



THREAT ANALYSIS & RISK ASSESSMENT

Threat Analysis and Risk Assessment

ISO/SAE 21434 cybersecurity threat analysis for the open-source
OpenVVF traction inverter / VCU.

Supported by



MCUS:
STM32H723ZG + STM32G474RCTx
Operating Temp: -40 °C to +85 °C

1. Scope and Philosophy

This Threat Analysis and Risk Assessment (TARA) covers the cybersecurity aspects of an open-source aftermarket electric motorcycle traction inverter and vehicle control unit (VCU). It follows the methodology of **ISO/SAE 21434** (Road vehicles - Cybersecurity engineering) but adapts it for the realities of open-source hardware.

1.1. The Trust-the-User Model

This project is **open source**. The end user has full access to source code, schematics, firmware, and build tools. This fundamentally changes the cybersecurity posture compared to a proprietary automotive product:

Table 1: Trust Model: Open Source vs. Proprietary

Aspect	Proprietary (OEM)	This Project (Open Source)
Source code access	Restricted to manufacturer	Fully public; user can audit
Firmware signing keys	Manufacturer-controlled, burned into OTP	User-generated, user-controlled
Physical tamper resistance	Security screws, potting, encrypted flash	None by design; user can modify
Attack model	Remote attackers, supply chain, dishonest dealers	CAN bus attackers, user tampering
Remediation authority	Manufacturer issues OTA patches	Community issues patches
Liability	Manufacturer assumes product liability	User assumes full responsibility

DESIGN PRINCIPLE: USER SOVEREIGNTY OVER ANTI-USER DRM

This project explicitly rejects all anti-user security mechanisms: OTP fuses, write-once flash regions, vendor-signed firmware with unreplaceable keys, encrypted bootloaders that prevent user modification, and any form of DRM that treats the hardware owner as an adversary. These mechanisms are incompatible with the open-source philosophy and provide no meaningful security benefit when the attacker already has physical access and the full source code. If a user wishes to add their own tamper protection (RDP Level 1/2, encrypted external flash, physical tamper switches) for their specific threat model, they are free to do so - the base design imposes no barriers and documents how. The security model is

TRUST THE USER, PROTECT THE BUS

: the legitimate owner is never the threat; remote CAN bus attacks are.

1.2. What We Actually Protect Against

With physical access = game over (acceptable for open source), the meaningful attack surfaces are:

1. **CAN bus remote attacks:** A malicious or compromised node on the CAN bus (aftermarket accessory, diagnostic tool, charger) should not be able to cause unintended tractive effort, disable safety features, or flash malicious firmware without authorization.
2. **Firmware integrity during CAN updates:** Bit errors, EMI, or bus faults during a legitimate firmware update should not brick the device or install corrupted safety-critical code.
3. **Replay and downgrade:** An attacker capturing a valid firmware update should not be able to replay it to install older, potentially vulnerable firmware.
4. **Denial of service:** CAN bus flooding should not prevent the VCU from receiving critical safety messages (BMS heartbeat, brake input).

1.3. What We Explicitly Do Not Claim

Table 2: Out-of-Scope Security Claims

Claim	Why Not
Physical tamper resistance	Open hardware; user has PCB, schematic, JTAG access. Physical tamper
Supply chain security	Open BOM; user sources their own components. Component authentic
Side-channel attack resistance	No power analysis countermeasures, no timing randomization. Resear
Firmware extraction protection	Source code is public; extracting firmware from the device provides no
Anti-cloning	Open schematic and Gerbers; anyone can build a copy. This is a featu

2. Asset Identification

Table 3: Cybersecurity-Relevant Assets

ID	Asset	Location
A-01	Application firmware	STM32 internal flash
A-02	Bootloader	STM32 internal flash (separate sector)
A-03	Firmware signing key	User possession of build key; symmetric key compiled into STM32
A-04	Calibration data	STM32 flash (torque LUT, thresholds)
A-05	CAN bus communication	CAN1 (BMS), CAN2 (IO board, display, charger, ABS)
A-06	Runtime state (RAM)	STM32 ECC RAM

2.1. Attack Surface Map

Table 4: Attack Surfaces and Interfaces

Interface	Access	Risk
CAN1 (BMS)	Vehicle-internal, BMS node required	Medium: compromi
CAN2 (IO board, display, charger, ABS)	Vehicle-internal, multiple nodes	Higher: more nodes
SWD/JTAG	Physical access to debug header	Low (physical acce
USB (if present)	Physical access	Low (physical acce
UART console	Physical access or accessible header	Low

3. Threat Scenarios

Threat scenarios follow ISO/SAE 21434 threat analysis methodology: identify threat sources, attack vectors, and resulting damage scenarios.

Table 5: Threat Scenarios

ID	Threat	Source
T-01	Malicious firmware upload via CAN	Compromised accessory node, malicious diagnostic
T-02	Firmware corruption during CAN update	EMI, loose connector, bus fault
T-03	Firmware downgrade (replay attack)	Attacker with CAN bus access and captured update
T-04	CAN bus flooding (DoS)	Compromised node, malicious accessory
T-05	Forged safety-critical CAN frames	Compromised display/charger/IO node
T-06	Firmware update during vehicle motion	Buggy update tool, malicious update trigger
T-07	Bootloader replacement via application	Malicious firmware already running

4. Risk Assessment

Risk rating uses ISO/SAE 21434's impact/likelihood framework adapted for the open-source trust model. **Impact** considers vehicle-level safety consequences. **Likelihood** considers attack feasibility given the open-source context (physical access already assumed = user).

Table 6: Risk Rating Matrix

Threat	Safety Impact	Attack Likelihood	Risk Level	Rationale
T-01: Malicious firmware upload	Severe	Possible	HIGH	CAN bus is
T-02: Firmware corruption	Severe	Likely	HIGH	CAN is inh
T-03: Firmware downgrade	Major	Possible	MEDIUM	Requires a
T-04: CAN bus flooding	Moderate	Possible	MEDIUM	Safe state
T-05: Forged safety frames	Severe	Unlikely	MEDIUM	Requires c
T-06: Update during motion	Major	Unlikely	LOW	Update too
T-07: Bootloader replacement	Severe	Unlikely	LOW	Requires a

5. Cybersecurity Requirements

These cybersecurity requirements (CSRs) are independent of the HARA's Functional Safety Requirements (FSRs). Where a CSR and FSR overlap (e.g., CAN heartbeat timeout), the CSR references the FSR rather than duplicating it.

Table 7: Cybersecurity Requirements

ID	Requirement
CSR-01	CAN firmware update requires cryptographic signature verification. Unsigned or incorrectly signed
CSR-02	CAN firmware update protocol: each chunk carries a 32-bit CRC. Overall firmware image can
CSR-03	Anti-rollback counter stored in flash sector reserved for configuration data. Firmware with cou
CSR-04	Firmware update preconditions: vehicle speed = 0 (encoder < 10 rpm for >2 s), brake applied
CSR-05	CAN bus heartbeat timeout (FSR-17) serves as DoS protection. IO board loss → safe state w
CSR-06	CAN message freshness: safety-critical frames (BMS heartbeat, IO board inputs) include a 32
CSR-07	Bootloader sector is not write-protected by hardware. This is intentional: the user must be abl

WHY HMAC-SHA256 OVER RSA/ECDSA

HMAC-SHA256 is recommended over asymmetric algorithms (RSA, ECDSA, Ed25519) for three reasons: (1)

SMALLER CODE SIZE

in the bootloader - critical for size-constrained bootloaders; (2) **Faster verification** - SHA-256 is hardware-accelerated on STM32H7; (3) **No PKI complexity** - the user generates one key, stores it in their build environment, and signs their own firmware. The threat model does not require protection against key compromise (the user owns the key) so the non-repudiation benefits of asymmetric crypto are irrelevant here.

6. Test Plan

Nine test cases validate the cybersecurity requirements. CT-01 through CT-06 and CT-09 require a CAN interface and the VCU. CT-07 and CT-08 additionally require SWD/JTAG access to the bootloader flash.

6.0.1. CT-01: Unsigned Firmware Update Rejected

Objective: Verify CSR-01 - Unsigned firmware update via CAN is rejected.

Threat: T-01 (malicious firmware upload)

Setup: VCU with CAN update enabled. Bootloader built with known HMAC key.

Procedure:

1. Enter update mode: key-on, brake applied, speed=0.
2. Send firmware chunks over CAN without HMAC signature.
3. Send commit command.
4. Verify bootloader rejects update. Current firmware remains active.

Pass: Rejected. DTC logged: "signature verification failed." No partial flash write.

6.0.2. CT-02: Signed Firmware Update Accepted

Objective: Verify CSR-01 positive path - Signed update accepted.

Threat: T-01 (legitimate update)

Procedure:

1. Sign firmware with correct HMAC key.
2. Send signed chunks with per-chunk CRC (CSR-02).

3. Send SHA-256 hash of complete image.
4. Verify accepted, flash committed, system boots.
5. Verify POST passes. New version active.

Pass: Full update chain succeeds.

6.0.3. CT-03: Corrupted Chunk Detected

Objective: Verify CSR-02 - Bit flip in CAN chunk aborts update.

Threat: T-02 (firmware corruption)

Procedure:

1. Begin signed firmware update.
2. Flip one bit in a chunk's data payload.
3. Verify chunk CRC detects error. Update aborts.
4. Repeat with corrupted CRC field and corrupted sequence number.

Pass: All corruption detected. Update aborted. Partial image erased.

6.0.4. CT-04: Anti-Rollback Counter

Objective: Verify CSR-03 - Downgrade attack rejected.

Threat: T-03 (replay / downgrade)

Procedure:

1. System running firmware with counter = 5.
2. Attempt flash of older firmware with counter = 4. Verify rejected.
3. Flash newer firmware with counter = 6. Verify accepted. Counter = 6.
4. Attempt reflash with counter = 6. Verify rejected (must be > current).
5. Via debugger, reset counter to 0. Verify older firmware now accepted. (*Physical access = user can reset*)

Pass: Counter logic correct. User can reset via debugger (documented behavior).

6.0.5. CT-05: Update Preconditions

Objective: Verify CSR-04 - Update only when stationary.

Threat: T-06 (update during motion)

Procedure:

1. All preconditions met. Verify update accepted.
2. Speed > 0. Verify rejected.
3. Brake = off. Verify rejected.
4. During update, release brake. Verify aborts immediately.

Pass: All precondition violations rejected or aborted.

6.0.6. CT-06: CAN Freshness Counter - Replay Detected

Objective: Verify CSR-06 - Stale CAN frames treated as lost.

Threat: T-05 (forged safety frames)

Procedure:

1. Normal operation with freshness counters active.
2. Replay captured BMS heartbeat from 10 s ago. Verify detected as stale.
3. Verify VCU applies safe defaults (treats as heartbeat lost).
4. Verify DTC logged: "stale CAN frame detected."

Pass: Replay detected. Safe defaults applied. No tractive effort from stale frames.

6.0.7. CT-07: Bootloader Mutable (Design Verification)

Objective: Verify CSR-07 - Bootloader can be updated by user.

Threat: T-07 (bootloader replacement - intentional user action)

Procedure:

1. Via SWD/JTAG, read current bootloader flash sector. Verify readable (RDP Level 0).
2. Via SWD/JTAG, write modified bootloader (e.g., change version string). Verify succeeds.
3. Boot system. Verify modified bootloader runs.
4. Restore original bootloader. Verify system returns to original state.

Pass: Bootloader is mutable via SWD. User has full control. Document this as intentional.

6.0.8. CT-08: HMAC Key Rotation

Objective: Verify CSR-01 - user can rotate firmware signing keys.

Threat: T-01 (key compromise recovery)

Procedure:

1. Build bootloader with HMAC key K1. Flash to device.
2. Sign firmware with K1. Verify accepted.
3. Sign firmware with K2 (different key). Verify rejected.
4. Build and flash new bootloader with HMAC key K2 (via SWD).
5. Sign firmware with K2. Verify accepted.
6. Sign firmware with K1. Verify rejected.

Pass: Key rotation works. Old key rejected after rotation. Documented in user manual.

6.0.9. CT-09: CAN DoS / Heartbeat Timeout

Objective: Verify CSR-05 - CAN bus flooding causes heartbeat loss and safe state entry.

Threat: T-04 (CAN bus flooding / DoS)

Procedure:

1. Establish normal operation with valid IO board and BMS heartbeat traffic.
2. Flood CAN2 with high-priority frames at >80 % bus load.
3. Verify IO board heartbeat timeout triggers safe state within 1 s.
4. Repeat on CAN1 with BMS heartbeat and verify safe state within 5 s.
5. Verify normal operation resumes automatically after the bus flood stops and heartbeats return.

Pass: Safe state is entered within the FSR-17 timeout under bus flood, and the system recovers cleanly when flooding stops.

7. Implementation Guidance

7.1. HMAC Key Management

The user manages their own signing key. The project will provide a build script that:

1. Generates a random 256-bit key if none exists (stored in `keys/fw_signing_key.bin`).
2. Embeds the key into the bootloader binary at build time.
3. Provides a signing tool (`sign_firmware.py`) that takes a compiled firmware binary and produces a signed update package.

The key is never transmitted over CAN. It never leaves the user's build machine. Key compromise means someone gained access to the user's build environment, at which point they can already build and flash arbitrary firmware (physical access or remote access to build machine).

7.2. CAN Update Protocol

Recommended protocol (kept simple for open-source implementability):

Table 8: CAN Update Frame Format

Frame Type	CAN ID	Payload	Description
INIT	0x7F0	[0x01][version_major][version_minor][size_kb:2B]	Start update session
CHUNK	0x7F1	[seq:2B][data:0-6B]	Firmware data chunk
CHUNK_ACK	0x7F2	[seq:2B][crc32:4B]	Chunk received, CRC
HASH	0x7F3	[sha256_first_8B]	Image hash (send 4
SIGN	0x7F4	[hmac_first_8B]	HMAC signature (se
COMMIT	0x7F5	[0x02]	Verify and commit
STATUS	0x7F6	[status_code][progress_pct]	Bootloader response
ABORT	0x7F7	[error_code]	Error / user abort

7.3. Enabling RDP for Security-Conscious Users

For users who want additional protection, document (but do not mandate) how to enable STM32 Read Protection:

OPTIONAL: STM32 RDP LEVEL 1

RDP Level 1 prevents external debuggers from reading flash memory via SWD/JTAG. The bootloader and application remain fully functional. Flash can still be mass-erased (which also erases the key).

To enable: ST-Link Utility → Target → Option Bytes → RDP = Level 1

WARNING:

RDP Level 2 is irreversible and disables debug permanently. Do not recommend Level 2 in documentation; mention it exists but strongly warn against it for a development-friendly open-source project.

8. Gap Analysis

Table 9: Cybersecurity Gaps

Gap	Risk	Mitigation
No HMAC signature verification yet	HIGH	Add HMAC-SHA256 to bootloader; provide sig
No CAN update integrity protocol yet	HIGH	Define chunk format + CRC + SHA-256 hash
No anti-rollback counter yet	MEDIUM	Add 64-bit counter to config flash page
No freshness counter on CAN frames yet	MEDIUM	Add 32-bit rolling counter to BMS/IO board he

8.1. HARA Cross-Reference

This TARA is a companion document to the HARA. Safety-relevant cybersecurity threats (those that could cause unintended tractive effort or prevent safe state entry) are linked to HARA hazards:

Table 10: TARA-to-HARA Cross-Reference

TARA Threat	HARA Hazard	Shared M
T-01 (malicious firmware upload)	H-01 (unintended tractive effort), H-15 (software error)	CSR-01 (
T-02 (firmware corruption)	H-15 (software error), H-14 (latched tractive effort)	CSR-02 (
T-04 (CAN DoS)	H-03 (loss of tractive effort)	CSR-05 r
T-05 (forged CAN frames)	H-01 (unintended tractive effort)	CSR-06 (

i

Note on Separation of Concerns

The HARA addresses **RANDOM HARDWARE FAILURES** and **systematic design faults** that cause hazardous events. This TARA addresses **intentional malicious actions** that cause the same hazardous events. The requirements are separate (FSRs vs. CSRs) because they are verified against different fault models (random fault injection vs. adversarial attack simulation), but they share the same safe-state destination: SSO when anything goes wrong.

9. References

Table 11: Referenced Standards and Documents

Reference	Title / Description
ISO/SAE 21434:2021	<i>Road vehicles - Cybersecurity engineering</i> . Defines threat
SAE J3061	<i>Cybersecurity Guidebook for Cyber-Physical Vehicle Systems</i>
ISO 26262-3:2018	<i>Road vehicles - Functional safety - Concept phase</i> . HARA
FIPS 198-1 / NIST SP 800-107	<i>FIPS 198-1 / NIST SP 800-107</i> - HMAC-SHA256 implementation
STM32H723 Reference Manual (RM0433)	<i>STM32H723/733 - Arm Cortex-M7 MCU</i> . Flash protection (
HARA (this project)	OV-SAF-HARA-CORE - <i>Hazard Analysis and Risk Assessment</i>

9.1. Document History

Table 12: Revision History

Version	Date	Changes
1.0	June 13, 2026	Initial TARA release. 7 threat scenarios (T-01 through T-07), 7 cybersecurity
1.1	July 8, 2026	User-sovereignty security model; 7 threat scenarios (T-01 through T-07);
1.2	July 13, 2026	Editorial consistency pass: restored missing v1.1 revision-history entry; ci
1.3	August 7, 2026	Migrated to Documentation repo as <code>OV-SAF-TARA-INDEX</code> . Added frontr