OWL Information Security Policy

Organization: Own What's Left LLC (Kentucky)
Owner and sole developer: Kevin Michael Smith
Security contact: security@ownwhatsleft.com
Version: 1.10
Effective: 11/06/2026
Next review: 02/09/2027 (every six months, and after any incident)

This document states what OWL actually does today. Where a control does not exist, it says so and gives the plan. Nothing here is aspirational.

---

1. Scope and organization

OWL is a household budgeting application for Windows and Android, plus a sync service ("the relay") that stores encrypted data and proxies a small number of third-party calls.

The business is a single-member Kentucky LLC with no employees and no contractors. One person writes the code, operates the server, and holds every credential. There is therefore no internal access-control hierarchy to describe: the control is that exactly one person has access, and that person is named above.

2. What data exists, and where it lives

On the person's own device. The budget itself: accounts, transactions, categories, and settings, in a local SQLite database in the application's data directory. This is the system of record.

On the relay, when the person turns on sync. Encrypted blobs, plus any budget or device names, which are stored in plain text so the account page can show them. The server stores ciphertext it cannot read and has no key to read it with. It also holds a per-household revision counter and a change log of file names and sizes so devices can ask what is new.

On the relay, as service records. A license registry (license id, entitlement flags, expiry, the household it is bound to, and the website account it belongs to, if any), a crash log of stability data, and each household's bank access tokens, encrypted at rest (see section 5).

On the web server, for website accounts. An email address, a password hash, the license the account owns, and a beta application if one is sent. Nothing about any budget. Accounts are open. Buying is not open yet. The email is encrypted at rest, passwords are hashed with scrypt, two-step sign-in is available, and a budget can be unlinked from an account once per 90 days.

Nowhere else. There is no analytics service, no advertising network, no data broker, and no third party receiving user data other than those listed in the Privacy Policy.

3. What is never collected or stored

These are architectural guarantees, not settings:

- **No bank credentials.** Sign-in happens at the bank's own site through Plaid. OWL never sees or stores a banking password.
- **No account numbers or card numbers.** Payments, when they exist, will run at the payment processor.
- **No budget data on the server in readable form.** See section 4.
- **No receipt photographs leave the device.** Receipt scanning uses on-device text recognition.
- **No user accounts are required** to use the software.

4. Encryption

In transit. The application reaches our servers over HTTPS. A household running its own relay can instead use a relay+http:// address, which connects over plain HTTP on Windows and Android.

Budget data at rest on the server. Every file a device writes to the relay is encrypted on the device before it leaves. The scheme is AES-256-GCM with a fresh 12-byte IV per record, under a key derived from the household passphrase using PBKDF2-HMAC-SHA256 at 310,000 iterations. The passphrase is never transmitted and is not recoverable by us. A forgotten passphrase means the data cannot be decrypted by anyone, including the operator. This is by design and is stated plainly to users rather than hidden.

Authentication to the relay. A household proves itself with a token that is a one-way SHA-256 digest of its own sync key. The server can verify the token but cannot derive the key from it, so possession of the entire server database does not yield the ability to decrypt anything.

Identifiers. Households are opaque UUIDs.

License keys are Ed25519-signed and verified offline by the client. A website account holds the key of each license it owns, encrypted at rest like the email, so it can return the key to the account holder.

Bank data specifically. Transactions fetched from the bank aggregator are never written to disk on our servers. They pass through the proxy in memory and go straight to the device that asked for them. Once there they join the budget, so every copy that leaves the device again, including everything held on the relay, is AES-256-GCM ciphertext under the household's own key. The one plaintext copy lives in the local database on the person's own device, where it is covered by that device's own protections and nothing of ours. See section 14, item 3.

5. Secrets and key management

Third-party API secrets (Plaid, the AI provider) exist **only** on the server and are injected into outbound requests at the point of call. No client build contains them. Every such call is proxied by a single file in one codebase.

Signing keys (for desktop updates, the Android app and licenses) are held in a directory excluded from version control. Loss of these keys is treated as a recovery event. They are backed up off-site in an encrypted archive, kept apart from every automated credential.

Administrative access requires a secret held only on the server, readable only by the operator.

Bank access tokens belong to the household. The device that connects a bank keeps its token. The household's other devices receive it only inside the encrypted budget, under the household's own key. The server cannot read that budget. The relay keeps its own copy of each token, encrypted at rest under a server-held key. It decrypts them only to run the connections: to check that a token belongs to the household using it, to connect, release, replace or refresh a connection, and to record, at most once a day, that the household used its connections, so connections left unused can be released. It asks the bank aggregator for data only when one of the household's devices asks, and returns that data only to that device. Losing that key strands the stored tokens, which then require removal through the aggregator's support.

6. Access control

This section is the access control policy.

Who has access. One person, named in the header. There are no employees, contractors, shared logins or service accounts held by anyone else. Nobody has been granted access and nobody has had access revoked, because there has never been a second person to grant it to.

Production assets, and what reaches them.

Asset	Reached by	How access is controlled
Relay server (host)	Operator only	Hosted provider in the United States, key-based access
Relay administration	Operator only	A secret held only on the server
Encrypted household data	The household that owns it	Token derived by SHA-256 from that household's own sync key. The operator cannot read the contents.
Provider secrets (Plaid, AI)	Server process only	Held only on the server. Never in a client build.
Bank access tokens	The household's devices and the server process	Between devices, only inside the encrypted budget. On the server, encrypted at rest (AES-256-GCM) under a server-held key.
Signing keys	Operator only	Excluded from version control, backed up off-site in encrypted form
Source repository	Operator only	On the operator's own equipment.

Granting and revoking. Any change to the list above requires this document to be updated first, including the reason and the date. If a second person is ever given access to any production asset, the roles, the

review cadence and the de-provisioning steps are written here before the access is granted, not afterwards.

Non-human authentication. No service authenticates by username and password. Devices prove themselves to the relay with a token derived from their household key. Outbound calls to providers authenticate with API credentials held only on the server. Cloud storage backups use OAuth tokens held on the person's own device. Traffic to our servers runs over TLS.

Least privilege in practice. Each credential reaches exactly one thing: a household token opens one household, a provider secret calls one provider, an administrative secret reaches administration and nothing else. No credential in the system grants blanket access to everything.

Periodic access review. Every six months, on the cadence in the header, the operator works down the asset table above and records the result in section 15. The pass covers: whether the list of assets is still complete, whether anyone or anything has gained access that is not listed, that multi-factor authentication is still enabled wherever the account supports it, rotation of administrative secrets, and that the bank token store holds only tokens with a living purpose. Any credential that no longer has a purpose is destroyed rather than left dormant. A review that finds nothing is still recorded, because the record is the evidence that the pass happened.

Endpoint hardening. Rate limiting protects every proxied route so that a stolen household token cannot run up cost or volume. Limits apply per household and separately for each service, and AI requests also have daily and monthly ceilings. Diagnostic detail is not served publicly.

Physical. The server runs at a hosted provider.

7. Subprocessors

The third parties that receive personal information, what each receives and when, are listed in the Privacy Policy. This policy does not keep a second list.

The bank feed is the one place transactions pass through our server in readable form, in memory and in transit only, and this is stated to users rather than buried. Receipt text is capped, scrubbed of card numbers before the request is built, and never stored by us.

8. Monitoring, logging and incident response

The relay records unexpected process deaths to a crash log with timestamp, cause and stack. The service restarts automatically, and restarts are logged.

Application crashes on user devices upload stability data only: the error, a stack, and the application version. Never budget content, never account or transaction data.

During the beta these uploads are automatic and cannot be switched off. Consent is given by joining the beta: the terms testers agree to state that crashes are reported, and a tester whose crashes are never reported cannot help find the crash. The application says the same thing again on the crash log row in Settings, so the arrangement is visible in two places rather than buried in one.

This holds because testers opt in knowingly. General availability is different, because the people running the software will be users who never agreed to anything of the sort, so a consent choice is required before then and is tracked as a release blocker (see section 14, item 6).

Our servers also keep a request log, which is how an attack gets noticed and traced: the address a request came from, the time, what was asked for, whether it worked, and what kind of browser or app made it. The one-time codes in the links we email you are removed before a line is written, and passwords, sign-in cookies and the keys your devices use to authenticate are never recorded. Only the server account that writes the log can read it, and records are kept for ninety days at most.

Provider error text is logged server-side and never returned to clients, so an upstream message cannot leak request identifiers or institution detail to a user.

Incident response. With one operator the plan is short and real.

1. **Detect** through the crash log, the restart log, or a report to the security address.
2. **Contain** by taking the affected service offline. This does not stop anyone budgeting, because every device holds a complete local copy and the application works entirely offline.
3. **Assess** what could have been exposed, remembering that budget content on the server is ciphertext.
4. **Rotate** any credential that could plausibly have been reached.
5. **Notify** affected users directly, in plain language, including when the honest answer is that the full extent is not yet known. Target for first notification is 72 hours from confirmation.

9. Change management and testing

All code is version controlled. Every change ships with an automated suite of more than 4,000 tests, plus two end-to-end browser smokes covering the desktop and phone applications. The suite pins security-relevant behavior by name, including that the server stores no plaintext, that credentials never appear in client-bound responses, and that a stranger's token is refused before any third party is contacted.

Money-affecting logic requires tests before release, without exception. Releases are signed and distributed over TLS with signature verification in the updater.

Dependencies are managed through npm and Cargo lockfiles and updated deliberately rather than automatically.

Patch service levels. Operating system security updates are installed automatically on the server. Everything else is patched against the clock below, measured from the point the operator becomes aware of a fix, whether from a vendor advisory, a dependency alert, or a report to the security address.

Severity	Applied within
Actively exploited, or critical and reachable from the internet	7 days
High	30 days
Everything else	90 days, or the next release, whichever comes first

These are deliberately achievable for one operator rather than impressive. A deadline that gets missed is worse than a longer one that is kept. If a fix cannot be applied inside its window, the reason and the mitigation are recorded in section 15 at the next review.

End-of-life software. Supported versions are an inventory, not an assumption. The list below is confirmed at every review, and anything inside twelve months of its end-of-support date gets a replacement plan recorded against it before that date arrives.

Component	Status	Support ends	Plan
Server operating system	Long-term support release, security updates installed automatically	Confirmed at each review	Move to the next long-term release before support ends
Node.js, server runtime	Long-term support	Confirmed at each review	Move to the next LTS before its maintenance ends
Application dependencies	Pinned by lockfile	Continuous	Reviewed when an advisory arrives, updated deliberately

No component that handles user data may pass its end-of-support date. Where that is unavoidable in the short term, the compensating control and the exit date are recorded rather than left implicit.

10. Data retention and deletion

On the device. Deletions are soft, and financial history is deliberately never destroyed: an account or category is closed or hidden rather than erased, so a past month always still adds up. Deleting a mistaken entry is a normal, undoable action. "Fresh start" wipes every budget record on that device and keeps only application preferences.

On the relay. Encrypted files persist until the household replaces them. Compaction folds old journals into a snapshot and removes what it superseded, so the server does not accumulate history indefinitely. Turning off sync on a device stops that device syncing and leaves the household's data in place for its other devices.

Deleting a household from the server. One authenticated action removes the household directory with every encrypted file, the change log and the device gossip. Authorization is the household token, which means possession of the budget. It is idempotent, so erasing something already gone reports success rather than an error.

License bindings are deliberately not touched by an erase. A license is something the person paid for and it outlives their data. Moving one to a new household is what the rebind route is for, and that route carries

the anti-sharing cooling-off. Clearing a binding here would be a way around it.

A person who wants everything gone can also write to the security address and have the operator do it.

Third parties. Disconnecting a bank releases the connection at Plaid, which revokes it at the bank. Receipt text sent for reading is not retained by us. It is sent through the provider's commercial API, whose terms do not use customer inputs for model training.

11. Reporting a vulnerability

Send it to security@ownwhatsleft.com. Reports are acknowledged within three business days. There is no bug bounty. Good-faith research is welcome and will not be met with legal threats, provided it does not access another person's data, degrade the service for others, or involve social engineering.

12. Backup and recovery

The application writes local backups on a rotation and takes a safety backup before destructive operations. Households using sync hold a full copy of their data on every device, so the loss of the server does not lose anyone's budget. Server records are kept in nightly encrypted backups with an off-site copy.

13. Risk register

Reviewed on the cadence in the header. "Likely" here means plausible within a year at current scale, not certain.

#	Risk	Likelihood	Impact	What reduces it
R1	The relay host is compromised	Low	High	Budgets are ciphertext the server cannot read. Bank data is in memory and in transit only. Provider secrets are held only on the server.
R2	A provider secret leaks through a client build	Very low	High	No client contains the Plaid or AI provider secrets. Every call that uses them is proxied by one file of server code.
R3	A person loses their passphrase	Likely	High for them	Unrecoverable by design and said plainly rather than hidden. The printable Recovery Kit exists to prevent it.
R4	A device is lost or stolen	Likely	Medium	The budget is only as protected as that device.
R5	A dependency is compromised upstream	Low	High	Lockfiles, deliberate updates rather than automatic, and a small dependency surface.
R6	The sole operator is unavailable	Certain eventually	High	Signing keys are backed up off-site in encrypted form. There is no succession plan, and this is an open weakness rather than a solved one.
R7	A third party mishandles data we sent	Low	Medium	Every third party is listed in the Privacy Policy, used only for the feature it serves, and given no more than that feature needs.
R8	Abuse drives cost or volume	Medium	Low	Per-household rate limits on every proxied route, tracked separately per service.
R9	A bug exposes one household's data to another	Low	High	Every route is scoped by household id and gated on a token derived from that household's own key. Suite-pinned, including that a stranger's token is refused before any third party is contacted.

14. Known gaps and remediation

Stated deliberately, because a policy that claims completeness is not credible for a one-person operation.

1. **No third-party penetration test or SOC 2 audit.** Not planned before public launch. Revisit at scale.
2. ~~The relay is not on hosted infrastructure.~~ **Closed 08/19/2026.** The relay runs at a hosted provider.
3. **Local budget databases are not encrypted by the application.** They are protected by whatever the device's own operating system provides, which on some machines is nothing. Encryption applies to everything that leaves the device. Encrypting the local database would force a passphrase on people who never turned on sync, so this is a deliberate trade rather than an oversight.
4. ~~Backup files include device-local credential settings.~~ **Closed 08/09/2026.** The credential namespace is excluded from every exported backup, and a test fails the build if a token ever appears in one again.
5. ~~No formal risk register.~~ **Closed 08/09/2026.** See section 13.

6. **No consent choice for crash reports at general availability.** Beta applications are open on the website. Public download is not open yet. The terms testers accept state that crashes are reported, so beta consent is given by joining. The gap is general availability, where users never agreed to be testers, and the choice is a release blocker before then.
7. ~~No single action erases a household from the server.~~ **Closed 08/09/2026.** One authenticated action, authorized by the household token, removes the household directory in one step, and is idempotent. License bindings are deliberately untouched, see section 10.
8. **Application-level authentication is optional.** No account is required to use OWL, and the budget lives on the device. An optional app lock (a PIN, plus a fingerprint or face where the phone offers one) is available on Windows and Android. It is a screen lock, not encryption. Without it, anyone holding an unlocked device can reach the bank connection flow, with the device's own lock screen as the only barrier.
9. **No multi-factor authentication on the server itself.** Provider accounts that support multi-factor authentication have it enabled.

15. Review log

Reviewed by the owner every six months and after any security incident, material architecture change, or new subprocessor. Each review updates the version, the gap list, the end-of-life inventory, and adds a row here. An empty finding is still a row: the record is the evidence.

Date	Type	Findings	Actions taken
08/09/2026	Initial	No written policy existed. Backups carried device credentials. No risk register. No way to erase a household.	Policy written. Credentials excluded from backups and pinned by test. Risk register created. Household erase built and tested.
08/12/2026	Update	Business formed as a Kentucky LLC. Web store with customer accounts entered build.	Organization updated. Account-store commitments recorded in section 2. Test count refreshed.
11/06/2026	Update	Plain HTTP to a household's own relay was not covered. Several facts had gone stale since 08/12/2026.	Transport wording corrected. Facts brought up to date. Server detail generalized for publication. Full policy published as a PDF beside the website summary.

Next review due: 02/09/2027.