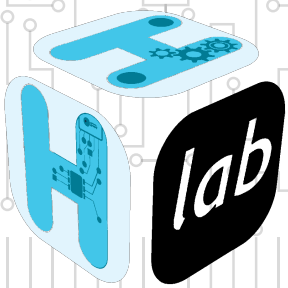
H2LAB-STIG-JTAG
07-2026

Systèmes embarqués : l'interface JTAG



STIG H²Lab

CC BY NC SA

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 bonnespratiques en matière de sécurité, de confidentialité et de durabilité dans l'écosystèmedes 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	Juillet 2026	H2Lab	Version initiale du guide

Table des matières

1
Glossaire et acronymes

1.1 Glossaire

Ancre de confiance. Donnée immuable (clef publique, hash de firmware, certificat racine) utilisée pour établir une relation de confiance initiale.

Anti-probing. Mesures matérielles et de design visant à compliquer l'identification et l'instrumentation des points de test sensibles.

Anti-rollback. Mécanisme empêchant le retour à une version antérieure valide cryptographiquement mais vulnérable.

Brown-out. Chute transitoire de tension d'alimentation pouvant provoquer des erreurs exploitables lors d'attaques par faute.

Chaîne d'approvisionnement. Ensemble des acteurs, outils et opérations intervenant entre la conception d'un produit et sa mise en service.

Challenge-réponse. Mécanisme d'authentification où la cible vérifie la possession d'un secret à partir d'un défi cryptographique.

Debug en production. Capacité d'analyse ou de reprogrammation conservée sur un produit livré, en général pour la maintenance.

Debug d'exception. Accès debug temporaire, strictement encadré et traçable, activé uniquement sur justification opérationnelle.

Deny by default. Principe de sécurité où tout accès est refusé par défaut, sauf autorisation explicite.

Fenêtre forensique. Ensemble des traces exploitables pour reconstituer un événement de sécurité après incident.

Fuse OTP. Bit de configuration programmable une seule fois (One-Time Programmable), souvent utilisé pour verrouiller des fonctions sensibles.

Lifecycle. États de cycle de vie d'un composant (développement, production, service, fin de vie) conditionnant les privilèges de sécurité.

Lock bits. Bits de configuration activant des restrictions d'accès (lecture, écriture, debug) selon l'état du composant.

Mode service. Mode opératoire permettant un débogage contrôlé, temporaire et traçable sur un équipement en exploitation.

Racine de confiance matérielle. Élément matériel assurant le stockage et l'usage sécurisé d'identifiants cryptographiques.

Supply-chain. Terme anglais couramment utilisé pour désigner la chaîne d'approvisionnement et ses risques de compromission.

Surface d'attaque. Ensemble des points d'entrée exploitables par un attaquant sur un système donné.

Traçabilité. Capacité à reconstituer qui a fait quoi, quand, comment et avec quel niveau d'autorisation.

1.2 Liste des acronymes

AM. Profil d'attaquant formalisé dans ce guide (AM-01 à AM-05).

ANSSI. Agence nationale de la sécurité des systèmes d'information.

ARM. *Advanced RISC Machines.*

BKPT. Instruction de point d'arrêt logiciel (*BreaK PoinT*).

BOM. *Bill Of Materials.*

CAP. Capacité technique associée à un profil d'attaquant (CAP-01 à CAP-08).

CPU. *Central Processing Unit.*

CSIRT. *Computer Security Incident Response Team.*

DFU. *Device Firmware Update.*

EM. *Electromagnetic* (électromagnétique).

EMFI. *ElectroMagnetic Fault Injection.*

ENISA. *European Union Agency for Cybersecurity.*

ESD. *ElectroStatic Discharge.*

ETM. *Embedded Trace Macrocell.*

FDA. *Food and Drug Administration.*

FPGA. *Field-Programmable Gate Array.*

FS. Fonction de sécurité requise dans ce guide (FS-01 à FS-09).

HSM. *Hardware Security Module.*

HW. *Hardware.*

IACS. *Industrial Automation and Control Systems.*

IEEE. *Institute of Electrical and Electronics Engineers.*

IEC. *International Electrotechnical Commission.*

ISO. *International Organization for Standardization.*

IoT. *Internet of Things.*

IR/DR. *Instruction Register / Data Register.*

ITM. *Instrumentation Trace Macrocell.*

JTAG. *Joint Test Action Group.*

MCU. *Microcontroller Unit.*

MDR. *Medical Device Regulation.*

NIST. *National Institute of Standards and Technology.*

NISTIR. *NIST Interagency/Internal Report.*

OEM. *Original Equipment Manufacturer.*

OS. Objectif de sécurité dans ce guide (OS-01 à OS-08).

OT. *Operational Technology.*

OTP. *One-Time Programmable.*

PMP. *Physical Memory Protection.*

PCB. *Printed Circuit Board.*

RAM. *Random Access Memory.*

RDP. *Readout Protection.*

RMA. *Return Merchandise Authorization.*

SE. *Secure Element.*

SIEM. *Security Information and Event Management.*

SL. *Security Level* (IEC 62443).

SOC. *Security Operations Center.*

SP. *Special Publication* (NIST SP).

SRAM. *Static Random Access Memory.*

SSI. Sécurité des systèmes d'information.

STIG. *Security Technical Implementation Guide.*

SWD. *Serial Wire Debug.*

SWO. *Serial Wire Output.*

SVC. *Supervisor Call.*

TAP. *Test Access Port.*

TCK/TMS/TDI/TDO. Signaux standards de l'interface JTAG.

TEE. *Trusted Execution Environment.*

TRST. Signal optionnel de reset TAP (*Test ReSeT*).

TPM. *Trusted Platform Module.*

TTY. *Teletype* (canal terminal de type console).

TVS. *Transient Voltage Suppressor.*

UART. *Universal Asynchronous Receiver-Transmitter.*

USB. *Universal Serial Bus.*

2
Introduction et périmètre

2.1 Objectif du guide

Ce document constitue un **STIG** (*Security Technical Implementation Guide*) dédié à la sécurisation des interfaces de débogage JTAG/SWD dans les systèmes embarqués critiques. L'objectif est de fournir un cadre d'implémentation qui soit :

- techniquement actionnable (exigences concrètes et testables)
- opérable en contexte industriel OT (production, maintenance, incident)
- vérifiable en audit (preuves, traçabilité, critères de conformité)

Le guide couvre le cycle de vie complet : conception, intégration, production, déploiement et maintenance. Il traite explicitement la tension entre maintenabilité et robustesse cyber, selon une logique de défense en profondeur et de traçabilité opérationnelle [9, 10, 18, 1].

2.2 Public visé

Ce guide est conçu pour une lecture à deux niveaux : niveau stratégique (architecture, gouvernance du risque, conformité) et niveau implémentation (réglages concrets, validation technique, preuves d'audit). Cette double lecture facilite l'alignement entre décisions d'ingénierie et contraintes métier.

Le document s'adresse aux profils suivants :

- Architectes cybersécurité embarquée (systèmes OT, IoT industriel, automobile, médical)
- Ingénieurs hardware/firmware responsables des choix MCU/SoC, boot et provisioning
- Développeurs de produits de sécurité critiques amenés à intégrer des mécanismes de debug contrôlé
- Responsables SSI, auditeurs techniques et équipes qualité en charge des preuves de conformité

Les recommandations sont formulées pour être utilisées à la fois comme base de conception et comme référentiel d'audit interne ou tiers.

2.3 Périmètre technique et hypothèses

2.3.1 Type d'équipement et contexte OT

Le périmètre couvre des équipements embarqués connectés ou semi-connectés, reposant sur des MCU/SoC/FPGA avec interfaces de debug de type JTAG/SWD. Les cas d'usage visés incluent la mise en service usine, le diagnostic terrain et la maintenance corrective dans des environnements OT à contraintes fortes de disponibilité.

2.3.2 Cycle de vie considéré

Le document traite la sécurisation du debug sur l'ensemble du cycle de vie :

- conception de la carte et de l'architecture boot/debug
- industrialisation (provisioning, fuses/lock bits, contrôle qualité)
- déploiement en exploitation (supervision, contrôle d'accès)
- maintenance (ouverture temporaire, fermeture et preuve post-intervention)

2.3.3 Hypothèses de menace

Les scénarios considérés incluent un attaquant local avec accès physique opportuniste, un attaquant interne ou un sous-traitant malveillant, ainsi qu'un attaquant outillé capable de combiner accès debug, extraction firmware et fautes transitoires. L'hypothèse défensive centrale est qu'un accès physique partiel est plausible en maintenance, et doit donc être encadré plutôt que supposé impossible.

2.3.4 Lecture opérationnelle du STIG

Le présent document peut être utilisé selon trois modes complémentaires :

- **Mode architecture** : définir les principes de sécurisation dès la conception
- **Mode industrialisation** : traduire les principes en paramètres usine et tests par lot
- **Mode audit** : vérifier la conformité effective et la résilience opérationnelle

Cette lecture en trois temps facilite le dialogue entre équipes hardware, firmware, cybersécurité et opérations OT, et réduit l'écart entre exigence théorique et applicabilité terrain.

3

Fondamentaux techniques du JTAG

3.1 Origine et standardisation

JTAG est historiquement né pour les tests d'interconnexion de cartes électroniques (*boundary scan*) avant de devenir un canal privilégié de débogage et de programmation. La normalisation IEEE 1149.1 définit le modèle de chaîne, les états du contrôleur TAP et les registres de commande/données [11].

Dans l'industrie, les usages légitimes du debug bas niveau sont multiples :

- Test en fin de ligne de production
- Bring-up matériel et validation firmware
- Diagnostic de défauts terrain et reprise de produits
- Programmation initiale ou reprogrammation contrôlée

En pratique, la même interface qui réduit fortement les temps de qualification peut aussi réduire le coût d'une attaque si elle reste ouverte en production. La logique STIG consiste donc à préserver la valeur opérationnelle du debug, tout en le transformant en capacité d'exception et non en capacité permanente.

Le problème de sécurité n'est donc pas l'existence de JTAG en soi, mais son niveau d'exposition hors du contexte maîtrisé de développement et de production.

3.2 Architecture physique et logique

L'architecture JTAG repose sur le TAP, piloté par les signaux TCK, TMS, TDI, TDO (et parfois TRST). Le TAP orchestre des transitions d'états permettant de charger des instructions (IR) puis d'échanger des données (DR). Cette mécanique donne un accès potentiellement profond à l'état interne du composant.

D'un point de vue sécurité, trois propriétés sont déterminantes :

- **Universalité d'accès** : lecture/écriture registres et mémoire selon l'implémentation
- **Précision** : contrôle fin de l'exécution (halt/step/breakpoint)

- **Bypass partiel** : selon les MCU, certaines politiques logicielles peuvent être court-circuitées par le plan debug matériel

Exemple concret de variabilité : sur des familles STM32 configurées en RDP niveau 0, la lecture mémoire via l'interface debug peut rester possible en phase de développement, alors qu'en configuration de verrouillage fort (ex : RDP niveau élevé), ce même accès est fortement restreint, voire bloqué. À l'inverse, sur certaines plateformes NXP disposant d'un mécanisme de *secure debug mailbox*, l'accès peut être réouvert de manière temporaire après authentification explicite, ce qui n'est pas équivalent à un debug libre permanent [26, 20].

3.3 Variantes et évolutions

Dans l'écosystème ARM, SWD remplace souvent JTAG sur les microcontrôleurs pour réduire le nombre de broches, tout en conservant des capacités de débogage élevées. D'autres extensions existent : trace embarquée (ETM), semihosting, interfaces propriétaires de maintenance [2].

Semihosting. Le semihosting est un mécanisme par lequel le firmware cible délègue certaines opérations d'entrée/sortie à l'hôte de debug (PC outillage), via un point d'arrêt logiciel (souvent instruction de type *BKPT/SVC* selon architecture). Concrètement, un appel applicatif comme *printf* peut être redirigé vers la console du débogueur, sans pile réseau ni pilote UART côté cible. En phase de développement, cela accélère fortement l'observabilité. En production, en revanche, un semihosting laissé actif peut créer un canal de fuite d'information ou de perturbation d'exécution si un outil de debug non autorisé est connecté.

ETM (Embedded Trace Macrocell). ETM est un bloc de trace matérielle qui exporte des informations fines d'exécution (notamment flux d'instructions, branches, événements) avec une surcharge CPU limitée par rapport à une instrumentation logicielle classique. Cette capacité est précieuse pour l'analyse de bugs temporels et l'investigation post-mortem. Du point de vue sécurité, la trace peut cependant révéler des séquences sensibles (chemins d'authentification, états cryptographiques, logique de contrôle). Un STIG robuste doit donc encadrer son activation, son périmètre et la conservation des traces.

TTY sur port JTAG/SWD. La notion de TTY désigne un canal terminal de type console (entrée/sortie texte) exposé via l'infrastructure de debug, souvent au travers de mécanismes comme semihosting, SWO/ITM ou passerelles outillage vers une pseudo-console hôte. En pratique, cela revient à obtenir une "console série virtuelle" sans UART physique dédié. L'avantage est opérationnel (diagnostic rapide), mais le risque est de réintroduire un canal interactif puissant sur des équipements censés être verrouillés. En production, ce canal doit être explicitement désactivé ou strictement conditionné (authentification, durée limitée, journalisation complète).

Les évolutions majeures à considérer dans un STIG sont :

- Multiplication des canaux de debug (JTAG, SWD, UART de secours, USB DFU)
- Chaînage de plusieurs composants (daisy-chain), avec impact sur l'isolation des domaines de confiance
- Fonctions de debug avancées pouvant exposer des données sensibles en clair (clefs, secrets de session, états d'authentification)

Du point de vue défensif, ces évolutions imposent une approche par surface agrégée : il faut sécuriser l'ensemble des voies de maintenance, et pas uniquement le connecteur JTAG principal, faute de quoi l'attaquant contourne la mesure la plus visible via un canal auxiliaire moins protégé.

4

Menaces et vecteurs d'attaque via JTAG

La menace JTAG doit être comprise comme une menace de *chaîne opérationnelle*. Dans les incidents observés, l'attaque aboutit rarement grâce à une unique faiblesse technique. Elle réussit plutôt par enchaînement : accès physique plausible, contrôle insuffisant du mode maintenance, paramètres de sécurité hétérogènes entre lots, et supervision incomplète. Cette perspective est essentielle pour éviter une réponse purement "bit de protection".

4.1 Intérêt pour un attaquant

Un accès JTAG/SWD non maîtrisé transforme souvent une attaque complexe en opération d'extraction et de modification directe :

- Dump de firmware depuis flash interne/externe
- Lecture/écriture arbitraire de SRAM, périphériques et registres de sécurité
- Injection de code et contournement de mécanismes de contrôle d'intégrité
- Récupération de secrets temporaires (clefs dérivées, tokens, mots de passe en mémoire)

Concrètement, ces actions donnent à l'attaquant une visibilité et un contrôle de très bas niveau qui court-circuitent la plupart des mécanismes applicatifs. Une authentification robuste côté application ne protège pas un secret déjà exposé en mémoire vive lors d'une session debug non autorisée.

L'intérêt stratégique est élevé : vol de propriété intellectuelle, clonage de produits, sabotage ciblé, compromission de chaînes de maintenance et d'approvisionnement. Les analyses de terrain montrent que les attaques réussies exploitent fréquemment un cumul de faiblesses (debug actif, logs insuffisants, secrets statiques) plutôt qu'une faille unique [22, 16].

4.2 Surface d'attaque

Contrairement à une idée reçue, la contrainte d'accès physique n'est pas une garantie suffisante en environnement industriel. Les scénarios réalistes incluent :

- Intervention malveillante en atelier de maintenance

- Compromission d'un sous-traitant de réparation
- Attaque opportuniste sur équipement exposé (armoire, véhicule, borne)
- Accès via pads de test non documentés ou connecteurs internes

Dans les environnements OT, la fenêtre d'opportunité est souvent liée au cycle de maintenance : ouverture de châssis, remplacement de module, accès temporaire aux cartes. Ce sont précisément ces moments qui doivent être protégés par des contrôles d'identité, de traçabilité et de supervision.

Les cibles habituelles sont les microcontrôleurs, SoC, FPGA et mémoires associées. Les outils modernes d'identification de broches réduisent fortement le coût d'entrée de l'attaque [7, 21].

4.3 Capacités d'un attaquant avec accès JTAG non protégé

Les capacités techniques observées incluent :

- Cartographie de mémoire et exfiltration complète des images firmware
- Désactivation de contrôles applicatifs (authentification, anti-rollback, checks métier)
- Persistance via patch binaire et réinjection de firmware
- Création de backdoors difficiles à distinguer d'une opération de support
- Rejeu sur série d'équipements lorsque les secrets ne sont pas individualisés

Le risque principal n'est pas seulement l'extraction d'information, mais la capacité à industrialiser l'attaque : une méthode validée sur un équipement pilote peut ensuite être reproduite à faible coût sur une flotte entière.

Dans des architectures multi-composants, un debug non protégé sur un seul maillon peut suffire à compromettre tout le système.

4.4 Implications pour la sécurité industrielle

Les impacts OT sont systémiques :

- **Sabotage opérationnel** : arrêt de ligne, dérive process, indisponibilité d'actifs
- **Vol de PI** : extraction d'algorithmes, paramètres propriétaires, calibration
- **Compromission supply chain** : insertion de firmware modifié avant livraison
- **Atteinte sûreté** : altération de fonctions critiques dans des environnements safety

Dans un contexte industriel, ces impacts se propagent souvent au-delà de l'actif compromis : indisponibilité de service, défaut qualité, non-conformité contractuelle, voire exposition réglementaire. Le pilotage du risque JTAG doit donc être traité comme un enjeu de continuité d'activité, pas uniquement comme une question de hardening local.

Les retours académiques, compétitions de sécurité embarquée et conférences spécialisées confirment la faisabilité opérationnelle de ces scénarios dès lors que la gouvernance du debug est insuffisante [15, 3, 4, 30].

4.5 Contexte des attaques de référence

Les cas suivants servent de contexte opérationnel pour la sécurisation du debug. L'objectif n'est pas d'établir une équivalence stricte entre incidents historiques, mais de relier des schémas d'attaque observés à des exigences techniques testables.

Cas (date)	Attaquant	Attaque	Actif ciblé	Gain associé
Extraction firmware IoT/OT (2018–2025)	Attaquant local avec accès physique opportuniste	Accès pads debug, dump mémoire, analyse binaire	MCU/SoC, secrets embarqués, logique de contrôle	Vol de PI, préparation de clones, recherche de vulnérabilités [14]
Bypass readout protection STM32 (2017–2024)	Attaquant physique outillé (laboratoire)	Glitch tension/horloge, contournement partiel de protections	Flash interne, états sécurité, chaîne de boot	Extraction de secrets, préparation d'attaque en série [26, 17]
Compromission maintenance terrain (retours OT)	Intervenant interne/sous-traitant malveillant	Abus de mode service, ouverture debug non tracée	Équipements en exploitation, journaux d'événements	Persistance discrète, sabotage ciblé, effacement de traces [16]
Fault injection MCU (2017–2024)	Attaquant physique avancé	EMFI/glitch durant vérifications critiques	Contrôles de boot, verrous debug, états lifecycle	Bypass de contrôle, élévation de capacités locales [5, 17, 25]
Outillage debug industriel détourné (continu)	Attaquant local formé aux outils constructeur	Scripts OpenOCD/-sondes debug, modification en exécution	Mémoire vive, registres sensibles, firmware actif	Injection de backdoor, altération du comportement process [21, 7]

TABLE 4.1 – Contexte des attaques de référence pour JTAG/SWD

4.6 Modèle d'attaquant et capacités techniques

Le modèle de menace est formalisé avec deux dimensions :

- **Profils AM-*** : position de l'attaquant dans le système (local, interne, physique avancé)
- **Capacités CAP-*** : moyens techniques effectivement maîtrisables

ID	Profil	Surface principale	Hypothèse de capacités
AM-01	Local opportuniste	Connecteurs externes, pads test accessibles, boîtier ouvert	CAP-01, CAP-02, CAP-03
AM-02	Mainteneur malveillant	Processus de maintenance, outillage atelier, mode service	CAP-02, CAP-03, CAP-04, CAP-06
AM-03	Interne écosystème (insider)	Chaîne de production, scripts de provisioning, secrets d'atelier	CAP-04, CAP-05, CAP-06
AM-04	Physique en laboratoire	PCB, alimentation, horloge, instrumentation EM	CAP-02, CAP-07, CAP-08
AM-05	Attaquant hybride supply chain + terrain	Lot de produits, sous-traitance maintenance, supervision insuffisante	CAP-03 à CAP-08

TABLE 4.2 – Profils d'attaquants considérés pour le risque debug

ID	Capacité	Description opérationnelle
CAP-01	Reconnaissance matérielle	Identification de broches debug, cartographie signaux, validation chaîne TAP/SWD.
CAP-02	Accès debug non autorisé	Halt/step, lecture/écriture mémoire, contrôle registres, exécution de scripts outillage.
CAP-03	Extraction et analyse firmware	Dump Flash/RAM, reverse engineering, recherche de secrets et mécanismes de contrôle.
CAP-04	Persistance locale	Patch binaire en mémoire/flash, insertion de mécanisme dormant, altération durable du comportement.
CAP-05	Compromission de secrets atelier	Abus de jetons/scripts de maintenance, réutilisation de credentials techniques.
CAP-06	Contournement de gouvernance debug	Ouverture hors procédure, usurpation d'identité outillage, suppression ou absence de traces.
CAP-07	Injection de fautes physiques	Voltage/clock glitch, EMFI, perturbation des vérifications critiques au boot.
CAP-08	Industrialisation de l'attaque	Reproductibilité multi-lots, automatisation des étapes d'extraction/modification, diffusion à grande échelle.

TABLE 4.3 – Capacités techniques du modèle de menace JTAG/SWD

4.7 Objectifs de sécurité liés à la gouvernance du debug

Les objectifs de sécurité sont normalisés ci-dessous sous un nommage `OS-*`.

ID	Objectif	Critère de vérification
OS-01	Confidentialité des actifs firmware	Un accès debug non autorisé ne permet pas l'extraction exploitable du firmware ni des secrets.
OS-02	Intégrité d'exécution	Un accès debug non autorisé ne permet pas l'altération persistante du comportement logiciel.
OS-03	Contrôle d'accès au mode debug	Toute ouverture debug est conditionnée à une identité forte et une autorisation explicite.
OS-04	Limitation temporelle et périmétrique	Les sessions debug sont bornées en durée, portée et révocables à tout moment.
OS-05	Cohérence boot/debug	L'autorisation debug reste cohérente avec l'état Secure Boot et la politique anti-rollback.
OS-06	Résistance aux attaques physiques de base	Les protections résistent aux tentatives opportunistes et détectent les perturbations simples.
OS-07	Traçabilité forensique des interventions	Toute session debug produit des preuves auditables, horodatées et corrélables.
OS-08	Continuité d'exploitation sécurisée	La maintenance reste possible en mode d'exception sans créer de canal permanent d'attaque.

TABLE 4.4 – Objectifs de sécurité normalisés pour la gouvernance JTAG/SWD

Fonctions de sécurité requises (FS-*). Les fonctions de sécurité implémentables sont identifiées sous un nommage `FS-*` et reliées aux objectifs `OS-*`.

ID	Fonction de sécurité	Exigence d'implémentation
FS-01	Verrouillage matériel de production	Suppression/protection connecteurs, anti-probing, séparation DEV/VAL/PROD traçable.
FS-02	Politique fuse/lock robuste	Profil OTP validé par famille composant, vérification post-programmation, preuve par lot.
FS-03	Authentification forte du debug	Challenge-réponse/certificats, refus par défaut des identités non autorisées.
FS-04	Contrôle temporel des sessions	Jetons courts, expiration automatique, révocation opérable et vérifiée.
FS-05	Couplage à l'état de boot	Ouverture conditionnée à l'intégrité de boot et à la politique de version active.
FS-06	Journalisation de bout en bout	Enregistrement demande/autorisation/actions/fermeture, horodatage fiable, export SIEM.
FS-07	Mécanismes d'exception maintenance	Procédure RMA/mode service à double validation et re-verrouillage automatique.
FS-08	Détection et réponse aux fautes	Détection d'anomalies physiques/logiques, réaction graduée, retour en état sûr.
FS-09	Gouvernance des secrets techniques	Individualisation des secrets, séparation des rôles, rotation/révocation des identités d'outil.

TABLE 4.5 – Fonctions de sécurité requises pour l'usage du debug

4.8 Traçabilité menaces → objectifs → fonctions

La matrice ci-dessous explicite la chaîne de justification de sécurité sur les scénarios d'attaque les plus représentatifs du risque JTAG/SWD.

Scénario	Capacités dominantes	Objectifs OS	Fonctions FS requises
Dump firmware via pads de test	AM-01 + CAP-01 / CAP-02 / CAP-03	OS-01, OS-03, OS-06	FS-01, FS-02, FS-03
Abus de mode maintenance terrain	AM-02 + CAP-04 / CAP-05 / CAP-06	OS-03, OS-04, OS-07, OS-08	FS-03, FS-04, FS-06, FS-07, FS-09
Bypass lock bits par fault injection	AM-04 + CAP-07 / CAP-08	OS-01, OS-02, OS-06	FS-02, FS-08
Injection de backdoor via session debug	AM-02 / AM-03 + CAP-02 / CAP-04 / CAP-06	OS-02, OS-05, OS-07	FS-05, FS-06, FS-07
Compromission de secrets d'atelier	AM-03 / AM-05 + CAP-05 / CAP-06 / CAP-08	OS-03, OS-04, OS-07	FS-03, FS-04, FS-06, FS-09
Attaque industrialisée multi-lots	AM-05 + CAP-03 à CAP-08	OS-01 à OS-08	FS-01 à FS-09

TABLE 4.6 – Traçabilité menaces-objectifs-fonctions pour le debug JTAG/SWD

5

Mécanismes de protection et verrouillage

Les exigences qui suivent ne sont pas des injonctions isolées. Elles doivent être lues comme un système cohérent, où chaque contrôle couvre une faiblesse différente : exposition physique, activation logique, identité des intervenants, et capacité de preuve post-événement.

5.1 Protections matérielles au niveau de la carte

Les protections carte sont la première couche de défense. Elles ne remplacent pas les mécanismes cryptographiques, mais elles augmentent significativement le coût et la discrétion nécessaires à une attaque physique.

STIG-JTAG-01 (OS-03, OS-06 → FS-01)	Supprimer les connecteurs de debug sur les versions de production, ou imposer un mécanisme d'accès physique exceptionnel (connecteur séquable, outillage dédié, scellement).
STIG-JTAG-02 (OS-01, OS-06 → FS-01)	Masquer, disperser ou enterrer les points de test critiques, et documenter une stratégie anti-probing proportionnée à la menace.
STIG-JTAG-03 (OS-06 → FS-01, FS-08)	Ajouter des contre-mesures passives (tirages, filtrage, diodes ESD/TVS, routage défensif) pour limiter l'activation non intentionnelle et augmenter le coût d'instrumentation.
STIG-JTAG-04 (OS-07 → FS-01, FS-09)	Séparer strictement les variantes de carte (développement, validation, production) avec nomenclature et traçabilité de configuration.

Cette séparation évite un défaut classique : la migration involontaire d'une configuration "lab" vers une série de production. En audit, la maîtrise de configuration est aussi importante que le

verrouillage technique lui-même.

Sur le plan opérationnel, cette section répond à une question simple : *un acteur non autorisé peut-il accéder physiquement et rapidement à un point de debug exploitable ?* Si la réponse est oui, les couches logiques ultérieures devront absorber une pression d'attaque beaucoup plus forte.

5.2 Verrouillage par fuse bits et lock bits

Le verrouillage par fuses et lock bits constitue le noyau dur de la stratégie "deny by default". Il doit être défini très tôt, puis validé sur des échantillons représentatifs de la production réelle (variation de lots, révisions silicon, conditions d'alimentation).

Différence technique OTP vs eFuse. Le terme OTP désigne une *propriété fonctionnelle* (programmable une seule fois), tandis que *eFuse* désigne une *implémentation physique* fréquente de cette propriété.

- **OTP (concept logique)** : zone non volatile écrite une fois, qui peut être réalisée de différentes manières (eFuse, anti-fuse, cellules OTP dédiées, etc.)
- **OTP basé Flash** : sur certaines plateformes, l'effet "one-time" est assuré par un contrôleur de sécurité et une séquence de programmation pilotée par logique embarquée (micro-code/firmware interne), et non par une irréversibilité physique intrinsèque de la cellule mémoire
- **eFuse (implémentation silicium)** : réseau de micro-fusibles "grillés" électriquement pour encoder un état irréversible, ensuite relu au boot ou via registres de sécurité
- **Lock bits** : bits de politique qui exploitent souvent ces zones OTP pour activer définitivement certaines restrictions (lecture flash, debug, transitions lifecycle)

Impact en résistance cyber. Du point de vue défensif, OTP/eFuse apportent une forte résistance à la reconfiguration logicielle distante, mais ne constituent pas une inviolabilité absolue face à un attaquant physique avancé.

- **Point fort** : irréversibilité matérielle, réduisant fortement le risque de réouverture accidentelle ou opportuniste du debug en production
- **Limite OTP Flash** : quand la protection repose sur un contrôleur embarqué (et non sur un fusible physique), la séquence peut être plus sensible aux attaques par faute (glitch/EMFI) sur la logique de contrôle
- **Propriété eFuse matériel** : le mécanisme interne de programmation/relecture est physique et ne dépend pas du même chemin logiciel de contrôle, ce qui réduit ce risque spécifique (sans supprimer les autres vecteurs physiques au niveau système)
- **Conséquence STIG** : combiner OTP/eFuse avec authentification forte, journalisation, contrôles post-programmation et tests de robustesse en conditions réalistes

STIG-JTAG-05 (OS-03, OS-06 → FS-02)	Définir un profil de verrouillage OTP par famille de composants et le valider sur banc avant industrialisation.
STIG-JTAG-06 (OS-01, OS-02, OS-06 → FS-02)	Appliquer les niveaux de protection maximum compatibles avec les besoins de maintenance et de certification.
STIG-JTAG-07 (OS-07 → FS-02, FS-06)	Vérifier après programmation l'état effectif des bits de sécurité et archiver la preuve par lot (traçabilité usine).

Les mécanismes varient selon les fabricants (RDP, secure debug mailbox, life-cycle states), mais le principe reste identique : réduire la surface debug active au strict nécessaire et contrôler explicitement les transitions d'état [26, 27, 20, 13, 29].

Exemple comparatif : certaines gammes ST peuvent aller vers un verrouillage très strict du debug en production, alors que certaines gammes NXP permettent un modèle d'ouverture conditionnelle par challenge-réponse. De même, sur des familles TI, l'accès JTAG peut dépendre d'un schéma de mot de passe et d'état de sécurité du composant, tandis que sur certaines familles Microchip la granularité des lock bits diffère d'une série à l'autre. En audit, cela implique de vérifier la documentation de la référence exacte (part number), et non de raisonner au seul niveau de la marque.

Une bonne pratique consiste à documenter une matrice "état produit / état debug" : pour chaque phase du cycle de vie, on explicite ce qui est autorisé, qui peut l'autoriser, et quelle preuve est conservée.

Cette approche aide les équipes d'audit à distinguer un comportement attendu (ex : ouverture temporaire contrôlée en atelier) d'une dérive non autorisée (ex : état debug actif persistant après maintenance).

5.3 Authentification et contrôle d'accès

Quand le debug doit rester possible pour la maintenance, l'objectif est de transformer une capacité technique en capacité gouvernée : identité forte, autorisation explicite, portée minimale, durée limitée et traçabilité complète.

STIG-JTAG-08 (OS-03 → FS-03)	Mettre en place une authentification forte du débogueur vers la cible (challenge-réponse, certificats, clefs asymétriques).
STIG-JTAG-09 (OS-04 → FS-04, FS-07)	Limiter temporellement les sessions de debug (jetons courts, expiration automatique, révocation).
STIG-JTAG-10 (OS-05 → FS-05)	Coupler l'autorisation debug au niveau d'intégrité du boot (Secure Boot valide, état anti-rollback conforme).
STIG-JTAG-11 (OS-07 → FS-06)	Journaliser les événements de debug (demande, autorisation, opération, fermeture), avec horodatage fiable et export SIEM.

Sans ce volet de gouvernance, un mécanisme de debug "sécurisé" reste difficile à auditer et à exploiter en réponse à incident. L'exigence de journalisation est donc une exigence de sécurité et d'exploitabilité.

En environnement OT, cette gouvernance doit rester compatible avec les contraintes terrain : fenêtres d'intervention courtes, connectivité parfois limitée et nécessité de reprise rapide. Les mécanismes retenus doivent donc être robustes *et* opérables.

5.4 Solutions avancées

Pour des environnements à criticité élevée (SL-3/SL-4), les mesures additionnelles suivantes sont recommandées :

- Debug sécurisé via tunnel chiffré ou protocole d'authentification dédié
- Restriction du debug à des régions mémoire non sensibles (partitionnement TEE/TrustZone, PMP)
- Détection d'intrusion physique avec réaction graduée (alerte, blocage, effacement conditionnel)
- Chaîne de confiance matérielle (HSM/TPM/SE) pour la gestion des secrets de maintenance

Ces mesures avancées sont particulièrement pertinentes lorsque l'attaquant visé dispose de moyens matériels significatifs (laboratoire, outillage de fautes, expertise reverse). Elles ne sont pas réservées aux secteurs régulés, mais deviennent indispensables dès que l'impact métier d'une compromission est majeur.

Il est recommandé de les déployer progressivement, selon une logique de maturité : d'abord la gouvernance de base et les verrous de production, puis l'authentification forte du debug, puis enfin les fonctions de détection/réaction avancées.

6
Limites des mécanismes de protection existants

6.1 Contournements de fuses et lock bits

Les verrouillages OTP réduisent fortement le risque mais ne constituent pas une preuve absolue d'inviolabilité. Des attaques par glitch (tension/horloge), fautes transitoires ou failles d'implémentation peuvent ouvrir des fenêtres de contournement sur certaines générations de composants [5, 17, 25].

Conséquence pour le STIG : les fuses doivent être complétés par des contrôles de détection, de limitation d'impact et de supervision opérationnelle.

6.2 Failles d'architecture

Les causes récurrentes observées en audit sont :

- Interface debug activable sans authentification matérielle
- Mode maintenance permanent faute d'état de cycle de vie terminal
- Voies de secours non documentées conservées en production
- Rupture d'alignement entre sécurité boot et sécurité debug

Ces situations résultent souvent d'un découplage organisationnel : équipes hardware, firmware et opérations appliquent des hypothèses de menace différentes. Le STIG doit précisément servir de langage commun pour éviter ces divergences.

Ces défauts relèvent de l'architecture système et non d'une simple mauvaise configuration locale.

6.3 Limites des approches de masquage

Le masquage physique seul n'est pas suffisant. Des outils de découverte de signaux, l'analyse de routage et des méthodes de rétroconception peuvent reconstituer l'interface de debug avec un coût décroissant.

L'usage de modèles d'IA pour assister la rétroingénierie de PCB (classification de pads, hypothèses de netlist) doit être considéré comme un facteur d'accélération pour l'attaquant, et non comme une hypothèse lointaine.

6.4 Contraintes des mécanismes de protection

L'implémentation d'un debug sécurisé pose des contraintes pratiques :

- Protection et renouvellement des secrets d'accès
- Individualisation des clefs par équipement à grande échelle
- Coût CPU/mémoire des protocoles cryptographiques embarqués
- Consommation énergétique et impact thermique en contexte edge
- Compatibilité avec les exigences temps réel et de sûreté

Une erreur fréquente est de traiter ces contraintes trop tard, au moment de l'industrialisation. Il est préférable de les intégrer dès les choix d'architecture afin d'éviter des compromis faibles de dernière minute (secrets partagés, durées de jeton excessives, journalisation incomplète).

Le STIG recommande d'arbitrer ces contraintes dès la phase d'architecture, avec une modélisation explicite des risques résiduels.

6.5 Contraintes opérationnelles industrielles

En OT, l'interdiction totale du debug peut être irréaliste en maintenance terrain. La bonne pratique est d'organiser un **debug d'exception**, strictement contrôlé :

- activation temporaire,
- approbation duale,
- session journalisée,
- révocation et re-verrouillage automatique,
- preuve d'intervention archivée

Ce modèle répond au besoin opérationnel sans normaliser un accès permanent. Il réduit fortement le risque d'abus interne, d'erreur de maintenance et de compromission de comptes techniques.

7
État de l'art des attaques et outillages

7.1 Outils de recherche et d'exploitation

Les outils open-source et commerciaux disponibles abaissent la barrière à l'entrée :

- **OpenOCD** et sondes compatibles pour scripts d'accès mémoire et contrôle d'exécution [21]
- **JTAGulator** pour identifier automatiquement interfaces et brochages [7]
- Suites de debug fournisseur (ST-LINK, Lauterbach, Segger, etc.) souvent très puissantes
- Outils de fuzzing firmware et d'analyse binaire combinables avec des dumps obtenus via debug [6, 24]
- Plateformes d'injection de fautes (glitching) pour tester robustesse réelle des verrouillages [17]

Du point de vue du défenseur, ces mêmes outils doivent être intégrés à la stratégie de test de sécurité produit. L'objectif n'est pas uniquement de vérifier qu'un bit de protection est activé, mais de démontrer qu'un attaquant outillé ne peut pas obtenir un accès utile dans des conditions réalistes de terrain.

7.2 Publications académiques récentes

La littérature utile pour un programme OT embarqué inclut notamment :

- analyses de compétitions embarquées (MITRE eCTF) mettant en évidence les lacunes fréquentes de design, notamment la gestion des secrets et des modes maintenance [15]
- études de cas sur extraction de firmware et contournement de protections debug, utiles pour comprendre les chemins d'attaque réellement employés [14]
- travaux de synthèse sur la sécurité des objets embarqués et les attaques physiques, qui aident à prioriser les contre-mesures selon le niveau de menace [30, 28]
- retours d'expérience de conférences techniques (Black Hat, DEF CON, Hardwear.io, REcon), précieux pour suivre les techniques émergentes et leurs preuves de faisabilité [3, 4, 8, 23]

Pour un usage STIG, ces sources doivent être lues avec une grille opérationnelle : quelles hypothèses d'attaque sont pertinentes pour votre contexte OT, quelles mesures sont testables et quelles preuves sont auditables.

7.3 Scénarios d'attaque documentés

Les scénarios suivants sont représentatifs de produits OT critiques :

- **Médical connecté** : extraction de firmware pour compréhension du protocole et préparation d'attaque sur la flotte
- **Automobile** : récupération de secrets dans des calculateurs périphériques puis pivot vers des fonctions de confiance
- **IoT industriel** : clonage d'équipements et insertion de firmware modifié dans la chaîne de maintenance
- **Bypass de protections lecture** : tentatives combinant accès debug, défauts transitoires et erreurs de configuration

Chaque scénario combine généralement trois dimensions : accès physique, faiblesse de gouvernance et défaut technique exploitable. Le traitement efficace impose donc une réponse croisée entre architecture, processus et supervision.

Pour les équipes produit, ces scénarios doivent être rejoués sous forme d'exercices de validation interne. L'objectif n'est pas de reproduire toute la sophistication d'un laboratoire offensif, mais de vérifier que les contrôles clés résistent à des tentatives crédibles, répétables et documentées.

Ces scénarios justifient une posture par défaut *deny-debug*, avec exceptions contractualisées.

8

Bonnes pratiques de sécurité pour équipements industriels

Ce chapitre fournit un cadre d'arbitrage. En pratique, les équipes OT doivent concilier disponibilité, maintenabilité et exigences cyber. Une bonne pratique de sécurité réduit le risque sans bloquer l'exploitation normale du système.

8.1 Alignement sur les standards industriels

Le cadrage de sécurité doit s'aligner sur :

- IEC 62443-4-1 pour les pratiques de développement sécurisé [9]
- IEC 62443-4-2 pour les exigences techniques de composants [10]
- NIST SP 800-193 pour la résilience de plateforme firmware [18]
- guides institutionnels (ANSSI, NIST) sur l'hygiène de sécurité embarquée [1, 19]

Pour les environnements contraints, la cible minimale recommandée est SL-2 avec trajectoire vers SL-3 sur les actifs critiques.

8.2 Stratégies de défense en profondeur

Une stratégie robuste combine au minimum :

- verrouillage matériel en production,
- Secure Boot et anti-rollback,
- authentification forte des opérations de maintenance,
- segmentation réseau OT,
- supervision SOC/CSIRT des événements de debug

La valeur de cette approche vient de la complémentarité des couches : si une mesure échoue (exemple : erreur de configuration lock bits), les autres couches doivent encore empêcher l'attaque ou au minimum en permettre la détection rapide.

Cette redondance contrôlée est particulièrement importante dans les programmes multi-sites, où les pratiques de production et de maintenance peuvent varier d'une usine à l'autre.

Aucune mesure individuelle n'est suffisante. Le niveau de risque est fonction de la cohérence globale de l'architecture et de la qualité d'exécution opérationnelle.

8.3 Gestion du compromis sécurité/maintainabilité

Le compromis s'organise autour d'un processus industrialisé :

- mode de récupération sécurisé, activable sur justification
- demandes d'accès debug approuvées et horodatées
- session limitée à un périmètre technique explicite
- fermeture automatique et vérification post-intervention
- conservation des preuves (journal, hash firmware, rapport d'action)

Ce processus doit être testé régulièrement, comme un exercice de crise : délai d'obtention d'accès, qualité des preuves produites, capacité de révocation, et retour à un état verrouillé vérifiable.

Dans une logique d'amélioration continue, les incidents et quasi-incidents liés au debug doivent alimenter un retour d'expérience formalisé : mise à jour des procédures, durcissement des seuils d'autorisation et adaptation des outils.

Cette approche permet de préserver la maintenabilité sans banaliser le privilège debug.

8.4 Recommandations par secteur industriel

Énergie et utilities. Prioriser le verrouillage permanent, avec un processus d'exception hors-ligne et un contrôle renforcé (objectif SL-3/SL-4).

Automobile. Séparer strictement les canaux diagnostic atelier et les fonctions client ; appliquer une gestion rigoureuse des identités d'outil.

Médical. Assurer la conformité réglementaire (MDR/FDA), l'auditabilité et la maîtrise des changements firmware.

Manufacturier. Protéger la propriété intellectuelle de procédés et prévenir les clones via individualisation des secrets et attestations d'intégrité.

A

Tableau comparatif des implémentations JTAG par fabricant

Fabricant	Mécanisme principal	Niveau de contrôle debug	Points d'attention STIG
STMicro	RDP, états lifecycle, authentification debug sur gammes récentes	De blocage lecture à accès conditionnel authentifié	Vérifier configuration usine, cohérence boot/debug et procédure de déverrouillage d'exception [26, 27]
NXP	Secure debug mailbox, cycle de vie, clés de provisionnement	Accès contrôlé via protocole d'authentification	Protéger la chaîne de provisioning et journaliser les autorisations [20]
Microchip	Bits de protection mémoire, modes lock/debug, variation selon familles	De verrouillage simple à modes de maintenance sécurisés	Exemple : certaines familles proposent un lock mémoire strict, d'autres des modes de service plus fins ; tester par référence exacte [13]
Texas Instruments	Protection JTAG par mot de passe, états sécurité et options spécifiques au composant	Accès conditionnel selon configuration et device	Exemple : selon la famille, l'ouverture JTAG peut être liée à un mot de passe/état sécurité ; valider en conditions réelles [29]

Table A.1 – Synthèse comparative des mécanismes JTAG/SWD par fabricant

B

Références normatives et réglementaires

Les textes suivants doivent être considérés comme la base documentaire du présent STIG :

- IEC 62443-4-1 : processus de développement sécurisé des composants
- IEC 62443-4-2 : exigences techniques de sécurité pour composants IACS
- ISO/SAE 21434 : cybersécurité automobile (pertinent pour équipements mobiles/transport)
- NIST SP 800-193 : résilience de plateforme et protection firmware
- Référentiels complémentaires : NISTIR 8259 et recommandations ANSSI sur sécurité IoT/embarqué

Ces références doivent être traduites en exigences internes vérifiables (spécification, test, audit, exploitation).

C

Checklist d'implémentation par phase du cycle de vie

La checklist suivante sert de base d'homologation interne. Chaque item doit être marqué *conforme / non conforme / non applicable* et rattaché à une preuve.

Pour un usage "niveau expert audit", cette checklist doit être couplée à des critères de profondeur : qualité des preuves, reproductibilité des tests, indépendance des vérifications et capacité à détecter des écarts de configuration entre lots.

Phase de conception

- Connecteurs debug absents de la version production ou protégés par mécanisme physique documenté
- Stratégie d'exposition minimale des pads de test (routage, camouflage, accès outillage)
- Sélection de composants supportant un debug sécurisé et des états lifecycle robustes
- Chaîne de boot sécurisée avec ancre de confiance et anti-rollback

Cette phase conditionne l'essentiel du niveau de sécurité atteignable. Un mauvais choix de composant ou d'architecture debug est difficilement rattrapable en fin de cycle.

Phase de production

- Programmation des fuses/locks selon profil validé et revu
- Vérification post-programmation automatisée et traçable par lot
- Gestion sécurisée des secrets usine (HSM, séparation des rôles, audit)
- Contrôle qualité incluant tentative de lecture mémoire non autorisée

La robustesse réelle se joue ici : une protection bien conçue mais mal industrialisée devient un risque systémique sur l'ensemble de la série.

Phase de déploiement

- Contrôle d'intégrité des équipements reçus et de leur état de verrouillage
- Supervision des événements de maintenance et de debug
- Procédure de révocation d'identités outils et techniciens
- Politique d'escalade incident en cas de suspicion de compromission physique

Le déploiement doit inclure une logique de surveillance continue. L'absence d'événement debug n'est pas une preuve de sécurité si aucune télémétrie fiable n'est collectée.

Phase de maintenance

- Debug d'exception uniquement sur ticket approuvé et horodaté
- Jetons d'accès temporaires, individuels, à portée minimale
- Journalisation complète des sessions et conservation des preuves
- Re-verrouillage automatique et vérification d'état en fin d'intervention

Cette phase est la plus exposée aux erreurs humaines. Les procédures doivent être suffisamment simples pour être appliquées en conditions terrain, mais suffisamment strictes pour rester auditables.

Procédure de test de validation sécurité JTAG (par lot)

- Test 1 : tentative d'accès debug sans authentification (échec attendu)
- Test 2 : tentative de lecture mémoire protégée (échec attendu)
- Test 3 : test de robustesse aux erreurs d'état (reboot, brown-out, reset)
- Test 4 : session de maintenance autorisée, traçabilité complète et fermeture propre
- Test 5 : vérification post-maintenance de l'état de verrouillage et de l'intégrité firmware

Les résultats doivent être historisés par lot, version hardware et version firmware afin d'identifier rapidement toute régression de sécurité.

C.1 Matrice d'audit expert des exigences STIG-JTAG-01 à STIG-JTAG-11

La matrice suivante formalise, pour chaque exigence, le triplet *critère de conformité / méthode de test / preuve attendue*. Elle peut être utilisée telle quelle en annexe d'un plan d'audit.

Cette matrice peut être enrichie avec des seuils quantitatifs internes (taux d'échec acceptable, délai maximum de révocation, taux de couverture de vérification par lot) afin de piloter la maturité sur plusieurs trimestres.

Exigence	Critère de conformité	Méthode de test	Preuve attendue	Fréquence	Sévérité
STIG-JTAG-01	Absence de connecteur debug exploitable en production ou accès physique exceptionnel documenté	Inspection physique, revue BOM/PCB, test d'accès avec outillage standard	Photos d'échantillons, revue de plans, compte-rendu de tentative d'accès	À chaque révision HW	Critique
STIG-JTAG-02	Points de test sensibles non accessibles sans démontage/instrumentation spécifique	Revue layout, inspection visuelle, essai de repérage broches	Rapport de revue PCB, checklist anti-probing signée	À chaque révision HW	Élevée
STIG-JTAG-03	Contre-mesures passives présentes et conformes au design de référence	Contrôle schéma, mesure électrique ponctuelle, test de stabilité activation debug	Schémas approuvés, relevés de mesure, rapport de validation	Par lot initial + changements	Élevée
STIG-JTAG-04	Variantes DEV/VAL/-PROD clairement séparées et traçables	Audit configuration, vérification nomenclature et marquage	Registre de configuration, traçabilité de fabrication	Mensuelle + à chaque release	Critique
STIG-JTAG-05	Profil fuse/lock défini et validé sur la famille composant	Exécution script de lecture état sécurité sur banc	Résultats automatiques signés, baseline de configuration	Chaque lot	Critique
STIG-JTAG-06	Niveau de protection maximal appliqué selon politique approuvée	Revue politique vs état réel composant	Compte-rendu de conformité + dérogations validées	Chaque lot	Critique
STIG-JTAG-07	Vérification post-programmation archivée	Contrôle en fin de ligne + échantillonnage aléatoire	Journaux usine horodatés, preuves d'échantillonnage	Chaque lot	Critique
STIG-JTAG-08	Authentification forte requise pour toute ouverture debug	Test d'accès avec identités valides/invalides, test rejeu	Traces d'authentification, journal des rejets	Trimestrielle + releases	Critique
STIG-JTAG-09	Durée et portée des sessions debug limitées et révocables	Test d'expiration jeton, test révocation en temps contraint	Logs d'expiration, preuve de révocation	Trimestrielle	Élevée
STIG-JTAG-10	Autorisation debug conditionnée à l'intégrité boot	Test sur firmware valide/invalide/rollback	Journaux de décision d'accès, hashes et états boot	Chaque release firmware	Critique
STIG-JTAG-11	Journalisation complète et exploitable SOC/SIEM	Revue logs, test de corrélation, simulation incident	Extraits SIEM, règles de corrélation, piste d'audit bout en bout	Mensuelle + exercices	Élevée

Table C.1 – Matrice d'audit expert pour les exigences STIG-JTAG-01 à STIG-JTAG-11

D

Ressources et outils de référence

Les ressources ci-dessous doivent être vues comme un socle de travail, et non comme une liste exhaustive. L'enjeu est de maintenir une capacité d'évaluation continue des attaques et des contre-mesures, adaptée au rythme des évolutions produit.

Outils open-source utiles en défense et en audit

- OpenOCD (pilotage JTAG/SWD et scripts de test)
- JTAGulator (identification de broches debug)
- GDBFuzz (fuzzing assisté de firmwares)
- Binwalk et chaînes d'analyse binaire pour inspection post-extraction

Ces outils permettent de reproduire des chaînes d'attaque de manière contrôlée et de mesurer objectivement la robustesse d'un produit.

Outils commerciaux fréquemment rencontrés

- Débogueurs professionnels haut de gamme (Lauterbach, Segger, etc.)
- Plateformes de test d'injection de fautes et d'évaluation matérielle
- Solutions de gestion de clés et HSM pour l'industrialisation sécurisée

Leur intérêt principal est la profondeur d'analyse et l'automatisation des campagnes de test, particulièrement utile pour les programmes à forte volumétrie ou à exigences de certification élevées.

Communautés et conférences à suivre

- Black Hat, DEF CON, Hardwear.io, REcon
- Publications NIST, ANSSI, ENISA et retours d'expérience industriels
- Compétitions de type eCTF pour observer les tendances d'attaque/défense

Ce corpus permet de maintenir le STIG vivant et aligné sur l'état de l'art technique.

Bibliographie

[1] ANSSI. Recommandations de sécurité relatives à l'iot. Guide ANSSI, 2020.

[2] Arm Ltd. *ARM Debug Interface Architecture Specification (ADIv5/ADIv6)*, 2023.

[3] Black Hat. Black hat conference archives. Conference archives, 2026.

[4] DEF CON. Def con talks and archives. Conference archives, 2026.

[5] Jean-Max Dutertre et al. *Fault Injection Attacks and Countermeasures*. Springer, 2021.

[6] GDBFuzz Contributors. Gdbfuzz. Projet open-source, 2026.

[7] Joe Grand. Jtagulator : Assisted discovery of on-chip debug interfaces. Projet open-source, 2026.

[8] Hardwear.io. Hardwear.io conference archive. Conference archives, 2026.

[9] IEC. Security for industrial automation and control systems – part 4-1 : Secure product development lifecycle requirements (iec 62443-4-1). Norme IEC, 2018.

[10] IEC. Security for industrial automation and control systems – part 4-2 : Technical security requirements for iacs components (iec 62443-4-2). Norme IEC, 2019.

[11] IEEE. Ieee standard for test access port and boundary-scan architecture (ieee 1149.1). Standard IEEE, 2013.

[12] ISO/SAE. Road vehicles – cybersecurity engineering (iso/sae 21434). Norme ISO/SAE, 2021.

[13] Microchip Technology. *Microchip MCU Security and Debug Access Control Documentation*, 2026.

[14] A. Miller et al. Practical firmware extraction from secure iot devices. *Proceedings of Embedded Security Workshop*, 2021.

[15] MITRE. Embedded capture the flag (ectf). Competition and technical resources, 2026.

[16] MITRE. Mitre att&ck for ics. Knowledge base, 2026.

[17] NewAE Technology. Chipwhisperer platform. Open hardware / open-source platform, 2026.

[18] NIST. Platform firmware resiliency guidelines. Technical Report SP 800-193, National Institute of Standards and Technology, 2018.

[19] NIST. Foundational cybersecurity activities for iot device manufacturers. Technical Report NISTIR 8259, National Institute of Standards and Technology, 2020.

[20] NXP Semiconductors. *NXP Secure Debug and Device Lifecycle Management Documentation*, 2026.

[21] OpenOCD Project. Open on-chip debugger. Projet open-source, 2026.

[22] OWASP. Firmware security testing guide. OWASP Project, 2024.

[23] REcon. Recon conference archive. Conference archives, 2026.

[24] ReFirmLabs. Binwalk firmware analysis tool. Projet open-source, 2026.

[25] Riscure. Fault injection knowledge base and training material. Industry resources, 2026.

[26] STMicroelectronics. *STM32 Reference Manuals and Readout Protection (RDP) documentation*, 2026.

[27] STMicroelectronics. *STM32H5 Debug Authentication Application Notes*, 2026.

[28] Mark Tehranipoor and Cliff Wang. *Introduction to Hardware Security and Trust*. Springer, 2011.

[29] Texas Instruments. *Texas Instruments Device Security and JTAG Lock Documentation*, 2026.

[30] X. Zhou et al. Security analysis of embedded devices : Threats, attacks, and defenses. *Sensors*, 2023.

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