Cybersecurity Policy

Entity: Velentrix LLC **Product:** RallyPoint By Velentrix
Version: 1.0 **Effective:** May 18, 2026 **Owner:** Chuck Labenski, Sole Member

---

This policy describes the security controls applied to the RallyPoint By Velentrix platform. Velentrix LLC is a single-member entity. There is no security team and no additional employees; the Sole Member acts as the operator, administrator, and incident responder. Controls are sized to that reality: small surface area, strong perimeter, encrypted credentials, and tight blast radius. This document is factual — it describes what is in place, not aspirational targets.

1. Data Classification and Handling

Data handled by the platform is classified into four tiers:

- **Restricted** — Brokerage OAuth access tokens, OAuth refresh tokens, and customer-supplied API keys/secrets. Encrypted at rest, never logged, never emitted in HTTP responses outside the owning user's session.
- **Confidential** — Customer email, password hashes (bcrypt), trading history (orders, fills, P&L), portfolio snapshots. Access scoped by authenticated user ID.
- **Internal** — Application logs, scan results, market-data caches, operational metrics. No customer secrets are present in logs.
- **Public** — Marketing site, terms of use, privacy policy.

Customer secrets (tokens and API keys) are written to the database via the application layer only; they never appear in stdout, log files, or error responses. Database backups of the application store inherit the same encryption-at-rest properties as the live database file.

2. Access Control and Privileged Access Management

Customer access. The application enforces username + password authentication backed by bcrypt password hashing. Session tokens are HTTP-only, bound to a single user ID, and validated on every API request via a FastAPI dependency. Sensitive actions (changing credentials, starting/stopping the trading engine, disconnecting the brokerage) require an active authenticated session; there is no public, unauthenticated path that reaches customer data or trading actions.

Operator (admin) access. The production environment is a single RunPod GPU pod. Shell access is available only to the Sole Member via SSH using a public-key pair held on a single workstation. Passwords for SSH login are disabled. There are no other privileged accounts.

Secrets management. Application secrets (OAuth client ID/secret, encryption keys, database paths, third-party API keys) live in a `.env` file on the production pod with file mode `600` and ownership restricted to the application user. The `.env` file is excluded from version control via `.gitignore` and is never copied off the pod. Source code is stored in a

private GitHub repository (`chucklabenski/velentrix-platform`) and pushed using SSH keys; the repository contains no credentials.

End-user broker credentials are obtained exclusively via OAuth. The platform does not accept, transmit, or store plaintext brokerage API keys or secrets from end users. The customer-facing surface area exposes only an OAuth "Link Brokerage Account" flow; on successful authorization the platform receives and encrypts a short-lived bearer token from the brokerage. Eliminating the manual key-paste path removes a class of credential-handling risk (clipboard capture, shoulder-surfing, accidental commits to support tickets) from the threat model.

Least privilege. The application process runs as a non-root user. Database queries use parameterized statements; there is no shared-credential database administrator account exposed to the application code path.

3. Encryption of Data at Rest and in Transit

In transit. All public traffic terminates TLS at the Cloudflare edge. The origin is reached exclusively through a named Cloudflare Tunnel (`cloudflared`) — there is no publicly routable origin IP and no port open to the public internet on the RunPod host. Modern TLS (1.2+) is enforced by Cloudflare; HTTP requests are upgraded to HTTPS at the edge.

At rest. Brokerage OAuth access tokens, refresh tokens, and any customer-supplied broker API keys are encrypted before being written to the application database using a symmetric application-layer encryption key. The encryption key is held in the `.env` file (file mode `600`) and is not committed to source control. Customer passwords are stored only as bcrypt hashes; the plaintext password is never persisted or logged.

Key management. The application encryption key is generated once during initial provisioning, stored in the production `.env` file, and is the responsibility of the Sole Member to safeguard and rotate. Rotation procedure is documented in the internal runbook and requires re-encrypting all stored tokens.

4. Vulnerability Management and Patch Management

Dependency surface. The application is built on Python 3.11 with FastAPI and Uvicorn, runs behind a Caddy reverse proxy, and uses SQLite as the application database. Front-end is static HTML/CSS/JS with no build pipeline and no JavaScript package manager dependencies. This keeps the dependency surface small and auditable.

Patching cadence.

- **OS & base image:** RunPod base images are refreshed when the pod is rebuilt; the Sole Member reviews and applies security updates to installed system packages on a monthly cadence, or sooner if a high-severity CVE is announced for a package in use.
- **Python dependencies:** `pip list --outdated` is reviewed at least quarterly. Security-flagged updates (via GitHub Dependabot alerts on the private repository) are applied as

soon as practical.
- **Caddy & Cloudflared:** Updated when a new minor release is published, on a monthly cadence.

Vulnerability discovery. GitHub Dependabot alerts are enabled on the private source repository. Cloudflare WAF protects against common injection and automated-abuse patterns at the edge. Application input that reaches the database uses parameterized queries; user-supplied HTML is escaped before being rendered.

5. Incident Response and Disaster Recovery

Incident response. The Sole Member is the single on-call responder. The response playbook is:

- **Detect** — alerts arrive via application error logs and Cloudflare anomaly notifications. Suspected credential compromise also triggers a manual review.
- **Contain** — for a brokerage-token compromise, the affected token is revoked from the database immediately, the trading engine is stopped for the affected account, and the customer is notified to re-authorize.
- **Eradicate** — root cause is identified, the relevant application or infrastructure fix is deployed.
- **Recover** — affected customers are notified, the brokerage relationship is restored, and a post-mortem note is written to the internal runbook.
- **Notify** — material security incidents affecting customer data or brokerage credentials are disclosed to the affected customer and, where applicable, to the brokerage partner, without undue delay.

Disaster recovery. The application database (SQLite) is backed up on a regular cadence to a separate location on the pod, with the backup file inheriting the same restrictive file permissions. The source code is mirrored to a private GitHub repository, allowing the application to be redeployed to a fresh pod from source. Recovery time objective for a total-host failure is measured in hours (re-provision pod, re-deploy from GitHub, restore database from backup, re-issue the Cloudflare tunnel credentials). Brokerage OAuth tokens are customer-recoverable via the standard re-authorization flow.

6. Physical Security

> **Not applicable.** Velentrix LLC has no physical office, no on-premise infrastructure, and no employees. The production environment is a single cloud-hosted GPU pod operated by RunPod, Inc. Physical-security controls of the hosting facility are the responsibility of the hosting provider; Velentrix does not maintain, store, or transport hardware, removable media, or printed materials containing customer data.

7. Vendor Risk Management

The platform depends on a small set of third-party vendors. Each was selected for its security posture and customer-data handling. The current vendor inventory is:

- **Alpaca Securities LLC** — Brokerage of record. Customer funds and securities are custodied at Alpaca, not at Velentrix. Velentrix sends order instructions and receives trading data via Alpaca's OAuth-authorized API. Velentrix never holds customer cash or securities.
- **RunPod, Inc.** — Compute hosting (single GPU pod). RunPod provides the underlying host, hypervisor, and physical-security controls.
- **Cloudflare, Inc.** — TLS termination, DDoS protection, WAF, and the named tunnel that connects the public hostname to the RunPod origin. No origin IP is exposed.
- **GitHub, Inc.** — Private source-code repository, branch protection, and Dependabot security alerts.
- **Stripe, Inc.** — Payment processing for subscription billing. Customer card data is held by Stripe; Velentrix never sees or stores PAN, CVV, or expiry.

New vendors are reviewed by the Sole Member before integration. The review covers: where customer data flows, how the vendor stores and encrypts that data, the vendor's stated security certifications (if any), and whether the integration can be revoked without disrupting customer trading. Vendors that would receive restricted or confidential data require evidence of an industry-standard security program before integration.

Document Control

This is Cybersecurity Policy v1.0, effective May 18, 2026. The policy is reviewed at least annually and after any material change to the platform architecture, vendor inventory, or organizational structure. Material revisions increment the major version number and are dated above. Questions about this policy should be directed to the Sole Member at `chuck@velentrix.com`.