



MYC-JMX95 Security Vulnerability Remediation and CRA Compliance Support

Document Type	Technical Statement / Compliance Support
Applicable Platforms	MYC-JMX95
Document Version	V0.9
Issued By	MYIR Electronics Limited

[Make Your Idea Real](#)

Web: en.myir.cn | Mail: sales@myir.cn | Tel: +86-755-22984836



1 Security Vulnerability Remediation Scope

This document explains, for customers planning to place final products on the EU market, the MYIR system-on-module and Linux BSP capabilities and processes in the following areas: security vulnerability remediation, security issue response, static security scanning, CRA compliance support and ongoing maintenance.

1.1 Objects Covered

MYIR security vulnerability remediation covers the complete Linux BSP software stack delivered by MYIR:

1. Linux kernel (including drivers and security subsystems)
2. U-Boot bootloader
3. OP-TEE trusted execution environment (where enabled on the platform)
4. BSP packages, user-space libraries and middleware in the root file system
5. Third-party open-source components integrated, built and validated by MYIR

MYC-JMX95 platform notes

- Objects covered: TF-A + U-Boot, OP-TEE, Linux kernel (6.12.49), user space and middleware
- SDK: L6.12.49 V1.0.0 (Yocto 5.2 walnascar); sources are delivered under **04-Sources**
- **Security material in the release package:** the *MYD-JMX95X Security Practice Guide* in EN and ZH (SoftwareFiles/Security/Linux6.12.49 Distribution V1.0.0/) together with **04-Sources/Security-package.tar.gz** (security toolkit: CST signing tool, imx-mkimage, the OP-TEE build script and the SWUpdate packaging scripts)

1.2 Ownership and Division of Responsibility

- **MYIR is responsible for:** vulnerability investigation, assessment and remediation of the components contained in the BSP it delivers, delivered as a security update (OTA) or a version release.
- **The customer is responsible for:** remediation of vulnerabilities in application software, business logic and code developed, integrated or deployed by the customer. MYIR provides remediation methods and interface guidance.

1.3 Basis for Remediation

- Sources: vendor advisories, the CVE database (NVD), upstream patches and public information from security research organisations.
- Process: the exploitability and impact of each vulnerability under MYIR's default configuration are assessed before remediation.
- Marking: components that are not enabled or not reachable in the current BSP version are explicitly marked with their affected status in the VEX report.

1.4 Remediation Artifacts

1. **Security update (OTA):** an A/B partition scheme based on an OTA solution (RAUC, SWUpdate, Mender, etc.) with signature verification and automatic rollback on failure.

2. **Version release:** complete fixes shipped with a new BSP release.
3. **Standalone security patch:** security fixes can be separated from feature updates and delivered independently.

1.5 Support Period

- MYIR provides security vulnerability remediation only for BSP versions within the support period.
- Support duration and the version policy are described in Chapter 5; for versions past the support period MYIR provides upgrade and migration guidance.

2 Security Issue Response Mechanism

To meet the CRA requirements for vulnerability management and incident response (the EU vulnerability reporting obligation applies from 11 September 2026), MYIR has established a Product Security Incident Response Team (PSIRT).

2.1 PSIRT Responsibilities

- Receiving and acknowledging security vulnerability reports
- Vulnerability impact assessment
- Defining and executing the remediation plan
- Publishing security advisories and VEX reports
- Sharing security information with customers

2.2 Reporting Channel

- **Where to report:** the **PSIRT** page in the MYIR Developer Center — developer.myir.cn/user/security.html. That page publishes the reporting e-mail, the PSIRT scope and workflow, security advisories and the **PGP public key**.
- **Obtain the reporting mailbox from that page.** In addition, **every vulnerability report must be encrypted with the MYIR PSIRT PGP public key published there before it is sent — never send a report unencrypted** (vulnerability details are sensitive and an unencrypted e-mail may leak them).
- **How to encrypt:** import that PGP public key with a tool such as OpenPGP / GnuPG and encrypt the report; e-mail built-in plugins and S/MIME are **not recommended**, and passphrases, symmetric encryption or passwords should **never** be used.
- **Suggested content:** contact details (name, contact information, location, company, position); affected product model and software/hardware versions, how the issue was found, reproduction steps, impact description and on-site handling; attachments (full vulnerability report, logs, screenshots, remediation suggestions).

2.3 Response Timeframe (SLA)

Stage	Time	Description
Initial alert	Within 24 hours	Acknowledge the report, give a preliminary assessment and receipt
Detailed notification	Within 72 hours	List of affected components and versions, preliminary impact analysis
Remediation report	Within 14 days	Remediation plan, mitigations and update schedule

2.4 Vulnerability Assessment and VEX Disclosure

MYIR follows a coordinated vulnerability disclosure (CVD) policy, assesses exploitability and impact, and issues VEX reports in accordance with the CycloneDX VEX specification.

VEX status	Meaning
Not Affected	Not exploitable under the default BSP configuration (the vulnerable module is not enabled)
Under Investigation	Assessment in progress
Fixed	Fixed in the current version
Affected	Confirmed exploitable; upgrade or mitigation required immediately
False Positive	Scanner false positive (version match but the patch is already merged)

3 Handling of Static Security Scan Results

Static security scanning identifies potential security defects in BSP source code and open-source components (mapped to CWE categories). It complements manual audit and vulnerability management and can be performed with tools such as CodeQL.

3.1 Position and Scope of Scanning

- Position: one of the static analysis inputs to the vulnerability management process.
- Scope: semantic-level defect detection for C/C++ components and key sources inside the BSP.
- Detects: buffer overflows, injection, out-of-bounds access, unsafe dereferences and similar defect classes.

3.2 Review of Results

MYIR reviews scan results supplied by customers item by item and decides whether each alert constitutes a real, exploitable vulnerability, based on:

1. Component version and origin
2. Call path and reachability
3. Mitigations present in the default configuration
4. False-positive identification (version match with the patch already merged, unreachable code, and so on)

3.3 Classification and Disposition

- Confirmed issues enter a single vulnerability tracking process by risk level (high / medium / low).
- Results are merged and prioritised together with CVE scanning (cve-check) and VEX analysis.
- High-risk issues are handled first.

3.4 Closing the Loop

- Confirmed issues are fixed through BSP security updates and re-verified in subsequent versions.
- The output of a single tool is never the final verdict; static scanning, CVE monitoring, manual audit and VEX analysis together form a closed loop.

4 Support for CRA Requirements

The EU Cyber Resilience Act (CRA), **Regulation (EU) 2024/2847**, establishes horizontal cybersecurity requirements for products with digital elements placed on the EU market and imposes lifecycle security requirements on manufacturers. Relevant cybersecurity standards, including IEC 62443 where applicable, may support implementation and assessment; **IEC 62443 is not itself the legal basis of the CRA.**

4.1 CRA Timeline

Date	Event	Impact
2024-12-10	CRA enters into force	The regulation is in force; start compliance preparation
2026-09-11	Vulnerability reporting obligation applies	A PSIRT and a vulnerability reporting process are required
2027-12-11	Full application	All products placed on the EU market must be fully compliant

4.2 Key CRA Requirements

1. Secure by default and secure by design: default security configuration, minimised attack surface, no weak credentials and no unnecessary open ports
2. Software bill of materials (SBOM): a machine-readable component inventory
3. Vulnerability management and incident response: discovery / assessment (VEX) / remediation / reporting, plus a PSIRT
4. Security updates (OTA): provided continuously during the support period and protected by signature and integrity verification
5. Technical documentation and risk assessment: security architecture documentation and a TARA risk assessment report

4.3 Four Compliance Mechanisms and Corresponding Capabilities

Mechanism	Description	Benefit to the customer
Automatic SBOM generation	Yocto create-spdx produces SPDX (machine-readable JSON), converted to CycloneDX by script; covers the whole image / rootfs / recipes / packages — kernel, U-Boot, BSP packages and user space	No toolchain to build; the result imports directly into vulnerability scanners
CVE monitoring + VEX	cve-check matches NVD automatically; High / Critical findings receive manual VEX analysis	The VEX report can be used directly in compliance filing
PSIRT response	Handled through the PSIRT page in the MYIR Developer Center (see Chapter 2); 24 h alert → 72 h notification → 14 day report	Meets the CRA vulnerability reporting obligation
Security update (OTA)	RAUC / SWUpdate etc. (A/B partitions + signature verification, tamper and rollback protected)	Patches deploy directly; at least 5 years of free security updates
Compliance documents	TARA risk assessment / security architecture document / CVD policy / declaration templates	Reuse of document templates

4.4 IEC 62443-4-2 Product-Level Security Implementation

IEC 62443-4-2 defines component-level (SL-C) technical security requirements covering identification and authentication control (IAC), use control (UC), system integrity (SI), data confidentiality (DC), restricted data flow (RDF), timely response to events (TRE) and resource availability (RA). MYIR provides concrete implementation for four areas: **secure boot, secure storage, secure update and access control**.

Each subsection below adds the platform-specific implementation and document sections for the NXP platform (MYC-JMX95) on top of the generic capability.

4.4.1 Secure Boot

- Root of trust: the public key hash (SRK / ROTPK) stored in on-chip OTP, verifying bootloader and kernel signatures step by step
- Chain of trust: BootROM → Loader/SPL → OP-TEE → U-Boot → boot.img (kernel/DTB); boot stops if any stage fails verification
- Trusted execution environment: OP-TEE based on ARM TrustZone
- Anti-rollback: OTP version counters / eFuse counters prevent downgrading to an older firmware version

MYC-JMX95 implementation

- **Implementation guide:** SoftwareFiles/Security/Linux6.12.49 Distribution V1.0.0/MYD-JMX95X Security Practice Guide-V1.0.pdf — §1 Overview (AHAB architecture) / §2 Safe Startup (basic sec BOOT, full-link) / §3 Generating a secure image with Yocto / §4 Manual sign and encrypt (§4.1 CST, §4.2 PKI, §4.3 SRK)
- Mechanism: NXP **AHAB** (Advanced High Assurance Boot) chain of trust, rooted in the **ELE** (EdgeLock Enclave) hardware security subsystem
- **SDK path:** the basic route is enabled by the single line `DISTRO_FEATURES:append = "sec_boot"` in the machine configuration `$SDK/sources/meta-myr/meta-myr-bsp/conf/machine/myd-jmx95-15x15-lpddr5.conf`; the full-link route sets the four switches `ENABLE_DM_VERITY` / `ENABLE_OVERLAYFS` / `ENABLE_ROOTHASH_SIGN` / `ENABLE_FITIMAGE_SIGN` to "1" in the same file. The signing logic sits in the security layer `$SDK/sources/meta-myr/meta-myr-security/classes/` (myir-boot-signed.bbclass is the entry point, myir-ahab.bbclass applies the AHAB signature to the imx-boot container, myir-kernel-container-sign.bbclass signs the kernel container, myir-dmverity.bbclass handles rootfs verification); the partition templates are under `meta-myr-security/wic/` (myd-jmx95-15x15-lpddr5-signed.wks.in for the basic route, ...-dmverity-overlay.wks.in for the full link); preset keys and certificate templates are under `meta-myr-security/files/keys/` and the layer `README.md` gives the environment setup, key generation, build and flashing steps. Per §3 of the guide, the security-layer content is provided through the channel listed in the release package's after-sales and technical support appendix
- **Deliverables: in the package** — 04-Sources/Security-package.tar.gz (the guide's command examples of §4 unpack and use it under 04-Sources/), which contains `IMX_CST_TOOL_NEW.tgz` with the `cst` and `srktool` utilities, `imx-mkimage` and the one-

shot OP-TEE build script `build.sh`, plus the secure-boot burn image `myir-image-burn-...rootfs.wic.zst` under `02-Images/`;

configuration — the machine file, security-layer bbclasses and WKS templates listed above;

build output — after `bitbake myir-image-emmc` (or `myir-image-burn`) the signed firmware `imx-boot-signed` and kernel containers `os_cntr_signed*.bin` (for the full link, `fitImage-<machine>.bin` and `*.ext4.verity`), together with the SRK hash `SRK_1_2_3_4_fuse.bin` and the eFuse write commands `fuse-cmds.txt`, under `$SDK/build-*/tmp/deploy/images/myd-jmx95-15x15-lpddr5/`;

on the device — the SRK hash is programmed by running the `fuse prog -y 16 / 17 ...` lines from `fuse-cmds.txt` in U-Boot, and `ahab_close` is executed only once the signed image boots with no HAB events (one-time programmable, irreversible)

4.4.2 Secure Storage

- Secure storage medium: RPMB (hardware replay and rollback protection in eMMC)
- Key management: encryption keys are managed by a key-management trusted application (TA / KeyBox) inside OP-TEE and never appear in normal-world memory in plaintext
- System integrity: `dm-verity` provides block-level integrity verification of read-only system partitions; `dm-crypt` provides transparent partition encryption
- Cryptography: AES, RSA, SHA and a hardware true random number generator (TRNG)

MYC-JMX95 implementation

- **Implementation guide:** `SoftWareFiles/Security/Linux6.12.49 Distribution V1.0.0/MYD-JMX95X Security Practice Guide-V1.0.pdf §5 Secure Storage` (§5.2 REE-FS, §5.3 RPMB-FS, §5.4 dm-Crypt)
- Mechanism: REE-FS / RPMB-FS secure storage plus `dm-crypt` disk encryption
- **Hardware root of trust: an onboard standalone TPM 2.0 chip** (listed in the MYC core board product manual's on-board device list), usable for secure key storage and platform integrity measurement
- **Key binding to the device:** OP-TEE HUK / TA identity derivation plus the `TRUSTED_KEYS_TEE` kernel option; keys are sealed by the TEE
- **SDK path:** the storage backend is selected in `$SDK/sources/meta-myir/meta-myir-security/recipes-security/optee/` (`optee-os-myir.bbappend`, which pulls in `optee-os-rpmb-fs.inc`) — `MYIR_OPTEE_STORAGE` decides the OP-TEE build-time `CFG_REE_FS` / `CFG_RPMB_FS` (`ree` = the default REE-FS, `rpmb` = RPMB-FS) and `MYIR_RPMB_MODE` decides the RPMB access mode (`test` allows direct reads and writes with the `mmc` tool during development; production has to switch to `prod`, which closes direct access). Both variables are set by the machine configuration `$SDK/sources/meta-myir/meta-myir-bsp/conf/machine/myd-jmx95-15x15-lpddr5.conf` through the `:dm-verity` `OVERRIDES` when `ENABLE_DM_VERITY=1`, to `rpmb` and `test`; the same file carries the four full-chain boot switches `ENABLE_DM_VERITY` / `ENABLE_OVERLAYFS` / `ENABLE_ROOTHASH_SIGN` / `ENABLE_FITIMAGE_SIGN`. The disk-encryption kernel options are in `$SDK/sources/myir-imx-linux/arch/arm64/configs/`

myd_jmx95_15x15_defconfig (CONFIG_DM_CRYPT / CONFIG_TRUSTED_KEYS / CONFIG_TRUSTED_KEYS_TEE / CONFIG_TEE_CRYPT0) and are enabled by default in the released kernel source; the encrypted data volume is created at build time by \$SDK/sources/meta-myr/meta-myr-security/classes/myir-encrypted.bbclass (key backend selectable between CAAM / TPM / TEE) together with myir-data-partition.bbclass. The RPMB route takes effect through StandAloneMM (MACHINE_FEATURES stmm) and the meta-myr-security layer is added to bblayers automatically by tools/myir-setup-release.sh when it is detected

- **Mass-production note:** a test key must not be used in production; the production key has to be programmed board by board and read back for the record. **Writing the eMMC RPMB key is irreversible**, so the key management scheme should be fixed before production starts
- **Deliverables: in the package** — 04-Sources/Security-package.tar.gz (the security feature toolkit, used with §5.3 / §5.4 of the guide);
configuration — the machine file, OP-TEE recipe and kernel defconfig above (change the value and rebuild);
build output — the OP-TEE firmware and images carrying the selected storage backend are output to \$SDK/build-*/tmp/deploy/images/myd-jmx95-15x15-lpddr5/ by the build;
on the device — REE-FS ciphertext lands in /var/lib/tee/ (dirf.db holds the metadata and encryption keys of each TA) and RPMB-FS is written to /dev/mmcblk0rpmb through tee-supplciant;
the key programming flow is given in §5.3 of the guide: dd if=/dev/urandom of=/tmp/rpmb_key.bin bs=32 count=1 to generate a random key, mmc rpmb write-block /dev/mmcblk0rpmb 0x01 ... /tmp/rpmb_key.bin to program it (irreversible), then mmc rpmb read-block to read it back and compare

4.4.3 Secure Update / OTA

- Signature verification: the update image is signed and is written only after verification succeeds
- Integrity check: the update package carries a checksum, preventing man-in-the-middle injection
- A/B partitions + rollback: automatic rollback when an update fails; key subsystems can be updated independently
- Support commitment: **at least 5 years** of free security updates; security patches can be pushed independently

MYC-JMX95 implementation

- **Implementation guide:** SoftWareFiles/Security/Linux6.12.49 Distribution V1.0.0/MYD-JMX95X Security Practice Guide-V1.0.pdf §6 Security Upgrade (§6.1 OTA principle, §6.2 creating a secure OTA package, §6.3 OTA upgrade)
- Mechanism: **SWUpdate** signed secure update (the image already integrates the SWUpdate client, IPC and Web components)
- **This platform integrates the SWUpdate path only:** the SDK layer list contains meta-swupdate plus meta-swupdate-imx and no RAUC / Mender; MYIR delivers the device-

side integration and **the update server is built by the customer**

- Signing and integrity: `sw-description.sig` plus a payload SHA-256, verified against the device public key (`public-key-file = /etc/swu_public.pem`)
- Features enabled in the build: `CONFIG_SIGNED_IMAGES` / `CONFIG_ENCRYPTED_IMAGES` / `CONFIG_HASH_VERIFY` (signature + encryption + checksum), `CONFIG_DOWNLOAD` with `CHANNEL_CURL` (HTTPS fetch), `CONFIG_SURICATTA` (can pair with a Hawkbit-style management platform), `CONFIG_WEBSERVER` with `MongooseSSL` (local Web/IPC delivery) and the Lua handler
- **A/B slots**: dualA / dualB slot selection with `bootlimit=3` and automatic slot switch on failure; version compatibility is checked in software by `hwcompat` and `no-reinstalling` (not a hardware anti-rollback counter)
- **SDK paths**: the device-side components and configuration are in `$SDK/sources/meta-myr/meta-myr-bsp/recipes-myr/swupdate/swupdate/` (`defconfig / swupdate.cfg`), the image integration is in `$SDK/sources/meta-myr/meta-myr-sdk/recipes-fsl/images/myir-image-multimedia.bb`, and the A/B slots and `bootlimit` are in `$SDK/sources/myir-imx-uboot/board/myir/myd_jmx95/myd_jmx95.env`
- **Deliverables: in the package** — `swupdate-scripts.tar.gz` inside `04-Sources/Security-package.tar.gz` (the packaging script `swu_update_image_build.sh` with `boards/cfg_boards.cfg`, `cfg_myir_update_image.cfg` and `sw-description-myr-emmc-dualcopy-image.template`, plus `emmc_bootpart.sh` run during the update);
configuration — the `swupdate.cfg` / `defconfig` files and the U-Boot slot environment file above;
build output — the signed `*.swu` update package created with `./swu_update_image_build.sh -e -s ./priv.pem -b myir -g` as described in §6.2 of the guide;
on the device — the pre-built `swupdate` client with the Web / IPC components ships in the image, with the configuration at `/etc/swupdate.cfg` and the verification public key at `/etc/swu_public.pem`; the update server is built by the customer

4.4.4 Access Control (Mandatory Access Control / AppArmor)

- Mechanism: Linux LSM mandatory access control (AppArmor) that grants access per process profile, **entry by entry — anything not listed is denied**
- Complements DAC: even when a service runs as root, a compromised service can only touch the resources listed in its profile (permissions travel with the program, not with the file)
- Policy persistence: profiles live in `/etc/apparmor.d/` and are loaded automatically at boot by `apparmor.service`; a **path-attached** profile confines plain execution with no extra interaction
- Two modes: **enforce** (blocking) and **complain** (log only, for pre-production tuning)
- Auditing: denials are written to the systemd journal (query with `journalctl`)

This addresses the use control (UC) and identification and authentication control (IAC) requirements of IEC 62443-4-2, as well as the CRA access-control and least-privilege expectations.

MYC-JMX95 implementation

- **Implementation guide:** Chapter 4.4.4 of this document (mechanism, modes and persistence); the configuration and commands below are reproducible on the board
- **Kernel:** CONFIG_SECURITY_APPARMOR=y plus an explicit CONFIG_LSM init order, merged through DELTA_KERNEL_DEFCONFIG with apparmor.cfg (the linux-imx do_copy_defconfig step discards the KERNEL_FEATURES route)
- **User space:** apparmor 4.0.3 (apparmor_parser, aa-status, aa-enforce|aa-complain) plus around 150 profiles under /etc/apparmor.d/; apparmor.service loads them at boot
- **Enabling:** DISTRO_FEATURES:append = " apparmor" in conf/local.conf (systemd integration pulls the apparmor package into the rootfs through its shared-library dependency)
- **SDK-relative path** (source tree): 04-Sources → myd-jmx95x-yocto/sources/meta-myr/meta-myr-bsp/recipes-kernel/linux/linux-imx/apparmor.cfg + linux-imx_%.bbappend
- **Measured** (2026-09-22, myd-jmx95-15x15-lpddr5 / kernel 6.12.49): /sys/kernel/security/lsm = capability, apparmor; aa-status reports 160 profiles loaded, 64 in enforce mode; a path-attached profile blocks **plain execution**, keeps blocking **after a reboot**, and **root cannot escape it**
- **Note:** the apparmor package currently arrives indirectly through the systemd shared-library dependency (it is not listed explicitly in the image recipe); switching to sysvinit requires adding it explicitly. Some upstream profiles do not match this image's services and should be tuned in complain mode before being switched to enforce
- **Deliverables:** the kernel apparmor.cfg configuration and bbappend patch, the pre-built apparmor 4.0.3 user-space packages and the /etc/apparmor.d/ policy set

4.5 Platform SBOM Support

For the platform in use, MYIR runs a complete SBOM generation and delivery process that feeds directly into the customer's CRA compliance filing; the process does not depend on the SoC vendor:

1. **Format:** **SPDX (machine-readable JSON, *.spdx.json)**, satisfying the CRA requirement for a machine-readable SBOM; **CycloneDX** output is available on request
2. **Generation:** produced automatically by Yocto create-spdx during the BSP build, written component by component by the build system, with no manual compilation and always in step with the firmware image
3. **Coverage:** issued layer by layer — whole image → rootfs → each recipe → each package; including **U-Boot, the Linux kernel (with modules and device trees), TF-A, OP-TEE and BSP packages**, plus all user-space libraries and middleware in the rootfs, covering the in-house and third-party components the CRA looks for
4. **Fields:** component name, version, supplier, licence, file checksums, the mapping between sources and binaries and the relationships between components, all recorded per the SPDX specification
5. **Delivery:** each SBOM corresponds one-to-one to a firmware version and is issued together with every image and security patch release; customers may also request the SBOM of any

specific version, and MYIR keeps providing it throughout the support period

6. **Link to vulnerability status:** the SBOM itself is a component inventory and **carries no CVE / VEX fields**; vulnerability status is issued separately in MYIR's **CVE scan report and VEX report**, matched item by item against the SBOM components so that the customer can merge them directly into its own compliance file

The platform SBOM is a technical input to the customer's CRA compliance process; it does not by itself constitute final-product compliance, nor does it replace the assessment and documentation obligations of the final-product manufacturer.

MYC-JMX95 implementation

- **Implementation guide:** Chapter 4.5 of this document (complete description of the SBOM format, generation, coverage and delivery)
- Format and location: `create-spxd` is enabled by default; deliverables in `build-jmx95x/tmp/deplo/spxd/3.0.1/`, format **SPDX 3.0.1 (machine-readable JSON, *.spxd.json)**
- Coverage: image level `myir-image-...-image.spxd.json`, rootfs level `myir-image-...-rootfs.spxd.json`; 48 recipes (including `u-boot-imx`, `linux-imx 6.12.49`, `imx-atf`, `optee-os`, `jailhouse-imx`) and 1089 packages (including `kernel-image` and all `kernel-module-*`)
- The release package also provides `04-Sources/sbom.tar.gz` (listed as "Software list" in the SDK Release Note §2.2)

4.6 Customer Responsibilities and Incident Reporting

- The CRA imposes incident reporting obligations on products placed on the EU market.
- As the placer of the final product, the customer is responsible for reporting exploited vulnerabilities and serious security incidents to the relevant EU bodies (ENISA and the national CSIRT).
- As the component and module supplier, MYIR provides SBOMs, VEX reports, security advisories and vulnerability response capabilities to support the customer's compliance filing.

4.7 Platform CRA Capability Index (MYC-JMX95)

The table below gives, for each CRA capability: what MYIR provides, which document and section the customer follows to complete the implementation, and the **relative paths of the deliverables inside the SDK release package**. Paths start at the package root: 01-Docs (documents), 02-Images (pre-built images), 03-Tools (tools), 04-Sources (sources); a platform security directory is written out in full. \$SDK stands for the platform project root unpacked under 04-Sources; "build output" means the directory produced after building as documented; paths beginning with / are on-device (root filesystem) paths.

CRA capability	What MYIR provides	Implementation guide	Deliverables / SDK path
Secure Boot (chain of trust)	NXP AHAB chain of trust: ELE hardware root of trust + AHAB	<i>MYD-JMX95X Security Practice Guide</i> §1-§4 (SoftwareFiles/Security/Linux6.12.49	In the package: the Practice Guide (EN and ZH) under SoftwareFiles/Security/Linux6.12.49 Distribution V1.0.0/ plus

	signatures on the imx-boot and kernel containers; the full link adds fitImage signing / dm-verity / overlays; eFuse SRK hash and OEM Closed configuration	Distribution V1.0.0/); Chapter 4.4.1 of this document	04-Sources/Security-package.tar.gz (CST signing and packaging toolkit) and the secure-boot burn image myir-image-burn-... rootfs.wic.zst under 02-Images/; class and machine configuration: \$SDK/sources/meta-myir/meta-myir-security/classes/ (myir-boot-signed.bbclass / myir-ahab.bbclass / myir-kernel-container-sign.bbclass), \$SDK/sources/meta-myir/meta-myir-bsp/conf/machine/myd-jmx95-15x15-lpddr5.conf; build output: imx-boot-signed, os_cntr_signed*.bin, SRK_1_2_3_4_fuse.bin and fuse-cmds.txt under \$SDK/build-*/tmp/deploy/images/myd-jmx95-15x15-lpddr5/
Secure execution environment (TEE / OP-TEE)	OP-TEE (TrustZone secure world), provided in the default image	<i>MYD-JMX95X Security Practice Guide</i> §2; Chapter 4.4.2 of this document	Pre-built OP-TEE components ship in the image; optee-os sources and recipes: \$SDK/sources/meta-myir/meta-myir-security/recipes-security/optee/; build output: the OP-TEE firmware tee-raw.bin is packed into the imx-boot container by imx-mkimage (Guide §4)
Secure storage and encryption of keys and sensitive data (hardware root of trust + hardware crypto)	REE-FS / RPMB-FS secure storage + dm-crypt disk encryption + onboard standalone TPM 2.0 chip + i.MX95 EdgeLock / SECO and CAAM hardware crypto	<i>MYD-JMX95X Security Practice Guide</i> §5 (§5.2 REE-FS, §5.3 RPMB-FS, §5.4 dm-Crypt); Chapter 4.4.2 of this document	Secure-storage recipe: \$SDK/sources/meta-myir/meta-myir-security/recipes-security/optee/optee-os-myir.bbappend (pulling in optee-os-rpmb-fs.inc); enabling configuration: MYIR_OPTEE_STORAGE / MYIR_RPMB_MODE in \$SDK/sources/meta-myir/meta-myir-bsp/conf/machine/myd-jmx95-15x15-lpddr5.conf, encrypted-volume template meta-myir-security/wic/myd-jmx95-15x15-lpddr5-enc.wks.in; in the package: 04-Sources/Security-package.tar.gz; the board-by-board key

			programming and read-back steps are in Guide §5.3; on the device REE-FS writes to <code>/var/lib/tee/</code> and RPMB-FS to <code>/dev/mmcblk0rpmb</code> ; the TPM driver and <code>tpm2-*</code> pre-built components ship in the image
Device identity / key binding	Onboard EEPROM holding P/N and S/N (eeprom-pnsn) + OP-TEE HUK / TA identity derivation + the TRUSTED_KEYS_TEE kernel option: keys are sealed by the TEE and bound to the device	<i>MYD-JMX95X Security Practice Guide</i> §5; Chapter 4.4.2 of this document	Sources and pre-built package: <code>\$SDK/sources/meta-myr/meta-myr-bsp/recipes-myr/eeprom-pnsn/</code> ; the optee-os recipes and kernel configuration come with <code>\$SDK/sources/meta-myr/</code>

Secure update (OTA)	SWUpdate signed + encrypted + checksummed update, HTTPS fetch, A/B slots with automatic bootlimit failover, Suricata for a management platform	<i>MYD-JMX95X Security Practice Guide</i> §6 (§6.1 principle, §6.2 creating a secure OTA package, §6.3 upgrade); Chapter 4.4.3 of this document	In the package: swupdate-scripts.tar.gz inside Security-package.tar.gz (swu_update_image_build.sh with boards/cfg_boards.cfg, cfg_myir_update_image.cfg and sw-description-myir-emmc-dualcopy-image.template); configuration: \$SDK/sources/meta-myir/meta-myir-bsp/recipes-myir/swupdate/swupdate/ (defconfig, swupdate.cfg); image recipe: \$SDK/sources/meta-myir/meta-myir-sdk/recipes-fsl/images/myir-image-multimedia.bb; slots and bootlimit: \$SDK/sources/myir-imx-uboot/board/myir/myd_jmx95/myd_jmx95.env; build output: the signed *.swu package created with Guide §6.2; on the device /etc/swupdate.cfg and /etc/swu_public.pem (the update server is built by the customer)
Access control (MAC)	Enabled and measured on this platform: AppArmor LSM (CONFIG_SECURITY_APPARMOR=y) + apparmor 4.0.3 policy set	Chapter 4.4.4 of this document (the configuration and commands are reproducible on the board)	Kernel patch and configuration: \$SDK/sources/meta-myir/meta-myir-bsp/recipes-kernel/linux/linux-imx/apparmor.cfg + linux-imx_%.bbappend; enabling switch: DISTRO_FEATURES:append = " apparmor" in \$SDK/conf/local.conf; on-device policy set /etc/apparmor.d/ (about 150 profiles)
SBOM (supply-chain transparency)	Yocto create-spdx produces SPDX 3.0.1 machine-readable JSON automatically; CycloneDX conversion on request	Chapter 4.5 of this document	Build output: image-level and rootfs-level *.spdx.json under \$SDK/build-jmx95x/tmp/deploy/spdx/3.0.1/; in the package: 04-Sources/sbom.tar.gz

Vulnerability handling (CVE / VEX / PSIRT)	MYIR PSIRT receiving, assessment and disclosure process; CVE scanning and VEX reports mapped component by component to the SBOM	Chapter 2 of this document (the PSIRT page is given there)	Security advisories, VEX reports and security patch packages: delivered through MYIR's release channel (download link) with every version release / patch package, matching the version recorded in the SDK Release Note under 01-Docs /
--	---	--	--

5 Maintenance Strategy

5.1 Support Period

- MYIR's standard commercial policy provides **at least five years** of free security updates and fixes for products covered by that policy.
- **The CRA support period is product-specific:** it must reflect the expected use of the product, reasonable user expectations, the nature and intended purpose of the product and other applicable requirements. The five-year commercial commitment should therefore not be read as a universal CRA support-period determination.
- As required by the CRA, MYIR states the applicable support-period end date in the technical documentation (to at least month precision) and informs customers accordingly.

5.2 Update Method

- Security patches are delivered as **version releases or standalone patch packages**, downloadable from MYIR's release channel.
- Security updates are separated from feature updates and can be released independently, so that customers can schedule deployment in their own maintenance window.
- Under **Article 13(9) of Regulation (EU) 2024/2847**, each security update made available during the applicable support period must remain available after issuance **for at least 10 years**, or for the remainder of the support period, whichever is longer; customers should incorporate this into their product lifecycle and documentation planning.

5.3 Version Policy

- Security fixes are applied continuously to the latest BSP version and backported to affected versions still in support, subject to technical feasibility and component availability.
- For versions that have reached end of support, upgrade and migration guidance is provided.

5.4 Continued Vulnerability Response

- While a version is in support, the PSIRT receiving, assessment and disclosure process remains in place.
- Security advisories and VEX reports continue to be shared with customers.

5.5 Long-Term Support

- MYIR tracks the evolution of the CRA, IEC 62443 and related standards.
- The security capability and compliance documentation set (security architecture document, TARA, CVD policy, declaration templates) is extended progressively.

This is the platform-specific edition. Every capability is mapped to the document section to follow, the deliverables provided and the SDK paths; MYIR technical support assists with implementation on request.