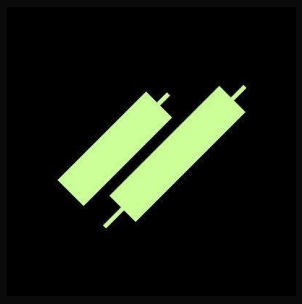
certora

Security Assessment

Final Report



Voltr Vault

February 2026

Prepared for Ranger Finance

Table of Contents

certora

Project Summary

Project Scope

Project Name	Repository Link	Commit Hash	Platform
Voltr Vault	https://github.com/voltrxyz/vault-program	84fba2e	SVM, Solana/Anchor

Project Overview

This document describes the security review of **Voltr Vault**. The work was undertaken from **February 9th, 2026** to **March 2nd, 2026**.

The following contract list is included in our scope:

/programs/voltr-vault/src/lib.rs
/programs/voltr-vault/src/error.rs
/programs/voltr-vault/src/instructions/mod.rs
/programs/voltr-vault/src/instructions/add_adaptor.rs
/programs/voltr-vault/src/instructions/calibrate_high_water_mark.rs
/programs/voltr-vault/src/instructions/cancel_request_withdraw_vault.rs
/programs/voltr-vault/src/instructions/close_strategy.rs
/programs/voltr-vault/src/instructions/create_lp_metadata.rs
/programs/voltr-vault/src/instructions/deposit_strategy.rs
/programs/voltr-vault/src/instructions/deposit_vault.rs
/programs/voltr-vault/src/instructions/direct_withdraw_strategy.rs
/programs/voltr-vault/src/instructions/harvest_fee.rs
/programs/voltr-vault/src/instructions/init_or_update_protocol.rs
/programs/voltr-vault/src/instructions/initialize_direct_withdraw_strategy.rs
/programs/voltr-vault/src/instructions/initialize_strategy.rs
/programs/voltr-vault/src/instructions/initialize_vault.rs

/programs/voltr-vault/src/instructions/instant_withdraw_vault.rs

/programs/voltr-vault/src/instructions/instant_withdraw_strategy.rs
/programs/voltr-vault/src/instructions/remove_adaptor.rs
/programs/voltr-vault/src/instructions/request_withdraw_vault.rs
/programs/voltr-vault/src/instructions/update_vault_config.rs
/programs/voltr-vault/src/instructions/update_vault_protocol_fee.rs
/programs/voltr-vault/src/instructions/withdraw_strategy.rs
/programs/voltr-vault/src/instructions/withdraw_vault.rs
/programs/voltr-vault/src/state/mod.rs
/programs/voltr-vault/src/state/adaptor_add_receipt.rs
/programs/voltr-vault/src/state/direct_withdraw_init_receipt.rs
/programs/voltr-vault/src/state/events.rs
/programs/voltr-vault/src/state/protocol.rs
/programs/voltr-vault/src/state/request_withdraw_vault_receipt.rs
/programs/voltr-vault/src/state/strategy_init_receipt.rs
/programs/voltr-vault/src/state/vault.rs
/programs/voltr-vault/src/utils/mod.rs
/programs/voltr-vault/src/utils/accounting.rs
/programs/voltr-vault/src/utils/constants.rs
/programs/voltr-vault/src/utils/decimal.rs
/programs/voltr-vault/src/utils/helpers.rs
/programs/voltr-vault/src/utils/math.rs
/programs/voltr-vault/src/utils/metadata_account_v3.rs
/programs/voltr-basic-adaptor/src/lib.rs
/programs/voltr-basic-adaptor/src/instructions/mod.rs
/programs/voltr-basic-adaptor/src/instructions/deposit.rs
/programs/voltr-basic-adaptor/src/instructions/initialize.rs
/programs/voltr-basic-adaptor/src/instructions/withdraw.rs

The team performed a manual audit of all the files in scope**.** During the manual audit, the Certora team discovered bugs in the code, as listed on the following page.

Protocol Overview

The Voltr Vault Program is a Solana-based yield vault protocol that allows users to deposit assets in exchange for LP tokens representing their proportional share of the vault. A vault manager deploys deposited funds into external yield strategies via trusted registered adaptor programs, while fees (management, performance, redemption, and issuance) are collected and distributed to the vault admin, manager, and protocol. The protocol supports SPL Token and Token-2022 assets and enforces a two-epoch delay before newly registered adaptors can be used, ensuring protection from newly added adaptors and allowing users a time frame into which they can do their research of said adaptor.

Assessment Methodology

Our assessment approach combines design level analysis with a deep review of the implementation to ensure that a protocol is secure, economically sound, and behaves as intended under realistic conditions.

At the design level, we evaluate the architecture, the economic assumptions behind the protocol, and the safety properties that should hold independently of a specific chain or environment. This process includes reviewing internal and cross protocol interactions, state transition flows, trust boundaries, and any mechanism that could be exploited to extract value, deny service, or alter core system behavior. At this stage, a focused threat modelling exercise helps identify key attack surfaces and adversarial capabilities relevant to the system. Design level issues often relate to incentive structures, governance implications, or systemic behavior that emerges under adversarial conditions.

Implementation analysis focuses on the concrete behavior of the code within the execution model of the target chain. This involves reviewing the correctness of logic, access control, state handling, arithmetic behavior, and the nuanced behaviors of the chain environment. Familiar classes of vulnerabilities such as reentrancy conditions, faulty permission checks, precision issues, or unsafe assumptions often surface at this layer. These findings require context aware reasoning that takes into account both the code and the architectural intent.

To support this analysis, the codebase is examined through repeated manual passes and supplemented by automated tools when appropriate. High-risk logic areas receive deeper scrutiny, invariants are validated against both design intent and actual implementation, and potential vulnerability leads are thoroughly investigated. Automated techniques such as static analysis, fuzzing, or symbolic execution may be used to complement manual review and provide additional insight.

Collaboration with the development team plays an important role throughout the audit. This helps confirm expected behaviors, clarify design assumptions, and ensure an accurate understanding of the protocol's intended operation. All findings are documented with clear reasoning, reproducible examples, and actionable recommendations. A follow up review is conducted to validate the applied fixes and verify that no regressions or secondary issues have been introduced.

Threat and Security Overview

The Voltr/Ranger finance programs introduce a customized vault implementation, similar to the ERC4626 standard, deployed on Solana. Users deposit a certain asset in return of shares, representing ownership in the vault. Vault managers and owners determine deposit allocation across strategies. Yield and losses are socialized across share holders minus management and performance fees. The former are charged by a flat rate over a period of time based on total asset value while the latter materialize only when the vault reaches a new all time high price-per-share ratio.

Additionally, issuance and redemption fees are optionally introduced, mostly due to potential slippage and other kinds of unexpected fees in third party integrators and strategies. These fees are relatively small (around 0.3%) and are charged by minting less shares to depositors. There is also a withdrawal period where users must first request a withdrawal receipt, wait for it to mature for a few days and then redeem it. The receipt stores the share value at the time of request e.g 100 shares worth 100 USD. At execution or cancellation time, the receipt value is compared to the current value of their shares and receives the lesser of the two - this is introduced to battle against free hedging and loss anticipation. On the other hand, users are fully exposed to all upcoming negative yields. In short, impermanent gains are forfeited by the withdrawer while impermanent losses are born by them.

The team performed extensive deep dive in an attempt to uncover high/critical vulnerabilities, most specifically theft of funds and denial of service attacks via improper input sanitization and sanity checks. The following attack vectors were checked and debunked:

- Theft of funds via vault substitution - attack involved inputting the account of a vault with cheaper share price and the `vault_lp_mint` of a more expensive one. Attack is not possible due to `vault_lp_mint` being a PDA derived with a seed of the `vault` pubkey. Each passed vault goes hand by hand with their lp mint.

- Theft of funds via asset recipient substitution - attack involved minting shares normally, but inputting a recipient of our assets as an ATA address controlled by the attacker himself. Attack is not possible since the recipient `vault_asset_idle_ata` has authority set to `vault_asset_idle_auth` which is a PDA derived from the vault pubkey. It's not possible for a user to create a phantom vault and pass their own idle ATA as recipient.
- Theft of funds via asset substitution - attack relies on supplying a different, cheaper asset mint than the vault one e.g pass 1e9 BONK instead of USDC, which is 150_000x cheaper. This is not possible due to a proper constraint ensuring that the mint of the vault matches the inputted `vault_asset_mint`

Furthermore we performed various other checks in terms of soundness, business logic fairness, ability to game the protocol, etc. Some inconsistencies came up:

- The redemption fee formula suffered 2 flaws - (1) it scaled exponentially with the size of the withdrawal and (2) it was applied on cancellations. See H-01
- Some state inconsistencies after neutralizing actions e.g +X profit followed by -X loss (net 0) did not result in starting state as it should. See M-01. Additionally some of the fees are mixed up and in some cases where issuance fees are turned on, they'd offset performance losses slightly, explained further in M-02.
- Attempting to inflate the price-per-share in order to negatively impact the vault's solvency, its users and the fees it charges

Disclaimer

There were several identified traces which didn't produce an outright vulnerability, but they're architecturally weak points, that could be addressed via raising signal about them:

- The deposit assets are tracked via a variable, but they also include the position held externally, e.g. inside Kamino. For such external integrations which issue protocol tokens per positions, there is slight room for a donation attack, via donating protocol tokens directly to the adaptor as it is about to report the vault's position
- minting of shares considers all assets as unlocked (following a profit yield)
- renouncing the admin would allow the protocol to be reinitialized
- fee parameters are instantly changed instead of undergoing a time-lock, which can impact UX

- withdrawals lack slippage, user has no control over his losses if the withdrawal period experienced negative PnL
- harvesting fees does not sync the PnL, leaving fees on the table for the next harvest, instead of collecting all outstanding fees at once

Findings Summary

The table below summarizes the findings of the review, including type and severity details.

Severity	Discovered	Confirmed	Fixed
Critical	-	-	-
High	1	1	1
Medium	2	2	2
Low	2	2	-
Informational	5	5	1
Total	10	10	4

Severity Matrix

Impact	High	Medium	High	Critical
	Medium	Low	Medium	High
	Low	Low	Low	Medium
		Low	Medium	High
			Likelihood	

certora

Detailed Findings

ID	Title	Severity	Status
H-01	Incorrect formula overcharges users	High	Fixed
M-01	Losses do not wipe locked profits and directly reduce PPS instead	Medium	Fixed
M-02	Performance fees include the issuance fee, inflating the overall fees paid	Medium	Fixed
L-01	Requesting and cancelling withdrawals does not accrue management fees	Low	Acknowledged
L-02	Creating idle ATA token account beforehand can brick vault initialization	Low	Acknowledged
I-01	Fee validation can exceed 100%	Informational	Acknowledged
I-02	Updating the profit lock period affects outstandings profits	Informational	Acknowledged
I-03	Frequent management fee updates accrue 0 fees	Informational	Acknowledged
I-04	Requesting and cancelling withdrawals use the same liveliness check	Informational	Acknowledged

I-05	Admin transfers and fee BPS updates are not 2 step	Informational	Fixed

High Severity Issues

H-01: Incorrect formula overcharges users

Severity: High	Impact: High	Likelihood: High
Files: src/utils/accounting.rs	Status: Fixed	

Description:

The current redemption formula is not-linear with respect to user withdrawal size. 100 shares - 100 assets vault, price-per-share = 1, redemption fee = 10%. The 3 cases are with a user requesting to withdraw that has a) 10% ownership, b) 50% ownership and c) 90% ownership. Plugging the numbers in the formula gives us:

$$clp = ia \times \frac{(tlp - ilp) \times (10000 - r)}{ta \times 10000 - ia \times (10000 - r)}$$

a - 10 x 0.8901 = 8.9, effective fee ~11%

b - 50 x 0.8181 = ~41, effective fee ~18%

c - 90 x 0.4736 = ~43, effective fee ~52%

The effective fee increases exponentially with the size of the withdrawal.

NB! This issue could've been 2 separate ones, however they share a mutual fix. The 2 issues are - (1) redemption fees are charged upon cancelling a withdrawal request and (2) redemption fees are overcharged due to incorrect formula. These fees should be only applied on successful withdrawals, not on cancellations which constitutes the first issue - they shouldn't be charged at all. The 2nd one is that despite being charged incorrectly, they are also charged more than the assigned percentage.

Due to them sharing the same fix, they will remain as a single High issue.

Recommendations:

Revise the formula

Customer Response:

Confirmed, replaced non-linear redemption formula with a linear one and removed fee charging on cancel withdrawal. Fixed in b46ecd5

Fix Review:

Fix confirmed.

Medium Severity Issues

M-01: Losses do not wipe locked profits and directly reduce PPS instead

Severity: Medium	Impact: Medium	Likelihood: Medium
Files: vault-program/programs/voltr-vault/src/state/vault.rs	Status: Fixed	

Description:

Vault positive PnL from external strategies is correctly locked and gradually released into the PPS to avoid sandwich attacks from users trying to capture incoming yield. However, if a loss occurs, thus a negative PnL, we subtract that loss directly from the global total assets and not the locked profits.

A simple scenario to trace:1. 100 assets (100 unlocked - 0 locked)2. We accrue 50 asset profit (100 unlocked - 50 locked)3. We accrue 50 loss, delta neutral state so we must end as we did in the start: 100 unlocked - 0 locked and NOT 50 unlocked - 50 locked

This means that users instantly bear the vault losses, but are only gradually exposed to the profit, leaving them in the negative in case of a profit → loss sequence, which is incorrect.

Recommendations:

Deduct the losses from the active profits first and reduce the total only if the loss amount > locked profit.

Customer Response:

Confirmed, Losses now deduct from locked profits first before reducing total assets. Fixed in 308bdd9.

Fix Review:

Fixes confirmed.

M-02: Performance fees include the issuance fee, inflating the overall fees paid

Severity: Medium	Impact: Medium	Likelihood: Medium
Files: voltr-vault/src/instructions/deposit_strategy.rs voltr-vault/src/instructions/withdraw_strategy.rs voltr-vault/src/instructions/direct_withdraw_strategy.rs	Status: Fixed	

Description:

The protocol charges both a performance fee, taken from positive PnL returned from strategies, and an issuance fee, charged from user deposits, via minting less shares for the same assets which inflated the exchange rate. In the happy path, any correlation between the exchange rate increase and PnL is handled via updating the HWM at every deposit, so issuance fees are not accounted as performance. However, in the event of losses reported from the strategies, subsequent deposits' issuance fees will inflate the performance fees charged the next time the vault surpasses its HWM.

A simple scenario to trace would be:1. totalAssets:totalSupply = 100:100, HWM = 12. A strategy reports a 10% loss, thus we have 10 assets less, rate is 90:1003. A user deposits 10 assets, with a 10% issuance fee and 0.9 rate, he receives 9 LP instead of 11, the 2 LP reduction is accounted for as the fee, thus the exchange rate goes back up, rate is 100:109 = 0.914. The strategy reports a 10% gain, the new rate is 110:108 = ~1.009, thus a performance fee will be charged on this gain in the exchange rate, which came purely from the issuance fee our user paid on their deposit

Recommendations:

Up for discussion

Idea: on deposit and withdraw they update the HWM due to the issuance fee which is fine in the happy path, but loss scenarios are unhandled with a simple update, instead they could instead increase the HWM by the PPS delta caused by the issuance fee. In the above case, when the user deposits in step3 they increase the PPS by 0.01 (0.91 - 0.9), thus we can increase the HWM to 1 +

0.01 = 1.01. This way when the profit comes in step 4 1.01 > 1.009 and no performance fee will be charged

Customer Response:

Confirmed, HWM adjusted by fee-induced PPS delta on deposit/withdraw, preventing issuance fees from inflating performance fees. Fixed in 1c80195

Fix Review:

Fix confirmed.

Low Severity Issues

L-01: Requesting and cancelling withdrawals does not accrue management fees

Severity: Low	Impact: Low	Likelihood: Medium
Files: programs\voltr-vault\src\instructions\request_withdraw_vault.rs programs\voltr-vault\src\instructions\cancel_request_withdraw_vault.rs	Status: Acknowledged	

Description:

The request_withdraw_vault and cancel_request_withdraw_vault instructions fail to trigger the management fee accrual before performing calculations that depend on the vault's price. This is inconsistent with the primary deposit_vault and withdraw_vault instructions, which correctly accrue fees before any state-dependent calculations.

Recommendations:

Apply the correct accrual logic.

Customer Response:

Acknowledged, management fee accrual gap on request/cancel withdraw is negligible — the time window is too short to materially affect fees

Fix Review:

Acknowledged.

L-02: Creating idle ATA token account beforehand can brick vault initialization

Severity: Low	Impact: Low	Likelihood: Medium
Files: vault-program\programs\voltr-vault\src\instructions\initialize_vault.rs	Status: Acknowledged	

Description:

The initialize_vault instruction attempts to create the vault's "idle asset" ATA using the standard associated_token::create CPI call. In the Solana SPL Associated Token Program, the standard create instruction is strict as it will return an error and cause the entire transaction to revert if the target account already exists.

Because the vault's address and its derived authorities (PDAs) are deterministic, an attacker can calculate the exact address of the vault's ATA before it is even created. By pre-initializing this account, the attacker prevents the official initialization transaction from completing successfully.

Recommendations:

Use create_idempotent instead of create. The idempotent version checks if the account exists; if it does, it simply succeeds.

Customer Response:

Acknowledged, vault initialization is permissioned; pre-creating the ATA is an unlikely griefing vector with no fund risk

Fix Review:

Acknowledged.

Informational Severity Issues

I-01: Fee validation can exceed 100%

Files:

programs\voltr-vault\src\instructions\update_vault_config.rs
programs\voltr-vault\src\instructions\update_vault_protocol_fee.rs

Description:

The vault's fee validation logic is inconsistent between the two primary setters. While update_vault_protocol_fee.rs correctly calculates the sum of all three fee components (admin, manager, and protocol) to ensure they do not exceed MAX_FEE_BPS, the logic in update_vault_config.rs only considers the admin and manager pair. Specifically, when updating the manager or admin performance/management fees in update_vault_config.rs, the code only adds the other "local" fee (admin + manager) and ignores the protocol's share.

Recommendations:

Properly validate the fees to be within the correct bounds.

Customer Response:

Acknowledged, fee validation is enforced at the admin/protocol level; misconfiguration requires a trusted admin acting against their own interest

Fix Review:

Acknowledged.

I-02: Updating the profit lock period affects outstandings profits

Files:

programs\voltr-vault\src\instructions\update_vault_config.rs
programs\voltr-vault\src\state\vault.rs

Description:

When LockedProfitDegradationDuration is updated, the new duration is applied retroactively to all currently locked profits. This happens because the LockedProfitState does not store the duration that was active when a profit was reported. Instead, calculate_locked_profit always pulls the current duration from the global VaultConfiguration, unfairly extending the unlock period for users.

Recommendations:

Update the duration while there are no outstanding deposits or keep track of which period the profits should be abided by, via a mapping + nonce.

Customer Response:

Acknowledged, profit lock duration updates are infrequent admin operations; retroactive effect is acceptable given operational simplicity

Fix Review:

Acknowledged.

I-03: Frequent management fee updates accrue 0 fees

Files:

voltr-vault/src/state/vault.rs
voltr-vault/src/utils/accounting.rs

Description:

The handle_management_fee function updates the last_management_fee_update_ts timestamp to the current time even if the calculated fee amount is zero. If the management fee calculation in accounting.rs rounds down to zero—as would happen with frequent updates over very short intervals — the accrued time is effectively discarded, preventing management fees from accumulating.

Recommendations:

Conditionally update the management fee only if the calculation does not round to 0 fees

Customer Response:

Acknowledged, management fee rounding to zero on frequent updates has negligible economic impact and self-corrects over longer intervals

Fix Review:

Acknowledged.

I-04: Requesting and cancelling withdrawals use the same liveliness check

Files:

vault-program\programs\voltr-vault\src\instructions\request_withdraw_vault.rs
vault-program\programs\voltr-vault\src\instructions\cancel_request_withdraw_vault.rs

Description:

Both instructions rely on the OperationalState::WITHDRAW_VAULT and DisabledOperation::WITHDRAW_VAULT flags to determine if the operation is allowed. This shared restriction means that if withdrawals are disabled (e.g., during an emergency), users are prevented from cancelling their existing pending requests, effectively trapping their escrowed funds until the "withdraw" operation is re-enabled, impacting their exposure to potential vault gains.

Recommendations:

Separate the 2 flags into a request and cancel flags respectively

Customer Response:

Acknowledged, disabling withdrawals is an emergency action; preventing cancellations during emergencies is intentional to preserve vault state

Fix Review:

Acknowledged.

I-05: Admin transfers and fee BPS updates are not 2 step

Files:

vault-program\programs\voltr-vault\src\instructions\init_or_update_protocol.rs
vault-program\programs\voltr-vault\src\instructions\update_vault_config.rs
vault-program\programs\voltr-vault\src\instructions\update_vault_protocol_fee.rs

Description:

Administrative roles in the Voltr program, such as the Protocol Admin and Vault Manager, as well as changes to the fee percentages, currently use a single-step update process that immediately updates the values. In the case of administrative roles, accidental transfers to an incorrect address can result in permanent loss of administrative control. In the case of fees, instant changes without a time-lock can disrupt the UX and trust.

Recommendations:

Add a pending->accept steps when transferring ownership to avoid accidental transfers. Add a timestamp based time-lock when changing fee BPS.

Customer Response:

Confirmed and partially fixed, admin transfers now use 2-step pending/accept flow; fee BPS time-lock not implemented, fixed in 9333d2a

Fix Review:

Fix confirmed.

Disclaimer

Even though we hope this information is helpful, we provide no warranty of any kind, explicit or implied. The contents of this report should not be construed as a complete guarantee that the contract is secure in all dimensions. In no event shall Certora or any of its employees be liable for any claim, damages, or other liability, whether in an action of contract, tort, or otherwise, arising from, out of, or in connection with the results reported here.

About Certora

Certora is a Web3 security company that provides industry-leading formal verification tools and smart contract audits. Certora's flagship security product, Certora Prover, is a unique SaaS product that automatically locates even the most rare & hard-to-find bugs on your smart contracts or mathematically proves their absence. The Certora Prover plugs into your standard deployment pipeline. It is helpful for smart contract developers and security researchers during auditing and bug bounties.

Certora also provides services such as auditing, formal verification projects, and incident response.