'un fondeur à l'autre. Leur configuration est au moins aussi complexe que celle des pare-feux TrustZone et doit être verrouillée au démarrage selon la même discipline. L'articulation entre domaine et attribut TrustZone (les deux dimensions coexistent) est une source classique d'angles morts — une ressource correctement filtrée par domaine mais pas par attribut, ou l'inverse. L'évaluation d'un MPSoC passe par la matrice complète maîtres × ressources, les deux dimensions comprises.

5.6 Cartographier une plateforme : la matrice maîtres–ressources

Les sections précédentes convergent vers un même exercice, suffisamment central pour être formalisé : établir, pour la plateforme évaluée, la matrice croisant chaque maître de bus avec chaque ressource sensible, et la confronter au partitionnement voulu par l'architecture du produit. La démarche comporte trois étapes. D'abord, l'inventaire : énumérer les maîtres (cœurs dans chacun de leurs états, DMA, accélérateurs, coprocesseurs à micrologiciel, interfaces de debug agissant comme maîtres) et les ressources (régions de mémoire par usage, périphériques, fusibles, registres des contrôleurs de protection eux-mêmes). Ensuite, pour chaque couple, déterminer la décision matérielle effective — autorisé, refusé, ou dépendant d'une configuration — et le composant qui la rend, à partir de la documentation du SoC. Enfin, confronter le résultat à la politique visée et qualifier chaque écart. Le tableau 5.1 esquisse une telle matrice pour une plateforme simplifiée.

Maître ↓ / Ressource →	**DDR sécurisée**	**DDR normale**	**Moteur crypto**	**Fusibles**
Cortex-A, état sécurisé	A	A	A	A (partiel)
Cortex-A, état normal	R (TZASC)	A	A (clés NS seules)	R (TZPC)
DMA généraliste	**?**	A	**?**	R
GPU	R (NSAID)	A (fenêtre)	R	R
Modem (micrologiciel tiers)	**?**	A (fenêtre SMMU)	R	R

Table 5.1 – Esquisse de matrice maîtres–ressources (A : autorisé, R : refusé par le composant indiqué, **?** : non documenté, à établir). Les cases « ? » sont le résultat principal de l'exercice : chacune est un risque non qualifié.

L'expérience montre que la valeur de l'exercice tient moins aux cases remplies qu'aux cases que la documentation ne permet pas de remplir : chaque « ? » matérialise soit une lacune de documentation à lever avec le fondeur, soit un test à réaliser sur cible, soit un risque à accepter explicitement. La matrice sert ensuite de référence de non-régression : toute évolution du logiciel de démarrage qui modifie une configuration de pare-feu doit pouvoir être confrontée à la politique qu'elle matérialise.

5.7 Le besoin d'un démarrage sécurisé

La section 3.3 a établi le principe. Il faut ici en préciser les conséquences en termes de limites. Toute la configuration de l'isolation — pare-feux, contrôleurs de domaines, attribution des périphériques, tables du TEE — est l'œuvre du logiciel de démarrage. TrustZone est donc exactement aussi robuste que la chaîne qui mène du reset à cette configuration verrouillée. Trois maillons concentrent le risque. La racine de confiance d'abord : ROM de démarrage et clés en fusibles, dont les vulnérabilités ne sont pas corrigeables — une faille de ROM compromet définitivement toutes les pièces produites (section 7.6 pour un cas concret). Les étages intermédiaires ensuite : chaque chargeur doit vérifier le suivant sans erreur de logique ni de cryptographie, et résister aux dégradations (*downgrade*) vers des versions anciennes vulnérables, ce qui exige des compteurs anti-retour en fusibles. Le recouvrement enfin : entre le reset et le verrouillage des pare-feux, la plateforme traverse un état transitoire où les protections ne sont pas encore en place. Les interfaces actives durant cette fenêtre (debug, modes de récupération, démarrage depuis un média externe) doivent être soit désactivées, soit authentifiées.

Il faut ajouter que la réciproque est également vraie : un démarrage sécurisé sans TrustZone (ou sans isolation d'exécution équivalente) garantit l'intégrité du logiciel au lancement mais ne protège rien ensuite. Les deux mécanismes couvrent des phases différentes de la vie du système, et le niveau de sécurité de la plateforme est celui du plus faible des deux.

5.8 Synthèse des limites

Le tableau 5.2 récapitule les limites analysées, leurs conséquences et les mesures dont la présence doit être vérifiée sur la plateforme.

Limite	**Conséquence si non traitée**	**Mesure à vérifier**
Deux OS TEE Secure sur des cœurs distincts	Aucune isolation entre eux du seul fait de TrustZone ; compromission possible de l'autre TEE et de ses ressources Secure	Mécanisme de partitionnement distinct (Secure EL2/SPMC ou contrôleur matériel de domaines), configuration et politique d'accès vérifiées
Sécurité déléguée à l'intégration	Isolation de façade malgré des cœurs conformes	Documentation de sécurité du SoC, cartographie des pare-feux, valeurs de reset, verrouillages
Maîtres non TrustZone	Lecture/écriture de la mémoire sécurisée via un DMA programmable depuis le monde normal	Attribut émis par chaque maître ; protection des registres de programmation ; SMMU ou pare-feux par maître
Multi-cœurs et énergie	Fenêtres de concurrence dans le moniteur ; injection de fautes via DVFS	Moniteur SMP éprouvé ; gestion d'énergie confinée à EL3 (PSCI)
MPSoC hétérogènes	Cœurs compagnons et coprocesseurs hors du périmètre TrustZone	Contrôleurs de domaines (xRDC, XMPU…) couvrant tous les maîtres, configuration verrouillée
Pas de garantie microarchitecturale	Fuites par caches, spéculation, contention (chap. 6)	Implémentations à temps constant, partitionnement des ressources, analyse dédiée
Dépendance au démarrage	Remplacement du moniteur ou du TEE avant verrouillage	Chaîne TBBR complète, anti-retour, gestion du debug et des modes de récupération

Table 5.2 – Synthèse des limites architecturales de TrustZone.

6
TrustZone et l'évolution des attaques

TrustZone met en œuvre un contrôle d'accès logique sur les transactions d'un SoC. Les attaques publiées depuis quinze ans ne contredisent pas ce contrôle : elles le contournent en opérant dans des couches qu'il ne gouverne pas — la microarchitecture partagée, la physique du circuit, les interfaces logicielles du monde sécurisé. Ce chapitre les présente par classe, avec pour chacune le principe, les préconditions, les démonstrations publiées et les protections disponibles. Les attaques propres à TrustZone-M sont traitées avec elle, au chapitre 7.

6.1 Attaques microarchitecturales

6.1.1 Canaux auxiliaires par les caches

Les deux mondes partagent les mêmes caches (section 4.2) : le marquage NS garantit qu'une ligne sécurisée n'est jamais *lue* par le monde normal, mais pas que sa *présence* soit indétectable. Or occuper un emplacement de cache se mesure : en remplissant un ensemble de lignes puis en chronométrant ses propres accès, un attaquant du monde normal observe quels emplacements le monde sécurisé a utilisés entre-temps (technique dite *Prime+Probe*). Si le code sécurisé effectue des accès mémoire dépendant d'un secret — tables de substitution d'un AES, fenêtres d'exposant d'un RSA, branchements dépendant des bits d'un nonce — cette trace d'occupation suffit à reconstruire le secret.

Les travaux ARMageddon ont établi en 2016 que l'arsenal des attaques de cache, développé sur x86, se transpose aux processeurs Arm des terminaux mobiles [21]. La même année, TruSpy démontrait la récupération d'une clé AES exécutée dans le monde sécurisé depuis le monde normal, y compris depuis une application sans privilège [39]. En 2019, l'attaque dite *Cachegrab* de Keegan Ryan a montré la portée produit de la classe : la trace de cache de l'implémentation ECDSA du keystore matériel de Qualcomm (QSEE) fuyait quelques bits du nonce par signature, assez pour reconstruire par treillis la clé privée P-256 d'un keystore Android « protégé par le matériel » [32]. La précondition est modeste — exécuter du code dans le monde normal et déclencher des opérations du service visé — et la cible est le cas d'usage central de TrustZone : la protection de clés.

Les protections relèvent du logiciel sécurisé, pas de TrustZone : implémentations à temps

constant et sans accès mémoire dépendant du secret, aveuglement des opérations à nonce, au besoin partitionnement des caches lorsque le SoC le permet. L'évaluation d'une plateforme doit donc porter sur la qualité des implémentations cryptographiques du TEE — information rarement publique, ce qui plaide pour les TEE auditables ou certifiés. L'analyse détaillée de cette classe d'attaques fait l'objet du guide H2Lab dédié aux attaques microarchitecturales [1].

6.1.2 Exécution spéculative

Les attaques de la famille Spectre et Meltdown [20, 22] ont montré en 2018 que l'exécution spéculative peut faire franchir aux données des frontières de privilège, la fuite s'opérant ensuite par un canal de cache. Les frontières TrustZone sont concernées au même titre que les autres sur les cœurs à exécution dans le désordre : un gadget spéculatif dans le code du moniteur ou du TEE peut être amorcé par le monde normal (les prédicteurs de branchement sont des ressources partagées) et faire fuir de la mémoire sécurisée. Arm a publié la liste des cœurs affectés par variante et les contre-mesures associées, intégrées à TF-A et aux noyaux : barrières de spéculation, assainissement des prédicteurs aux bascules de monde. Pour l'évaluateur, le point de contrôle est double : le cœur utilisé figure-t-il parmi les cœurs affectés, et la version du firmware embarque-t-elle les mitigations correspondantes — avec leur coût, car l'assainissement des prédicteurs renchérit chaque bascule de monde.

6.1.3 Injection de fautes par la gestion d'énergie

CLKSCREW a inauguré en 2017 une classe singulière : l'injection de fautes sans accès physique [35]. Les régulateurs de fréquence et de tension (DVFS) des SoC mobiles sont pilotés par logiciel, souvent sans garde-fou matériel interdisant les points de fonctionnement instables. Depuis le noyau du monde normal, les auteurs ont imposé des couples tension-fréquence hors domaine pendant quelques microsecondes, ciblant précisément une exécution du monde sécurisé sur un autre cœur : les fautes ainsi induites ont permis d'extraire une clé AES du TEE par analyse différentielle de fautes, puis de faire accepter une application de confiance non signée en corrompant la vérification RSA, sur un terminal commercial (Nexus 6). VoltJockey a étendu la démonstration par la voie de la tension sur d'autres plateformes [30].

La précondition — des privilèges noyau dans le monde normal — est exactement celle contre laquelle TrustZone prétend protéger : ces attaques brisent donc le modèle nominal, pas un usage marginal. La cause racine est une gestion d'énergie conçue sans considération de sécurité : les registres DVFS sont des ressources partagées de fait entre les mondes, puisque leurs effets physiques ne connaissent pas le bit NS. Les protections sont l'affaire de l'intégration : plages de fonctionnement bornées par le matériel, régulation confiée à EL3 ou à un contrôleur dédié hors de portée du monde normal, détecteurs de glitch. Lors d'une évaluation, la question « qui peut écrire dans les registres d'horloge et de tension ? » doit recevoir une réponse documentée.

6.1.4 Détournement des infrastructures de debug

L'étude Nailgun a montré en 2019 qu'un cœur peut, via l'infrastructure de debug CoreSight accessible par bus (*memory-mapped*), prendre le contrôle du debug d'un autre cœur — y compris pour inspecter son exécution sécurisée — lorsque les signaux d'autorisation de debug sécurisé sont laissés ouverts, ce qui était le cas sur plusieurs SoC et cartes de développement du commerce, jusqu'à l'extraction d'une empreinte digitale depuis le TEE d'un téléphone commercial [27]. La leçon dépasse le cas d'espèce : les chemins d'accès « latéraux » à l'état des cœurs (debug

1. https://blog.h2lab.org/stigs/new-technical-analysis-microarchitectural-attacks/

inter-processeur, trace, moniteurs de performance) font partie du périmètre de sécurité et doivent être gouvernés par le cycle de vie du composant avec la même rigueur que le port JTAG externe. Les compteurs de performance partagés relèvent de la même vigilance : ils constituent des chronomètres de précision offerts à l'attaquant microarchitectural.

6.1.5 Perturbations de la DRAM

Le marquage de sécurité s'arrête aux portes de la physique. Rowhammer — le basculement de bits dans des rangées DRAM voisines d'une rangée martelée — a été démontré sur plateformes mobiles Arm, avec obtention de privilèges racine depuis une application Android sans permission [38]. Un TZASC filtre les adresses des transactions. Il n'empêche pas une perturbation électrique d'affecter des cellules situées dans une région sécurisée adjacente. L'applicabilité à une cible TrustZone donnée dépend de la technologie DRAM, des protections du contrôleur (rafraîchissement adaptatif, ECC) et de la disposition mémoire. En l'absence de démonstration publique directe contre un TEE, le risque doit être qualifié de théorique mais crédible pour les plateformes à DRAM vulnérable, et traité par le choix de la mémoire plutôt que par le logiciel.

6.2 Attaques indirectes par maîtres DMA

La section 5.2 a décrit la limite. Les démonstrations publiques en confirment l'exploitabilité. Sur les systèmes généralistes, la famille Thunderclap a montré qu'un périphérique malveillant ou compromis dialoguant en DMA exploite les fenêtres laissées ouvertes par l'IOMMU pour lire et corrompre la mémoire du système [25]. La transposition à l'embarqué est directe, avec deux aggravations : d'une part, nombre de SoC embarqués ne disposent d'aucune SMMU, le confinement des maîtres reposant alors uniquement sur des pare-feux statiques dont la configuration est rarement auditée. D'autre part, les maîtres concernés ne sont pas des cartes enfichées par l'utilisateur mais des coprocesseurs internes (modem, Wi-Fi, GPU) exécutant des micrologiciels propriétaires, accessibles à distance pour certains, et dont la compromission a été documentée à plusieurs reprises dans l'industrie.

Le schéma d'attaque type est indirect : compromettre d'abord un coprocesseur par sa surface propre (pile radio, interface réseau), puis utiliser ses privilèges de bus pour atteindre la mémoire du monde sécurisé ou celle du noyau normal. L'évaluation doit donc traiter chaque maître programmable comme un attaquant potentiel et vérifier ce que la matrice de pare-feux l'autorise effectivement à atteindre — exercice qui recoupe la cartographie demandée à la section 5.5. Les protections sont celles du chapitre précédent : SMMU configurée finement, pare-feux par maître verrouillés, et refus par principe de tout maître émettant des transactions sécurisées s'il est programmable hors du monde sécurisé.

6.3 La surface logicielle du monde sécurisé

6.3.1 Un TCB plus grand que prévu

TrustZone réduit en théorie la base de confiance à un moniteur et un petit OS sécurisé. En pratique, les TEE commerciaux ont accumulé les services — gestion de clés, DRM, paiement, biométrie, attestation — et leur base de confiance se mesure en centaines de milliers de lignes, écrites en C, s'exécutant pour l'essentiel à des privilèges élevés. L'étude systématique de Cerdeira

et al. a recensé et analysé 207 vulnérabilités publiées dans les TEE majeurs (Qualcomm, Trustonic, Samsung, Nvidia, Huawei) : dépassements de mémoire dans les applications de confiance, validation insuffisante des paramètres reçus du monde normal, mais aussi des défauts architecturaux durables — absence d'ASLR et de protections mémoire élémentaires dans plusieurs OS sécurisés, applications de confiance révocables mais effectivement réactivables par dégradation de version [17]. Les campagnes d'analyse dynamique menées par émulation (PARTEMU) ont confirmé le constat en trouvant des défauts exploitables dans des applications de confiance déployées à grande échelle [19].

Le point mérite d'être formulé sans détour : la frontière TrustZone tient, mais ce qui vit derrière elle est du logiciel ordinaire, avec le taux de défauts du logiciel ordinaire — et des possibilités d'audit, de supervision et de mise à jour souvent inférieures à celles du monde normal. Ajouter un service au TEE n'augmente pas mécaniquement la sécurité du produit : cela déplace du code vers un environnement plus privilégié et moins observable.

6.3.2 Le fossé sémantique : l'attaque BOOMERANG

La frontière elle-même crée une classe de défauts qui n'existerait pas sans elle. Une application de confiance reçoit du monde normal des requêtes contenant des pointeurs vers la mémoire partagée (section 4.1.4). Le monde sécurisé peut écrire partout dans la mémoire non sécurisée : s'il n'impose pas que les pointeurs reçus restent dans la région légitime de l'application appelante, une application sans privilège peut lui faire écrire dans la mémoire du noyau du monde normal — le TEE devient un adjoint confus (*confused deputy*) qui élève les privilèges de son propre attaquant. L'attaque BOOMERANG a démontré ce schéma sur plusieurs TEE commerciaux simultanément, la cause étant un fossé sémantique : le monde sécurisé ne connaît pas les frontières internes (processus, noyau) du monde normal et ne peut donc pas les faire respecter sans coopération explicite [24]. Les protections consistent à faire valider les tampons par le noyau normal avant transmission et à borner strictement, côté TEE, les plages accessibles. Ces protections sont aujourd'hui présentes dans les TEE du marché. générale, mais le schéma reste un piège de conception pour tout service sécurisé développé sur mesure.

6.3.3 Cryptographie de confiance mal conçue : le cas Keymaster

L'analyse du Keymaster de Samsung (USENIX Security 2022) illustre une dernière leçon : le qualificatif « protégé par le matériel » ne dit rien de la conception cryptographique. Les clés du keystore, scellées par l'application de confiance, étaient chiffrées en AES-GCM avec un vecteur d'initialisation contrôlable par le monde normal sur certaines générations : la réutilisation d'IV permettait de récupérer le matériel de clé, et un mécanisme de compatibilité autorisait à ramener des clés récentes vers le format vulnérable (attaque par dégradation), avec pour conséquence démontrée le contournement d'authentifications FIDO2 [33]. Aucune frontière TrustZone n'a été franchie : l'attaque s'est déroulée entièrement dans les règles du modèle. L'isolation ne remplace ni la revue cryptographique ni la gestion des versions de formats scellés.

6.4 Évaluer l'applicabilité d'une attaque publiée

Les publications d'attaques alimentent régulièrement des conclusions hâtives dans les deux sens : « TrustZone est cassée » ou, symétriquement, « cela ne concerne que la plateforme du papier ». Ni l'une ni l'autre ne constitue une évaluation. Face à une publication, le raisonnement utile suit cinq questions.

La première porte sur la nature du résultat : s'agit-il d'une primitive (fuite de quelques bits, faute exploitable en laboratoire, écriture contrainte) ou d'une chaîne d'exploitation complète sur produit réel ? Une primitive démontrée est un signal sérieux, mais son coût de transformation en compromission effective peut rester élevé. Les publications le quantifient rarement. La deuxième porte sur les préconditions, à rapprocher des classes du tableau 2.1 : privilèges requis dans le monde normal, accès physique, co-résidence sur un cœur, nombre de mesures et répétabilité — une attaque exigeant des millions de mesures sur un service que le produit limite en débit change de statut. La troisième porte sur la dépendance à la plateforme : cœur et révision exacte (les errata et CVE d'Arm sont indexés par révision), présence de la ressource partagée exploitée, choix d'intégration incriminé — un résultat obtenu sur une carte de développement au debug ouvert ne se transpose pas de lui-même à un produit verrouillé. La quatrième porte sur l'état des contre-mesures : existe-t-il un correctif de firmware, une option de compilation, une révision de silicium, et la plateforme évaluée les embarque-t-elle ? La cinquième, enfin, porte sur la généralisation : l'attaque révèle-t-elle un défaut de classe (validation d'interface, ressource partagée non partitionnée) dont il faut chercher d'autres instances sur la plateforme, au-delà du cas démontré ?

Ce questionnaire transforme une publication en éléments de décision : soit la plateforme est hors périmètre démontré et le risque est qualifié avec ses incertitudes, soit elle est dans le périmètre et la publication fournit la précondition, l'impact et, le plus souvent, la contre-mesure à vérifier.

6.5 Enseignements

Le tableau 6.1 synthétise les classes d'attaques, leurs préconditions et les protections correspondantes.

Trois enseignements structurent l'évaluation d'un produit. Premièrement, les attaques les plus dommageables n'ont pas violé le contrôle d'accès TrustZone : elles ont exploité ce que le modèle ne couvre pas (microarchitecture, physique, debug) ou le logiciel qu'il héberge. Évaluer « TrustZone » n'a donc de sens qu'en évaluant la plateforme entière. Deuxièmement, les préconditions des attaques publiées sont pour la plupart réalistes — code non privilégié ou noyau compromis dans le monde normal, c'est-à-dire précisément le modèle de menace nominal de TrustZone. Troisièmement, la qualité du logiciel sécurisé domine le risque résiduel : à intégration matérielle égale, un TEE minimal, audité et à jour change la conclusion d'une évaluation. Un TEE obèse et opaque la condamne.

Classe	Exemples publiés	Préconditions	Protections
Canaux de cache	TruSpy, ARMageddon, Cachegrab/QSEE [39, 21, 32]	Exécution non privilégiée dans le monde normal ; code sécurisé non constant en temps	Implémentations à temps constant ; partitionnement de cache ; audit du TEE
Spéculation	Spectre et variantes [20]	Cœur à exécution spéculative affecté ; firmware sans mitigation	Mitigations Arm dans TF-A/noyau ; choix du cœur
Fautes par DVFS	CLKSCREW, VoltJockey [35, 30]	Privilèges noyau monde normal ; régulateurs accessibles	DVFS confiné à EL3 ; bornes matérielles ; détecteurs
Debug latéral	Nailgun [27]	Debug sécurisé non verrouillé ; accès noyau	Cycle de vie debug ; authentification ; audit SPIDEN/SPNIDEN
Perturbation DRAM	Drammer [38]	DRAM sensible ; motifs d'accès contrôlés	Choix mémoire (TRR, ECC) ; qualification
Maîtres DMA	Thunderclap (transposable) [25]	Maître compromis ou programmable ; pare-feux permissifs	SMMU fine ; pare-feux par maître verrouillés
Logiciel du TEE	SoK 207 vulns, BOOMERANG, PARTEMU, Keymaster [17, 24, 19, 33]	Accès à l'interface du TEE depuis le monde normal	TCB minimal ; validation des paramètres ; anti-dégradation ; revue crypto

TABLE 6.1 – Classes d'attaques contre TrustZone-A : préconditions et protections.

À retenir

⚠ TrustZone trace une frontière de contrôle d'accès. Les attaques publiées passent par-dessus (logiciel du TEE), par-dessous (microarchitecture, physique) ou à côté (maîtres DMA, debug). Un modèle de menace qui attribue à TrustZone une protection contre ces trois directions sans vérification plateforme par plateforme est invalide.

7
TrustZone-M

7.1 Présentation de TrustZone-M

TrustZone-M est l'extension de sécurité de l'architecture Armv8-M, implémentée en option par les cœurs Cortex-M23, M33, M35P, M55 et M85 [11]. Elle reprend le vocabulaire des deux mondes mais en change le mécanisme : sur un microcontrôleur, la sécurité est un attribut de l'*adresse*, pas un état global piloté par un moniteur. L'espace d'adressage, physique et unique (il n'y a pas de MMU), est partitionné en régions sécurisées, non sécurisées et sécurisées appelables (NSC). L'état de sécurité du cœur découle à tout instant de l'adresse du code en cours d'exécution. Deux unités matérielles déterminent cette partition : l'IDAU (*Implementation Defined Attribution Unit*), fixée par le fabricant du microcontrôleur et câblée dans le silicium, et la SAU (*Security Attribution Unit*), programmable par le logiciel sécurisé dans un nombre réduit de régions (typiquement huit). L'attribut effectif d'une adresse est le plus restrictif des deux [4].

Chaque monde dispose de ses propres piles (pointeurs `MSP` et `PSP` dédoublés, avec registres de limite de pile), de sa propre MPU optionnelle, de sa propre table de vecteurs et de deux niveaux de privilège : le modèle offre donc quatre contextes (sécurisé ou non, privilégié ou non). Le NVIC assigne chaque interruption à l'un des mondes (registre `NVIC_ITNS`), et le matériel sauvegarde et masque automatiquement les registres lors d'une exception non sécurisée survenant en code sécurisé, afin de ne pas exposer d'état résiduel.

La bascule entre mondes est directe et matérielle. Un appel du monde non sécurisé vers le monde sécurisé doit aboutir dans une région NSC, sur une instruction `SG` (*Secure Gateway*) : tout branchement vers une adresse sécurisée hors de ces points d'entrée lève une faute. Les fonctions sécurisées exposées sont donc précédées de veneers générés par la chaîne de compilation (extension CMSE), qui constituent la liste exhaustive des points d'entrée. Dans l'autre sens, le code sécurisé appelle le monde non sécurisé par les instructions `BLXNS` et `BXNS`, le matériel masquant les registres et empilant une adresse de retour spéciale (`FNC_RETURN`) côté sécurisé. Aucun moniteur, aucun appel système : la transition coûte quelques cycles et préserve le déterminisme des latences, y compris pour les interruptions, qui peuvent préempter directement dans l'autre monde. Un point d'entrée sécurisé se matérialise ainsi par un veneer de deux instructions, généré par la chaîne de compilation :

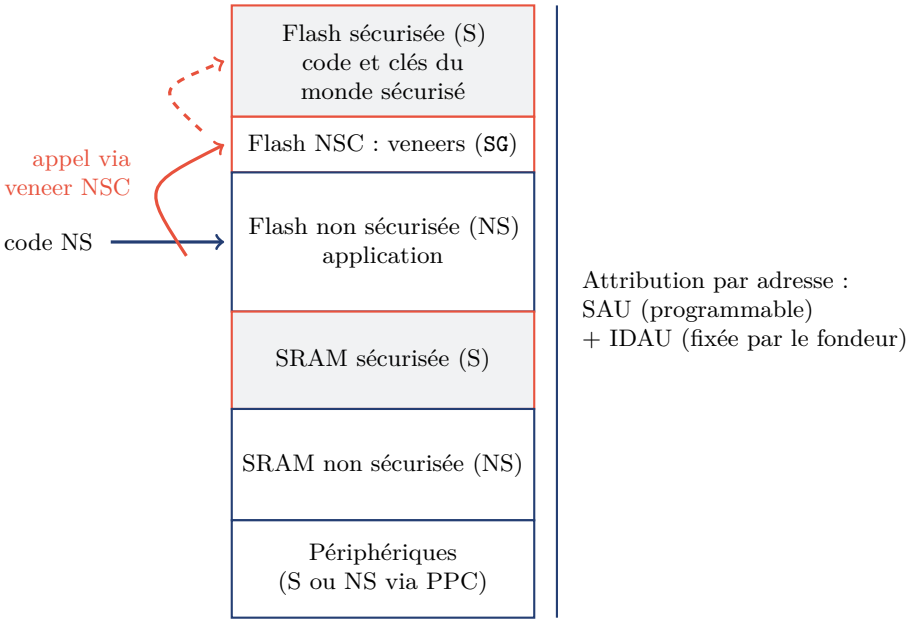


FIGURE 7.1 – Partitionnement type d'un microcontrôleur TrustZone-M. Le monde non sécurisé n'entre dans le code sécurisé que par les veneers de la région NSC, dont la première instruction est `SG`.

```
    .global tz_sign
tz_sign:                        /* en région NSC          */

    sg                          /* point d'entrée unique  */
    b.w  __acle_se_tz_sign      /* vers la région S       */
```

7.2 Différences avec TrustZone-A

Le tableau 7.1 résume les différences structurantes entre les deux déclinaisons.

Deux conséquences méritent d'être soulignées. La première est favorable : l'absence de moniteur supprime une pièce de logiciel critique et le modèle d'attribution par adresse, essentiellement statique, se prête à l'analyse — la surface de transition se réduit à la liste des veneers, vérifiable à la construction du binaire. La seconde l'est moins : la frontière de sécurité est désormais une frontière d'*appel de fonction*, traversée potentiellement des milliers de fois par seconde, avec passage d'arguments par registres et pointeurs. Chaque fonction d'entrée sécurisée est exposée aux mêmes exigences qu'un appel système de noyau : validation de l'appartenance des pointeurs au monde non sécurisé (instructions `TT` et routines CMSE dédiées), non-réentrance maîtrisée, absence de fuite d'état par les registres au retour. Le compilateur assiste (nettoyage des registres, veneers), mais la validation sémantique des arguments reste à la charge du développeur — la classe de défauts BOOMERANG (section 6.3) se transpose intégralement, à une échelle de fréquence d'appel supérieure.

	TrustZone-A (Cortex-A)	TrustZone-M (Cortex-M)
Attribution de sécurité	État du cœur (bit NS de `SCR_EL3`)	Attribut de l'adresse (SAU + IDAU)
Bascule de monde	Exception vers le moniteur EL3 (`SMC`, interruptions)	Branchements contrôlés (`SG`, `BLXNS`/`BXNS`), sans logiciel intermédiaire
Coût de bascule	Centaines de cycles (sauvegarde de contexte logicielle)	Quelques cycles, latence déterministe
Traduction mémoire	MMU par monde, tables distinctes	Pas de MMU ; MPU par monde
Niveaux de privilège	EL0–EL3, Secure EL2 possible	Privilégié/non privilégié dans chaque monde
Pile logicielle type	TF-A (EL3) + TEE (OP-TEE…) + OS riche	TF-M (ou équivalent) + RTOS ou bare metal
Interruptions	GIC, groupes sécurisé/non sécurisé	NVIC, assignation par ligne (`ITNS`), priorités dépréciables (`PRIS`)
Partition	Dynamique (tables, TZASC reprogrammable)	Essentiellement statique, figée au démarrage

Table 7.1 – Comparaison des mécanismes TrustZone-A et TrustZone-M.

7.3 Mémoire et périphériques

La SAU et l'IDAU n'attribuent que les accès du cœur : comme sur Cortex-A, l'isolation doit être prolongée dans le SoC. Les microcontrôleurs TrustZone-M embarquent pour cela des contrôleurs de protection mémoire (MPC) qui partitionnent flash et SRAM par blocs assignables à l'un des mondes, et des contrôleurs de protection de périphériques (PPC) qui assignent chaque périphérique, le bus AHB5 propageant l'attribut de sécurité (`HNONSEC`). La cohérence entre la carte SAU/IDAU et la configuration MPC/PPC relève du logiciel sécurisé d'initialisation : une région déclarée sécurisée côté cœur mais laissée ouverte côté contrôleur est accessible à tout maître non sécurisé — l'erreur type du portage TrustZone-M.

Les implémentations varient sensiblement d'un fondeur à l'autre, jusque dans la granularité et le vocabulaire : GTZC (MPCBB par blocs de SRAM, TZSC, TZIC pour les événements illégaux) chez STMicroelectronics [34] ; *Secure AHB Controller* avec règles par maître et par esclave, et niveaux de sécurité par maître, chez NXP sur LPC55 [29] ; SPU unifiée (mémoire, périphériques, broches GPIO) chez Nordic [28] ; attribution de la flash par rangées et périmètre réduit sur le Cortex-M23 du SAM L11, dépourvu de SAU [26]. Les maîtres non-cœur — contrôleurs DMA en premier lieu, mais aussi radios et accélérateurs — posent la même question que sur Cortex-A : l'attribut de leurs transactions et la protection de leurs registres de programmation doivent être vérifiés maître par maître, la plupart des DMA de microcontrôleurs étant configurables par canal avec un attribut hérité du monde qui les programme. Les broches d'entrée-sortie, enfin, peuvent être assignées individuellement à un monde sur la plupart des plateformes, ce qui permet de réserver au monde sécurisé un bouton, une LED ou un bus vers un élément sécurisé externe. Le tableau 7.2 résume ces déclinaisons.

7.4 Sécurité, isolation et écosystème

L'usage nominal de TrustZone-M s'est stabilisé autour d'un partitionnement simple : le monde sécurisé héberge une racine de confiance d'exécution — démarrage sécurisé, stockage de clés, cryptographie, attestation, mise à jour — et le monde non sécurisé l'application, RTOS

Famille	Mécanismes d'isolation SoC	Points notables
STMicroelectronics STM32L5/U5 [34]	GTZC : MPCBB (SRAM par blocs), TZSC (périphériques), TZIC (événements illégaux)	Interruption dédiée aux violations ; partition de flash par *watermark*
NXP LPC55S6x [29]	Matrice AHB sécurisée, MPC/PPC, niveaux de sécurité par maître	Services en ROM étendus ; retour d'expérience CVE-2021-31532 (section 7.6)
Nordic nRF9160, nRF5340 [28]	SPU unifiée : flash, RAM, périphériques, broches GPIO	Modèle homogène ; assignation des broches individuelle
Microchip SAM L10/L11 [26]	Cortex-M23 sans SAU : partition IDAU fixée par fusibles	Granularité par rangées de flash ; périmètre réduit du cœur

TABLE 7.2 – Déclinaisons des mécanismes d'isolation TrustZone-M chez quatre fondeurs. Les noms et les granularités ne sont pas portables d'une famille à l'autre.

compris. L'implémentation de référence est Trusted Firmware-M [37], qui fournit ces services derrière les API PSA, avec trois niveaux d'isolation croissants (du simple cloisonnement des deux mondes jusqu'à l'isolation mutuelle des partitions de service via la MPU sécurisée). Le démarrage s'appuie classiquement sur une ROM ou un BL2 de type MCUboot, avec vérification de signature et compteurs anti-retour. Le cadre PSA Certified value l'ensemble par des niveaux d'assurance évalués par des laboratoires tiers [15], le niveau le plus répandu (PSA Certified Level 2) correspondant à une évaluation de la cible contre des attaques logicielles « échelonnables », explicitement pas contre les attaques physiques.

Cette architecture tire le meilleur du mécanisme : le TCB sécurisé reste petit (dizaines de milliers de lignes), les points d'entrée sont peu nombreux et typés, et l'application peut être compromise sans que les clés d'identité ni la capacité de mise à jour ne le soient — ce qui est précisément le scénario de rétablissement recherché sur un objet connecté. Les écarts à ce schéma — multiplier les services sécurisés, faire tourner des piles protocolaires dans le monde sécurisé — reproduisent l'inflation de TCB observée sur les TEE Cortex-A, sans les protections mémoire d'une MMU.

7.5 Pièges de développement et de portage

La frontière TrustZone-M étant une frontière d'appel de fonction, sa robustesse dépend de détails de développement qu'une évaluation de firmware sécurisé doit contrôler explicitement. Le premier groupe concerne les entrées : toute fonction exposée (attribut `cmse_nonsecure_entry`) doit valider que les pointeurs reçus désignent de la mémoire accessible au monde non sécurisé avec les droits de l'appelant — c'est l'objet des routines CMSE de vérification d'adresses, bâties sur l'instruction `TT` — et copier les données avant de les valider, la mémoire partagée restant modifiable par l'appelant pendant le traitement. Le deuxième groupe concerne les sorties : les appels du monde sécurisé vers du code non sécurisé (pointeurs de fonction marqués `cmse_nonsecure_call`, mécanismes de rappel) doivent passer par les séquences générées par le compilateur, qui masquent les registres et empruntent `BLXNS`. Tout appel direct construit à la main est suspect. Les cibles de ces rappels doivent être traitées comme des valeurs hostiles — la classe Ret2NS (section 7.6) sanctionne précisément les validations manquantes à ce point.

Le troisième groupe relève de la configuration : compiler l'ensemble du monde sécurisé avec l'option CMSE (`-mcmse`) et les contournements d'errata applicables ; sceller les piles sécurisées ; verrouiller la SAU et les contrôleurs après initialisation ; installer un gestionnaire de `SecureFault` qui journalise et traite les violations plutôt que de les masquer ; et interdire au monde non sécurisé

de modifier les ressources dont dépend le monde sécurisé (horloges des périphériques sécurisés, alimentation des banques de SRAM). Ces points sont documentés par Arm et appliqués par TF-M [4, 37] ; leur vérification sur un monde sécurisé développé sur mesure est une étape obligée de revue, la plupart se prêtant à un contrôle outillé sur le binaire (liste des veneers, options de compilation, valeurs de scellement).

7.6 Limites et retour d'expérience des attaques

TrustZone-M est plus jeune que son aînée, mais le retour d'expérience publié est déjà substantiel, et il dessine trois familles de faiblesses : les défauts de l'architecture elle-même au niveau de ses transitions, les canaux microarchitecturaux qui subsistent malgré la simplicité des cœurs, et les défauts d'implémentation des plateformes.

7.6.1 Défauts aux frontières : scellement de pile et VLLDM

Deux avis de sécurité d'Arm illustrent la sensibilité des transitions. Le premier (CVE-2020-16273) concerne le scellement des piles sécurisées : le pointeur de pile sécurisé pouvant, dans certaines séquences, être influencé par des valeurs issues du monde non sécurisé, un logiciel sécurisé qui ne place pas de valeur de scellement au sommet de ses piles peut être amené à dépiler des données contrôlées par l'attaquant [6]. La contre-mesure — sceller chaque pile sécurisée avec une valeur réservée — est logicielle, documentée, et intégrée aux implémentations de référence. Son absence dans un firmware sécurisé développé sur mesure est un défaut exploitable. Le second (CVE-2021-35465) touche le mécanisme de sauvegarde paresseuse du contexte flottant : une séquence impliquant l'instruction `VLLDM` permettait, sur certaines révisions de Cortex-M33, M35P et M55, au monde non sécurisé de lire des registres flottants utilisés par le monde sécurisé [13]. Correction par révision du silicium et contournement par le compilateur : l'évaluateur doit vérifier la révision du cœur et les options de compilation du firmware sécurisé. Dans les deux cas, la leçon est la même : la frontière matérielle « en quelques cycles » repose sur des conventions fines dont le respect incombe au logiciel et à sa chaîne de compilation.

La recherche académique a généralisé le propos : l'étude Ret2NS a montré que les retours du monde sécurisé vers le monde non sécurisé (`BXNS`, `BLXNS`) insuffisamment validés permettent de détourner le flot d'exécution non sécurisé vers des cibles choisies — notamment d'élever ses privilèges dans le monde non sécurisé en faisant « atterrir » le retour en contexte privilégié — et a proposé des assainissements d'adresses à coût faible [23]. Là encore, le matériel vérifie la légalité de la transition, pas sa cohérence sémantique.

7.6.2 Canaux microarchitecturaux sans caches : BUSted

L'argument selon lequel les microcontrôleurs, dépourvus de caches et d'exécution spéculative, seraient à l'abri des canaux auxiliaires microarchitecturaux ne tient plus : l'attaque BUSted (IEEE S&P 2024) a démontré un canal temporel exploitant la contention de l'interconnect AHB des microcontrôleurs, mesurable depuis le monde non sécurisé et permettant d'inférer les motifs d'accès du monde sécurisé, sur des cibles TrustZone-M réelles [31]. La bande passante du canal est inférieure à celle d'un canal de cache, mais la conclusion d'évaluation est identique à celle du

chapitre 6 : le code sécurisé manipulant des secrets doit être à temps constant, y compris sur Cortex-M, et la revue de ce point fait partie de l'évaluation du firmware sécurisé.

7.6.3 Défauts de plateforme : le cas LPC55S69

Le microcontrôleur NXP LPC55S69 (Cortex-M33, TrustZone-M) a fourni le cas d'école des défauts d'implémentation. En 2021, l'analyse de sa ROM par Oxide Computer a révélé un bloc matériel non documenté de correction de ROM (*ROM patcher*), accessible depuis le monde non sécurisé et non privilégié, permettant de modifier à l'exécution le comportement des API de la ROM — y compris celles sur lesquelles reposent le démarrage sécurisé et la configuration TrustZone (CVE-2021-31532) [1]. Aucune faiblesse de l'architecture Armv8-M n'est en cause : le périmètre sécurisé du SoC omettait un esclave critique, non documenté de surcroît. Le cas concentre plusieurs enseignements de ce guide : la sécurité effective est celle de l'intégration (section 5.1). Les blocs non documentés font partie de la surface d'attaque et l'opacité d'une ROM propriétaire empêche l'audit. Une vulnérabilité de ROM n'est pas corrigeable sur le silicium déployé.

7.6.4 Attaques physiques

Les microcontrôleurs TrustZone-M sont déployés dans des objets physiquement accessibles, face à des attaquants dont l'outillage d'injection de fautes (glitch de tension ou d'horloge, impulsions électromagnétiques) est devenu abordable. La configuration de l'isolation — registres SAU, MPC, PPC, étapes de vérification du démarrage sécurisé — constitue une cible naturelle de ces techniques : faire échouer une instruction de comparaison ou une écriture de configuration suffit dans certains scénarios à dégrader le partitionnement, et la littérature comme les évaluations industrielles documentent régulièrement des contournements de démarrage sécurisé par injection de fautes sur microcontrôleurs, toutes architectures confondues. Le niveau de résistance est propre à chaque composant : présence de détecteurs de glitch, redondance des tests critiques dans la ROM, lecture multiple des fusibles de configuration. Ces informations sont rarement publiques. Les certifications visant les attaques matérielles (PSA Certified Level 3, évaluations de type AVA_VAN élevé) sont alors le seul indicateur disponible, et leur absence doit être interprétée comme une absence de garantie. TrustZone-M, pas plus que TrustZone-A, n'a été conçue comme une protection contre l'attaquant physique.

7.6.5 Étude de cas : évaluer un produit sur microcontrôleur TrustZone-M

Pour fixer la méthode, considérons un produit type : un objet connecté bâti sur un microcontrôleur Cortex-M33, dont le monde sécurisé (TF-M) détient une clé d'identité et assure la mise à jour signée, le monde non sécurisé exécutant l'application et la pile réseau. Le modèle de menace retient un attaquant distant capable de compromettre l'application, et un attaquant local sans outillage de laboratoire. L'évaluation suit alors les questions suivantes, dans l'ordre :

1. **Racine de confiance.** La ROM et la configuration de démarrage sont-elles documentées et exemptes de CVE connues (le précédent LPC55S69 justifie la question) ? Les clés publiques et compteurs anti-retour sont-ils en fusibles, verrouillés en production ?
2. **Partitionnement.** La carte mémoire SAU/IDAU et la configuration MPC/PPC sont-elles cohérentes, verrouillées après initialisation, et les violations remontent-elles une interruption sécurisée traitée ?
3. **Maîtres.** Pour chaque DMA et coprocesseur : quel attribut émet-il, qui peut le programmer, qu'atteint-il ? La réponse doit être documentée, pas supposée.

4. **Frontière d'appel.** Les fonctions d'entrée sécurisées valident-elles leurs pointeurs (routines CMSE), les piles sécurisées sont-elles scellées, la révision du cœur et les options de compilation couvrent-elles les CVE d'Arm ?
5. **Secrets.** Les opérations cryptographiques du monde sécurisé sont-elles à temps constant (BUSted rend la question pertinente même sans cache) ? Les clés sont-elles liées au matériel ?
6. **Debug et cycle de vie.** Le debug est-il désactivé ou authentifié en production, pour les deux mondes ? La réactivation est-elle destructive pour les secrets ?
7. **Résistance physique.** Si le modèle de menace inclut l'attaquant outillé : quelles contre-mesures d'injection de fautes le composant documente-t-il, quelle certification l'atteste ? À défaut, les secrets de forte valeur doivent migrer vers un élément sécurisé.

Si les points 1 à 6 reçoivent des réponses positives et vérifiables, TrustZone-M porte correctement le scénario « application compromise, identité et mise à jour préservées », qui est son cas d'usage de prédilection. Le point 7 délimite ce qu'elle ne portera pas seule.

8

Conclusion

8.1 Ce que TrustZone garantit, et à quelles conditions

Au terme de cette analyse, le périmètre de TrustZone peut être énoncé précisément. La technologie fournit, à un coût matériel marginal, une isolation de contrôle d'accès entre deux piles logicielles complètes partageant les mêmes cœurs : un monde normal compromis, y compris au niveau de son noyau, ne peut ni lire ni modifier les ressources du monde sécurisé *tant que* quatre conditions sont réunies. Premièrement, une chaîne de démarrage sécurisé ancrée dans une ROM et des fusibles sains amène la plateforme jusqu'à une configuration d'isolation verrouillée (sections 3.3 et 5.7). Deuxièmement, l'intégration SoC est complète : chaque mémoire, chaque périphérique et chaque maître de bus est couvert par un filtre dont la configuration est connue et verrouillée, les maîtres programmables étant contraints par SMMU, pare-feux ou contrôleurs de domaines (chapitre 5). Troisièmement, le logiciel du monde sécurisé est minimal, validé à ses frontières et à temps constant sur ses secrets, car il concentre l'essentiel du risque résiduel démontré (chapitre 6). Quatrièmement, le modèle de menace exclut — ou couvre par d'autres moyens — l'attaquant microarchitectural outillé et l'attaquant physique, contre lesquels TrustZone n'a jamais prétendu protéger.

Aucune de ces conditions n'est exotique, mais aucune n'est acquise d'office : chacune correspond à des défaillances publiées, sur des produits commerciaux, parfois emblématiques. C'est la raison d'être de ce guide : la question utile n'est pas « la plateforme a-t-elle TrustZone ? » mais « la plateforme réunit-elle les conditions sous lesquelles TrustZone tient ses promesses ? ».

8.2 Suffisante, à compléter, ou insuffisante

La décision finale dépend du modèle de menace et de la valeur des actifs. Trois situations types structurent l'arbitrage.

TrustZone (ou TrustZone-M) est **suffisante** lorsque l'attaquant dominant est logiciel et distant, que les actifs à protéger sont l'identité de l'appareil, la capacité de mise à jour et des clés dont la compromission reste d'impact borné, et que la plateforme satisfait aux quatre conditions ci-dessus. Le scénario de rétablissement — application compromise, racine d'exécution intacte,

correctif déployé par mise à jour signée — est le service rendu caractéristique, sur Cortex-A comme sur Cortex-M.

TrustZone est **à compléter** lorsque certains actifs justifient une protection contre les canaux auxiliaires ou l'attaquant physique modérément outillé : clés de forte valeur ou à longue durée de vie, secrets partagés de flotte, obligations de certification. Le complément prend la forme d'un élément sécurisé ou d'un HSE pour les opérations sur ces clés, TrustZone conservant le rôle d'isolation d'exécution et d'orchestration — architecture hybride qui est aujourd'hui l'état de l'art des produits exigeants, et que le guide H2Lab de sélection des composants de sécurité développe.

TrustZone est **insuffisante seule** — et sa présence ne doit pas peser dans la décision — face à un attaquant physique outillé disposant du temps et de l'accès au composant, pour des secrets dont la divulgation est critique à l'échelle d'une flotte, ou sur une plateforme dont l'intégration ne peut être ni documentée ni vérifiée. Dans ce dernier cas, le défaut n'est pas dans la technologie mais dans l'impossibilité de l'évaluer : une isolation invérifiable ne vaut pas mieux, du point de vue de l'assurance, qu'une absence d'isolation.

8.3 Grille d'évaluation

Liste de contrôle d'évaluation d'une plateforme TrustZone



1. Chaîne de démarrage : ROM sans CVE connue, signatures vérifiées à chaque étage, anti-retour en fusibles, fenêtre de reset maîtrisée.
2. Documentation : cartographie de sécurité du SoC disponible et suffisante pour tracer la protection de chaque ressource.
3. Pare-feux mémoire : couverture de tous les ports d'accès, granularité adaptée, configuration verrouillée, violations détectées et traitées.
4. Maîtres de bus : attribut émis, registres de programmation protégés, SMMU ou pare-feux par maître, micrologiciels des coprocesseurs dans le périmètre.
5. MPSoC : contrôleurs de domaines couvrant tous les maîtres, matrice maîtres × ressources vérifiée sur les deux dimensions (domaine et attribut).
6. Gestion d'énergie : DVFS et contrôle des cœurs confinés au monde sécurisé (PSCI) ou bornés matériellement.
7. Debug : accès sécurisé désactivé ou authentifié en production, chemins latéraux (debug inter-cœurs, trace) inclus dans l'analyse.
8. Logiciel sécurisé : TCB minimal et à jour, validation des paramètres aux frontières, cryptographie à temps constant, mécanismes anti-dégradation.
9. Spécifique TrustZone-M : cohérence SAU/IDAU/MPC/PPC, piles scellées, révisions de cœur et options de compilation couvrant les CVE publiées.
10. Adéquation au modèle de menace : actifs de forte valeur confiés à un composant résistant aux attaques physiques si le modèle l'exige.

8.4 Perspectives

Les évolutions en cours ne remettent pas en cause cette grille, elles la prolongent. Sur les processeurs d'application, Secure EL2 et l'architecture CCA d'Armv9 [8] répondent à la concentration de privilèges du monde sécurisé en fragmentant celui-ci et en ajoutant des domaines contrôlés par le matériel. Les plateformes qui en bénéficient restent soumises aux mêmes exigences d'intégration et de démarrage. Sur les microcontrôleurs, la consolidation autour de TF-M et de PSA Certified élève progressivement le plancher de qualité des mondes sécurisés, tandis que la recherche — de BUSted à Ret2NS — continue de documenter les angles morts du mécanisme, au bénéfice des évaluations futures. La constante, elle, ne changera pas : TrustZone est un outil d'architecture dont la valeur se décide à l'intégration, se vérifie à l'évaluation, et se perd dans l'implicite.

Bibliographie

[1] Laura Abbott and Rick Altherr. Exploiting undocumented hardware blocks in the LPC55S69. Oxide Computer Company, https://oxide.computer/blog/exploiting-undocumented-hardware-blocks-in-the-lpc55s69, 2021. CVE-2021-31532.

[2] ARM Limited. *ARM Security Technology : Building a Secure System using TrustZone Technology*. ARM Limited, 2009. PRD29-GENC-009492C.

[3] Arm Limited. *ARM CoreLink TZC-400 TrustZone Address Space Controller : Technical Reference Manual*. Arm Limited, 2013. DDI 0504.

[4] Arm Limited. *TrustZone Technology for the Armv8-M Architecture*. Arm Limited, 2017. Document 100690, version 2.1.

[5] Arm Limited. *Trusted Board Boot Requirements CLIENT (TBBR-CLIENT) Armv8-A*. Arm Limited, 2018. DEN 0006.

[6] Arm Limited. Armv8-M stack sealing vulnerability (CVE-2020-16273). Arm Security Center, avis de sécurité, 2020.

[7] Arm Limited. *AMBA AXI and ACE Protocol Specification*. Arm Limited, 2021. IHI 0022.

[8] Arm Limited. *Arm Confidential Compute Architecture (CCA) Security Model*. Arm Limited, 2021. DEN 0096.

[9] Arm Limited. *Arm Power State Coordination Interface (PSCI)*. Arm Limited, 2021. DEN 0022.

[10] Arm Limited. *Arm System Memory Management Unit Architecture Specification, SMMU architecture version 3*. Arm Limited, 2021. IHI 0070.

[11] Arm Limited. *Armv8-M Architecture Reference Manual*. Arm Limited, 2021. DDI 0553.

[12] Arm Limited. *SMC Calling Convention*. Arm Limited, 2021. DEN 0028.

[13] Arm Limited. VLLDM instruction security vulnerability (CVE-2021-35465). Arm Security Center, avis de sécurité, 2021.

[14] Arm Limited. *Arm Architecture Reference Manual for A-profile architecture*. Arm Limited, 2023. DDI 0487.

[15] Arm Limited and PSA Certified. PSA Certified : Security framework and certification scheme. https://www.psacertified.org/, 2021.

[16] El Mehdi Benhani, Lilian Bossuet, and Alain Aubert. The security of ARM TrustZone in a FPGA-based SoC. *IEEE Transactions on Computers*, 68(8) :1238–1248, 2019.

[17] David Cerdeira, Nuno Santos, Pedro Fonseca, and Sandro Pinto. SoK : Understanding the prevailing security vulnerabilities in TrustZone-assisted TEE systems. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2020.

[18] GlobalPlatform. *TEE System Architecture*. GlobalPlatform, 2022. GPD_SPE_009.

[19] Lee Harrison, Hayawardh Vijayakumar, Rohan Padhye, Koushik Sen, and Michael Grace. PARTEMU : Enabling dynamic analysis of real-world TrustZone software using emulation. In *Proceedings of the 29th USENIX Security Symposium*, 2020.

[20] Paul Kocher, Jann Horn, Anders Fogh, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. Spectre attacks : Exploiting speculative execution. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2019.

[21] Moritz Lipp, Daniel Gruss, Raphael Spreitzer, Clémentine Maurice, and Stefan Mangard. ARMageddon : Cache attacks on mobile devices. In *Proceedings of the 25th USENIX Security Symposium*, 2016.

[22] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Anders Fogh, Jann Horn, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, and Mike Hamburg. Meltdown : Reading kernel memory from user space. In *Proceedings of the 27th USENIX Security Symposium*, 2018.

[23] Zheyuan Ma, Xi Tan, Lukasz Ziarek, Ning Zhang, Hongxin Hu, and Ziming Zhao. Return-to-non-secure vulnerabilities on ARM Cortex-M TrustZone : Attack and defense. In *Proceedings of the 60th ACM/IEEE Design Automation Conference (DAC)*, 2023.

[24] Aravind Machiry, Eric Gustafson, Chad Spensky, Christopher Salls, Nick Stephens, Ruoyu Wang, Antonio Bianchi, Yung Ryn Choe, Christopher Kruegel, and Giovanni Vigna. BOOMERANG : Exploiting the semantic gap in trusted execution environments. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2017.

[25] A. Theodore Markettos, Colin Rothwell, Brett F. Gutstein, Allison Pearce, Peter G. Neumann, Simon W. Moore, and Robert N. M. Watson. Thunderclap : Exploring vulnerabilities in operating system IOMMU protection via DMA from untrustworthy peripherals. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2019.

[26] Microchip Technology. *SAM L10/L11 Family Data Sheet*. Microchip Technology, 2020. DS60001513.

[27] Zhenyu Ning and Fengwei Zhang. Understanding the security of ARM debugging features. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2019.

[28] Nordic Semiconductor. *nRF9160 Product Specification*. Nordic Semiconductor, 2021.

[29] NXP Semiconductors. *UM11126 : LPC55S6x/LPC55S2x/LPC552x User Manual*. NXP Semiconductors, 2021.

[30] Pengfei Qiu, Dongsheng Wang, Yongqiang Lyu, and Gang Qu. VoltJockey : Breaching TrustZone by software-controlled voltage manipulation over multi-core frequencies. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2019.

[31] Cristiano Rodrigues, Daniel Oliveira, and Sandro Pinto. BUSted!!! microarchitectural side-channel attacks on the MCU bus interconnect. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2024.

[32] Keegan Ryan. Hardware-backed heist : Extracting ECDSA keys from Qualcomm's TrustZone. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2019.

[33] Alon Shakevsky, Eyal Ronen, and Avishai Wool. Trust dies in darkness : Shedding light on Samsung's TrustZone keymaster design. In *Proceedings of the 31st USENIX Security Symposium*, 2022.

[34] STMicroelectronics. *RM0438 : STM32L552xx and STM32L562xx Advanced Arm-based 32-bit MCUs Reference Manual*. STMicroelectronics, 2021.

[35] Adrian Tang, Simha Sethumadhavan, and Salvatore Stolfo. CLKSCREW : Exposing the perils of security-oblivious energy management. In *Proceedings of the 26th USENIX Security Symposium*, 2017.

[36] TrustedFirmware.org. Trusted Firmware-A documentation. https://trustedfirmware-a.readthedocs.io/, 2024.

[37] TrustedFirmware.org. Trusted Firmware-M documentation. https://trustedfirmware-m.readthedocs.io/, 2024.

[38] Victor van der Veen, Yanick Fratantonio, Martina Lindorfer, Daniel Gruss, Clémentine Maurice, Giovanni Vigna, Herbert Bos, Kaveh Razavi, and Cristiano Giuffrida. Drammer : Deterministic rowhammer attacks on mobile platforms. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2016.

[39] Ning Zhang, Kun Sun, Deborah Shands, Wenjing Lou, and Y. Thomas Hou. TruSpy : Cache side-channel information leakage from the secure world on ARM devices. Cryptology ePrint Archive, Report 2016/980, 2016.

Licence

Ce document est diffuse sous licence **Creative Commons CC BY-NC-ND 4.0** (Attribution – Pas d'Utilisation Commerciale – Pas de Modification).
Texte integral de la licence : https://creativecommons.org/licenses/by-nc-nd/4.0/deed.fr

