Chasing a moving target

Maintaining Vaultwarden in 2026: client drift, LLM floods and static Rust builds

Mathijs van Veluw (BlackDex) · Vaultwarden maintainer · NixOS Meetup

Vaultwarden in one slide

67k

GitHub stars

Rust

single (static) binary

AGPLv3

since 1.28.0

It is

- A Bitwarden-compatible server, reimplemented in Rust
- Runs on SQLite, MySQL/MariaDB or PostgreSQL
- Ships the official web-vault with a minimal patch set
- Tiny footprint: fits on a Raspberry Pi or a €5 VPS (or on your Android)
- Unlocks most Bitwarden premium/org features for self-hosters

It is not

- Bitwarden. Not affiliated, not supported by them
- In control of any client: browser, desktop, mobile, CLI
- Feature-complete: we implement what the clients need
- Professionally audited (community reviewed, advisories fixed)
- A large team: a handful of maintainers, mostly volunteers

Three stories

What actually takes a maintainer's time in 2026

1. **Chasing a moving target**

   Bitwarden ships clients monthly. July and August 2026 broke us twice

2. **The LLM flood**

   More PRs, more reports, same few findings

3. **Building a (static) Rust binary**

   Cross-compiling for 4 musl targets and dependencies

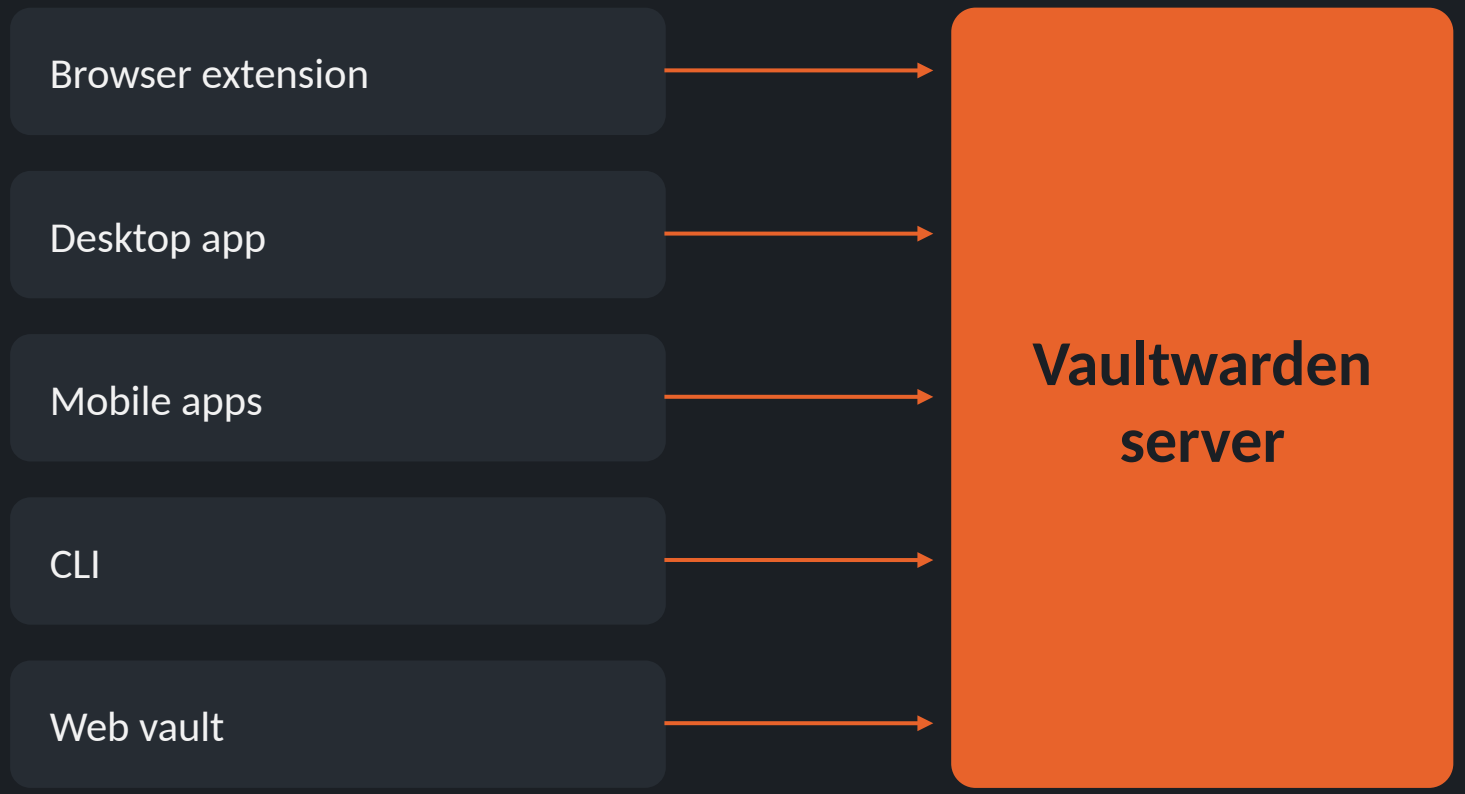
STORY 1

Chasing a moving target

We do not own the clients, Bitwarden does

The compatibility contract

Every client speaks to us as if we were the Bitwarden cloud/self-hosted



What moves under us

- Monthly client releases (2026.x.y)
- Shared Rust SDK compiled to WASM: crypto and data models live client-side
- Change without an API version bump
- Clients cache aggressively: stale state survives server fixes

Summer 2026, twice

Two client releases, two required server releases

Jul 2026

Clients 2026.7.0+

- Client SDK broke on old compatibility JSON fields
- Result: vault shows empty, sync 'works'
- Fix: 1.37.0, required for 2026.7.0+

Aug 2026

Clients 2026.8.0+

- Organization Policy response needed `revisionDate`
- Result: empty vault, 'invalid master password' loops
- Fix: 1.37.2, required for 2026.8.0+

After

The cache trap

- Server fixed, client still broken
- Users must log out fully or wipe the client’s local storage
- Half the issue traffic was this

How we find out, and how we ship

There is no heads-up from upstream. We watch, diff and test.

Detect

- Watch client and SDK releases, not just the server repo
- Diff the Bitwarden OpenAPI spec between versions
- Run dev clients against the :testing image
- Issue templates that force the client version and diagnostics string

Ship

- Fixes land on :testing first, then a release
- Release notes mention supported clients versions if needed
- Signed tags, attested images

Lesson for operators

- A client update could be a server upgrade trigger
- Pin the clients or delay updates if you do not love surprises
- Upgrade the server quickly if possible, else live on the edge and use `:testing`
  (It helps us too)
- When in doubt: log out, clear client data, log in

“Will Bitwarden break you on purpose?”

The question behind every issue in this category

Why people ask

- Bitwarden is pushing harder on money
- More of the new stuff lands behind a paywall, or behind a license we can't touch
- When things break, it's inside the SDK without any API versioning; Bitwarden does not need that
- It's a commercial company's client talking to a volunteer's server

Honest answer

- So far: breakage has been side effects of SDK work, not targeting
- The real risk is speed: Bitwarden keeps developing with a large team
- What would actually hurt: obscurity or signed client-server negotiation
- What keeps it alive: fast follow-up, testing image, a community that tries to report well

STORY 2

The LLM flood

Perfectly formatted, mostly correct, and we've seen it before

By the numbers

Is there really a difference?

208 → 239

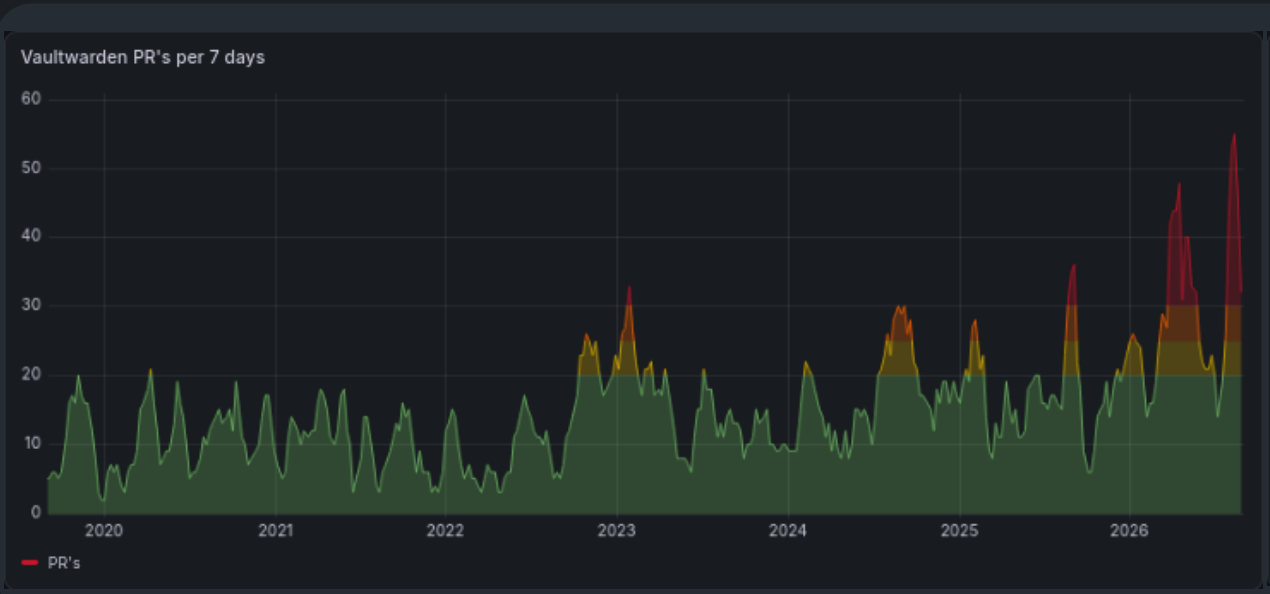
PRs opened per year
2024 vs Jan-Aug 2026

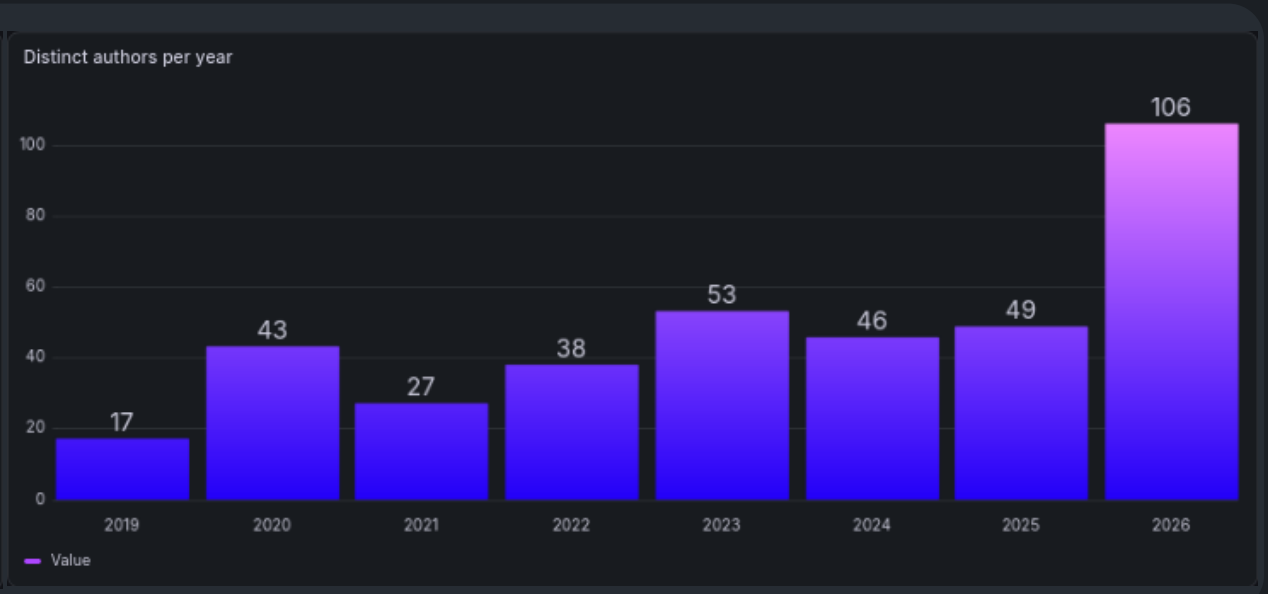
46 → 106

Distinct PR authors per year
2024 vs Jan-Aug 2026

4.5 → 2.3

PRs per author per year
2024 vs Jan-Aug 2026





Vaultwarden

The same one, again and… again

Real findings, already fixed, reported multiple times

The pattern

- Mostly real findings, mostly already known or even already fixed
- Same finding reported many times, each as a full write-up
- A few ungrounded ones: confident, structured, not valid, no clear reproduction steps
- Takes minutes to read, check and close
- The cost is triage, not the bug
- Review sometimes costs more than writing the fix ourselves

What actually helps

- Search advisories and closed reports before writing
- Say which version you tested; most duplicates are already fixed
- One reproduction beats a page of impact analysis
- Sent through the private advisory channel, not via an issue
- Example: the SSO and SSRF advisories mitigated in 1.36.0 came in like this both reported multiple times by different users, all with LLM usage

STORY 3

Building a (static) Rust binary

4 musl targets, 1 toolchain and abandoned crates

The crates nobody else maintains

When upstream stops moving, the dependency becomes yours

yubico_ng

- Fork of the unmaintained yubico crate
- No good alternative
- YubiKey OTP verification against YubiCloud
- Refactored, added a test suite
- Kept current with crate dependencies, and the 2024 edition

job_scheduler_ng

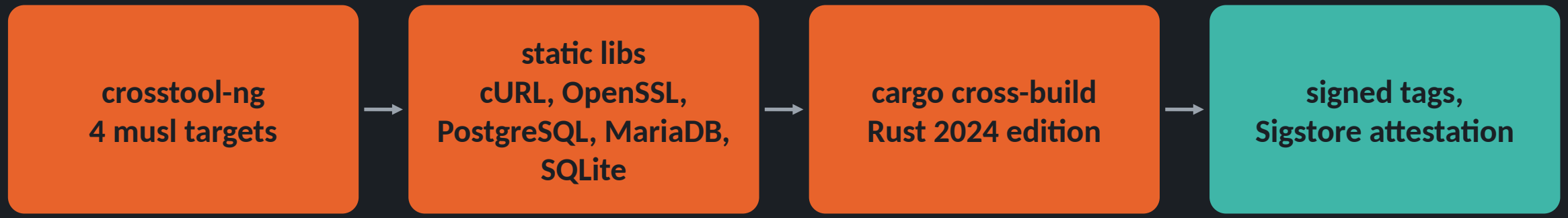
- Fork of job_scheduler
- Alternatives bring in more dependencies which we do not need
- Drives every periodic task: purges, trash, sends, incomplete 2FA, emergency access
- Cron parsing and time handling that stays on maintained crates
- Small crate, large blast radius

Why bother

- cargo audit and deps.rs stay green: no RUSTSEC advisories pinned forever
- One old crate can pin an old tokio, hyper or openssl for the whole tree
- Distro packagers (nixpkgs, Debian, Alpine, etc..) might refuse stale, insecure or duplicated deps
- Owning a fork is cheaper than carrying patches in every release

The build pipeline

Alpine (and Debian) images for amd64, arm64, armv7 and armv6



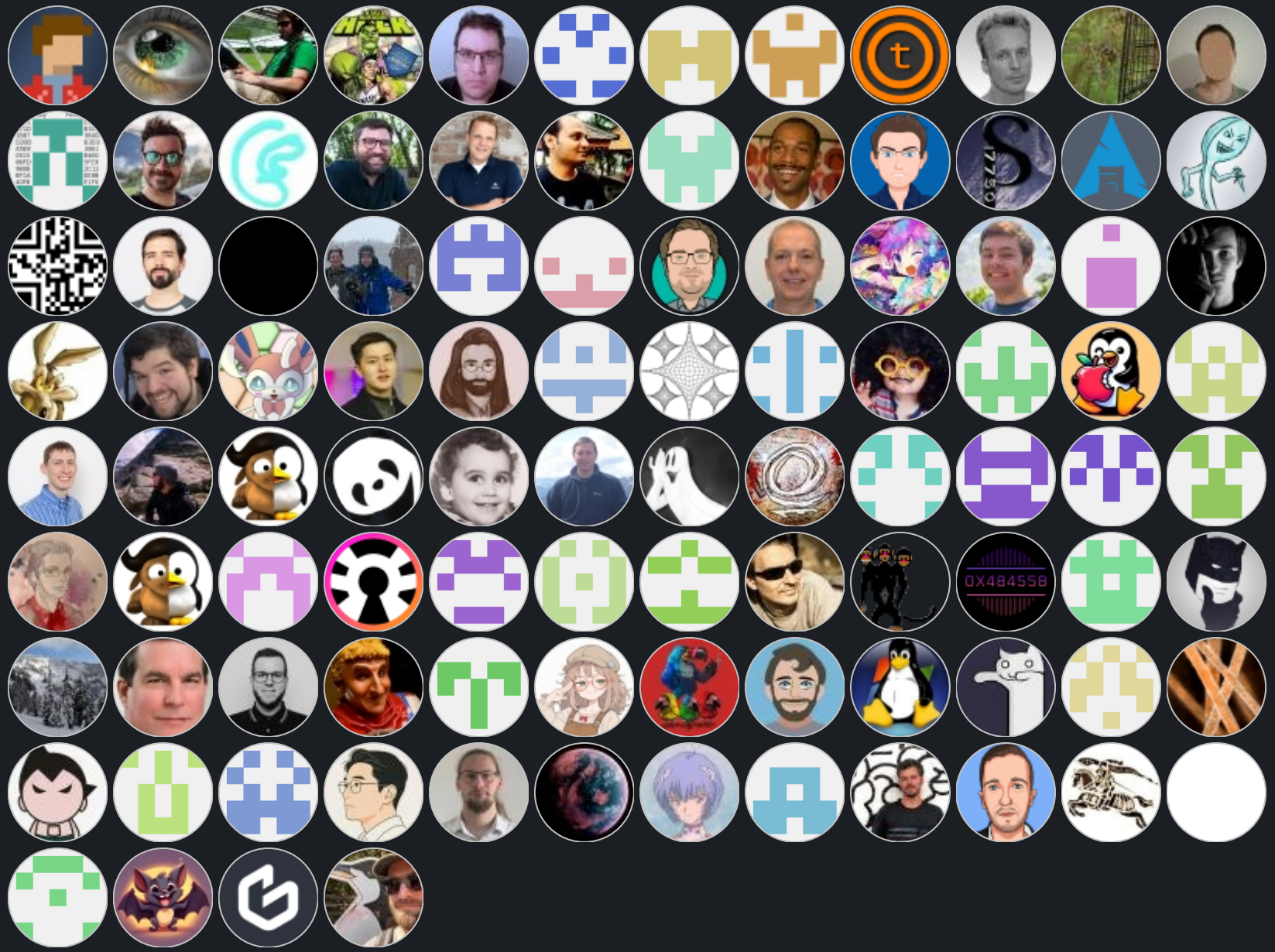
Why a custom build image

- Stay in control; enough moving targets already
- Rust's cross-rs does not provide needed libraries
- Crates do not always build correctly for all architectures
- Patching if needed; MariaDB has broken a few times already

The bug that could have taken weeks

- Segfault, Alpine images only, not reproducible on Debian
- Cause: a bare `-static` in TARGET_LDFLAGS built OpenSSL without thread support
- Tracked down with the help from an LLM

Thanks to all the contributors



Questions

BlackDex · github.com/BlackDex

Mathijs van Veluw · linkedin.com/in/mathijsvanveluw

Vaultwarden