

Readiness report — evcc Solar EV Charge Controller

Checked 2026-09-03 03:06 · 12 searches, 11 pages fetched

Readiness: evcc Solar EV Charge Controller

Verdict: Ready with caveats · 2026-09-03 · est. parts cost \$588-\$645 **Cybersecurity — software & network: Review before networking**

This is the cold check for this lab — there is no previous report, so it establishes the baseline every future check will measure drift against. The lab itself is software: evcc runs on a Raspberry Pi or Linux host and talks to commercial charging hardware, so there is no PCB, no schematic and no component-level BOM to source. What the snapshot does list is three external assemblies, and two of the three cannot be ordered as written. The Phoenix Contact EM-CP-PP-ETH charge controller (2902802) is **Obsolete** with zero stock at all five authorized distributors checked, and its sibling EV-CC-AC1-M3-CBC-RCM-ETH (1018701) is Obsolete too — Phoenix Contact's whole legacy charge-controller generation is end-of-life. The bare EVSE DIN module has no US retail source. The lab stays implementable because the snapshot allows any IEC 61851-1 DIN-rail controller that evcc supports, and Phoenix Contact's own recommended successor — the CHARX SEC-3000 — is Active, in stock at DigiKey, and listed on evcc's supported-charger page. The caveats are cost (about \$220 more than the obsolete part), thin stock (2 units at the only distributor holding any), and a supply voltage that could not be verified from any source reachable this run.

Action required

Part	Problem	Recommended action
Phoenix Contact 2902802 (EM-CP-PP-ETH)	Lifecycle Obsolete . Zero stock at all five authorized distributors checked (DigiKey, RS, Onlinecomponents.com, Master Electronics, TTI); lead times 42 days to 13 weeks. ECIA lifecycle risk High.	Substitute the Phoenix Contact CHARX SEC-3000 (order no. 1139022), the manufacturer's own recommended successor — see Substitutes. Confirm its supply voltage against the datasheet first.
EVSE DIN Charge Controller	No US retail source. Two Google Shopping searches returned only solar MPPT controllers and complete wallboxes (\$1,058.77-\$3,971.97), never the bare DIN-rail module.	Either import from an EU specialty vendor (EV Racing lists the EVSE DIN RS485 , DIN-rail, RS-485 Modbus at 9600 8N1 as the snapshot requires — no price shown on the page), or substitute the CHARX SEC-3000 and drive it over Modbus/TCP instead of RS-485.
Phoenix Contact EV-CC-AC1-M3-CBC-RCM-ETH (1018701)	Named in the snapshot as the alternative example for the same BOM line. Also Obsolete , \$519.86, no stock; 0 of 3 authorized distributors hold any.	Same successor applies. Do not order this as a way around the 2902802 shortage.

Cybersecurity — software & network

Review before networking — The pinned golang.org/x/crypto has advisories with an available upgrade, the lab's copy of the frontend ships no lockfile, and evcc opens eight network ports with the container running as root — all manageable, but they should be settled before the host joins a shared network.

Component	Version	How known	Support status
evcc (application)	not determined	not determined (No release or version string is pinned in go.mod, package.json, package-lock.json or the Dockerfile fetched this run.)	—
Go toolchain	1.27.0	read from project (go directive in fetched/01-go.mod; the build stage of fetched/04-Dockerfile uses golang:1.27.0-alpine.)	supported
Node.js runtime	26.0.0	inferred (Lower bound of engines.node ">=26.0.0" in fetched/02-package.json; the project pins no exact runtime, so the minimum it allows was checked.)	supported until 2029-04-30
npm package manager	not determined	unpinned (engines.npm and devEngines.packageManager in fetched/02-package.json give a lower bound ">=10.0.0", not a version.)	—
Alpine Linux (container base image)	3.23	read from project (Runtime stage FROM alpine: 3.23@sha256:5b10f432... in fetched/04-Dockerfile.)	supported until 2027-11-01
golang.org/x/crypto	v0.55.0	read from project (Pinned in fetched/01-go.mod.)	—
github.com/eclipse/paho.mqtt.golang (MQTT client)	v1.5.1	read from project (Pinned in fetched/01-go.mod.)	—
github.com/lorenzodonini/ocpp-go (OCPP central system)	v0.19.0	read from project (Pinned in fetched/01-go.mod; go.mod redirects it to github.com/evcc-io/ocpp-go via a replace directive.)	—

Component	Version	How known	Support status
github.com/andig/mbserver (Modbus TCP server)	v0.0.0-20230310211055-1d29cbb5820e	read from project (Pinned in fetched/01-go.mod.)	—
Vue (web UI framework)	3.5.41	read from project (Resolved in fetched/03-package-lock.json; package.json declares the range ^3.5.41.)	—
@voidzero-dev/vite-plus-core (frontend build toolchain)	0.2.6	read from project (fetched/02-package.json declares devDependencies.vite and overrides.vite as npm:@voidzero-dev/vite-plus-core@0.2.6; fetched/03-package-lock.json resolves node_modules/vite to the vite-plus-core tarball.)	—

Component	Finding	Severity	Fix available?
golang.org/x/crypto v0.55.0	The version of golang.org/x/crypto that evcc pins has two denial-of-service flaws in its SSH sub-package, which would let a peer that can open an SSH connection to the affected code hang it; whether evcc's build actually uses that sub-package was not established here. — GO-2026-6354 , GO-2026-6355	moderate	Yes — update to 0.56.0
golang.org/x/crypto v0.55.0 (openpgp)	The OpenPGP sub-package shipped inside the pinned golang.org/x/crypto is unmaintained and flagged unsafe by design, and no release of the library fixes it. — GO-2026-5932	low	Mitigation only — no fixed release exists; nothing in the lab's documented workflow uses OpenPGP, so treat it as a reason to keep the host off open networks rather than a blocker.
evcc container image	The runtime stage of evcc's Dockerfile declares no non-root USER, so the published container runs all of those network services as root.	posture	Mitigation only — run the container as a non-root user, or use the packaged Linux install instead, which creates a dedicated evcc account during installation.

Component	Finding	Severity	Fix available?
evcc network services	evcc's own container image declares eight listening ports on the local network — the web UI and REST API on 7070/tcp, OCPP on 8887/tcp, EEBus on 4712/tcp, Modbus on 8899/udp, and discovery on 5353, 7090, 28376 and 9522/udp — so a host running evcc is reachable on far more than a web page.	posture	Mitigation only — put the charger host on its own VLAN or lab subnet, expose only 7070/tcp to the classroom, and confirm the arrangement with campus IT before connecting.
Frontend dependencies (package.json)	dependencies unpinned: 58 of 60 in package.json — the lab's copy shipped no package-lock.json, so building the web UI from the uploaded materials with npm install resolves those caret ranges to whatever is current instead of to the set checked here.	posture	Mitigation only — build with the upstream package-lock.json using npm ci rather than npm install, so the versions checked in this report are the versions installed.

What this means for your lab

evcc is a network service, not a desktop program: the host you install it on starts listening on the classroom network as soon as it runs. Its own container image opens the web interface and REST API on port 7070, an OCPP endpoint on 8887 for wallboxes to connect back to, an EEBus port on 4712, Modbus on 8899, and several discovery ports. evcc's install guide has you set an administrator password the first time you open the web interface, and you should make sure that actually happened on every student host — but the password protects the web interface, not the other protocol ports, and the published container runs everything as root. In practice that means someone else on the same network who can reach the machine may be able to talk to the charging and metering interfaces directly, and if a student builds the web UI from the lab's own copy of the materials they will pull whatever dependency versions npm resolves that day rather than the set checked here. Nothing found in this check is being exploited by anyone at a distance today, and the platform pieces (Go 1.27, Node.js 26, Alpine 3.23) are all still supported. The sensible handling is to give the charger host its own VLAN or lab subnet, expose only the web interface to the classroom, set and record the administrator password, build the UI with npm ci against the project's lockfile, and talk to campus IT before putting any of it on the main network.

Not checked

- vite 0.2.6 (npm alias row in package-lock.json): FALSE POSITIVE, dismissed: this lockfile row is an npm alias. package.json declares vite as npm:@voidzero-dev/vite-plus-core@0.2.6 and the lockfile resolves node_modules/vite to the vite-plus-core tarball, so the 16 advisories OSV returned for the real 'vite' package at 0.2.6 do not apply to anything installed. The package actually installed, @voidzero-dev/vite-plus-core 0.2.6, was checked this run and has no known advisories. It is also a build-time tool that never runs on the charger host.

- Reachability of the golang.org/x/crypto advisories: The two SSH flaws and the OpenPGP warning are in sub-packages. The lab's Go source files were not fetched, so whether evcc imports golang.org/x/crypto/ssh or [/openpgp](https://golang.org/x/crypto/openpgp) at all was not determined.
- evcc application release version: No release or version string is pinned in any file fetched this run, so the application itself has no version to check against an advisory database.
- npm package manager: `engines.npm` gives the lower bound "`>=10.0.0`", not a version, and `endoflife.date` publishes no npm product, so no support status could be computed.
- Go tool-block entries (`enumer`, `gok`, `mockgen`, `modernize`, `cmd/implement`): The `go.mod` tool directives carry no version tags, as the snapshot records, so there is no version to look up.
- Web UI authentication as actually deployed: evcc's own install guide instructs the operator to set an administrator password at first start, but whether any given classroom host has one set cannot be checked from the project files.
- 320 further dependencies in `package-lock.json`: beyond the 500-dependency automatic check cap (e.g. `ansi-regex`, `ansi-styles`, `ansi-styles`, `argparse`, `aria-query`)

Parts

Hardware Bill of Materials

Part	Spec/Package	Status	Stock	Price	Source
Phoenix Contact 2902802 (EM-CP-PP-ETH)	DIN-rail AC charge controller; 230 V AC supply; IEC 61851-1 CP/PP; Modbus RTU/TCP. Qty 1	Obsolete — discontinued	0 at all 5 distributors checked (13-week factory lead at DigiKey)	\$342.34	DigiKey (Obsolete); ECIA: RS \$411.21, Onlinecomponents.com / Master Electronics \$350.78, TTI out of stock
EVSE DIN Charge Controller	DIN-rail module; 230 V AC or 12 V DC; IEC 61851-1 CP/PP; RS-485 Modbus. Qty 1	Hard to obtain — no US retail listing	—	—	Product confirmed at EV Racing (CZ) ; no price in the fetched page. Supported by evcc as "EVSE DIN"
USB to RS-485 / serial adapter	External dongle or DIN-rail converter; 5 V USB power; differential A/B. Any FTDI, CH340 or CP2102 part. Qty 1 (optional)	Readily available	95 at DigiKey; 125 at RS	\$28.30	DigiKey, FTDI USB-COM485-PLUS1 (Active). Alternative: USB-RS485-WE-1800-BT \$33.70 (Active; 1,008 at Mouser). Retail band \$20.39–\$35.49

The snapshot names EV-CC-AC1-M3-CBC-RCM-ETH as an alternative for the first line; it resolves to Phoenix Contact [1018701](#), [\\$519.86](#) and is also Obsolete with no stock, so it is not a way out.

Substitutes

For part	Substitute	Differences	Verified	Judgment
Phoenix Contact 2902802 (EM-CP-PP-ETH)	Phoenix Contact CHARX SEC-3000, order no. 1139022 — \$560.00, 2 in stock (also \$616.40 at RS Supply , "Normally In Stock")	Same class and same manufacturer, and DigiKey returns it as the "MFR Recommended" substitute for both obsolete parts. DIN-rail mount: matches. IEC 61851-1: matches. Communication: Modbus/TCP (plus OCPP 1.6J and MQTT) rather than Modbus RTU over RS-485 — the snapshot allows either, but this is an Ethernet device, so the USB-RS485 adapter is not used with it. Supply voltage: unknown, so it may not drop onto the same feed as the 230 V AC original. Cost: +\$218 to +\$274. Stock: 2 units, every other authorized distributor on 4–5 week lead.	Verified: DIN-rail mounting, IEC 61851-1 compliance and Modbus/TCP support against a fetched RS Supply product page; evcc support against evcc's fetched supported-charger list , which carries CHARX under Phoenix Contact; lifecycle, price and stock against DigiKey and ECIA. Not verified: supply voltage / operating rail (DigiKey publishes no parametrics for this part, the manufacturer page returned HTTP 403, and no text could be extracted from the vendor datasheet PDF); terminal layout and pinout; physical width.	Acceptable with caveats — right device class, manufacturer's own successor, evcc-supported and in stock, but more expensive, thin stock, and the supply rail is an open question.
EVSE DIN Charge Controller	Phoenix Contact CHARX SEC-3000, order no. 1139022 — \$560.00	Replaces an RS-485 Modbus RTU device with a Modbus/TCP one, so the wiring exercise changes from a serial fieldbus to Ethernet and the USB-RS485 adapter is no longer needed. Roughly four times the cost of a bare EVSE DIN module. Same DIN-rail form and same IEC 61851-1 role.	As above. Not verified: supply voltage.	Acceptable with caveats, and the only substitute priced and stocked in the US this run. The snapshot also permits openWB and Bender CC612/CC613, both confirmed on evcc's fetched charger list, but neither was priced or stock-checked here.

An EVSE controller that surfaced in the shopping search — the OpenEVSE v5.5 at \$149.00 — is **not** offered as a substitute: "OpenEVSE" does not appear anywhere on evcc's fetched

supported-charger list, so the snapshot's "supported by evcc" condition is unmet on the evidence gathered.

Verify any substitute against its own datasheet before ordering — especially the CHARX SEC-3000's supply voltage, which this check could not confirm.

Other dependencies

- **Upstream GitHub repository** — <https://github.com/evcc-io/evcc> — live. Four project files were fetched from it successfully during this check.
- **Documentation portal** — <https://docs.evcc.io/en/> — live. Two pages were fetched successfully (the supported-charger index and the EVSE DIN charger page).
- **Service status dashboard** — <https://status.evcc.io/> — not checked.
- **Package and binary repository** — <https://cloudsmith.io/~evcc/packages/> — not checked.
- **Translation platform** — <https://hosted.weblate.org/engage/evcc/> — not checked.
- **Container registry** — Docker Hub, evcc/evcc — not checked.
- **Third-party vehicle telematics services** — Tesla Fleet API, Nissan Carwings, EcoFlow, Tapo — not checked. These are vendor cloud APIs subject to deprecation and authentication changes; a distributor pass cannot verify them.
- **Push messaging gateways** — Telegram Bot API, Pushover, Shoutrrr providers — not checked.
- **Fabrication** — not applicable. The snapshot records no PCB, Gerber, drill or board mechanical files.

Watch list

- **Phoenix Contact CHARX SEC-3000 (1139022) stock.** DigiKey held 2 units at the time of this check and no other authorized distributor held any. This is the lab's only priced, in-stock controller path, and 2 units is one classroom order away from empty.
- **The CHARX SEC-3000's supply voltage.** Still unverified. If it turns out to need a supply the lab's bench cannot provide, the substitution needs rethinking, and that would change the verdict.
- **Phoenix Contact 2902802 and 1018701.** Both Obsolete with zero channel stock; expect the remaining broker listings and prices to keep drifting upward. Neither should return.
- **The EVSE DIN module's US availability.** Absent from US retail across two searches. If a US distributor picks it up, the cheapest path back to the snapshot's original design reopens.

- **USB-RS485 adapters.** No concern — Active, deep stock at several distributors, and the snapshot allows any FTDI, CH340 or CP2102 part. Included only as a baseline for the next check.