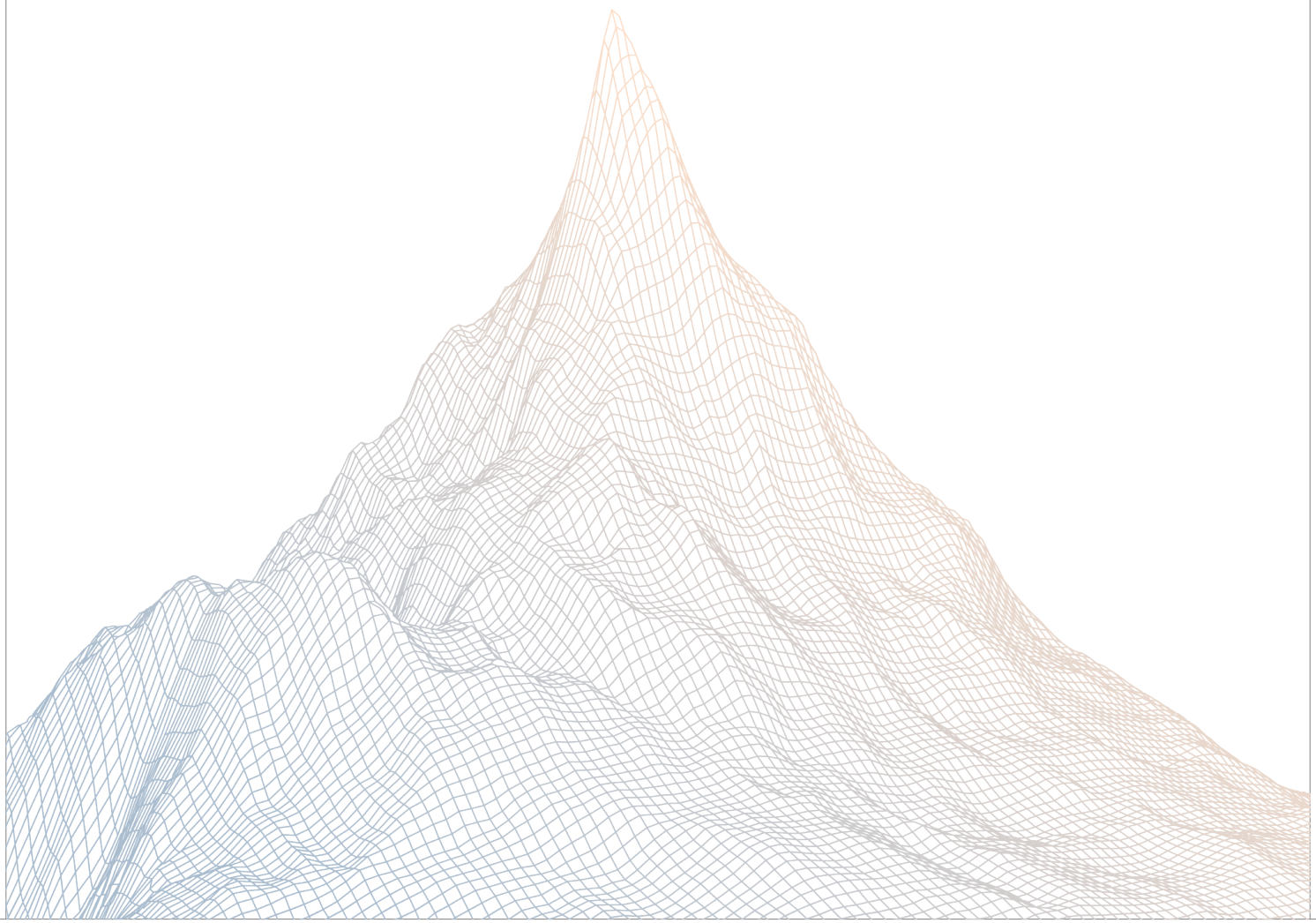
Zenith

Treehouse Finance

Smart Contract Security Assessment

VERSION 1.1



AUDIT DATES: March 24th to April 2nd, 2025

AUDITED BY: IIIIIII
Mario Poneder

Contents

1 Introduction

1.1 About Zenith

Zenith is an offering by Code4rena that provides consultative audits from the very best security researchers in the space. We focus on crafting a tailored security team specifically for the needs of your codebase.

Learn more about us at https://code4rena.com/zenith.

1.2 Disclaimer

This report reflects an analysis conducted within a defined scope and time frame, based on provided materials and documentation. It does not encompass all possible vulnerabilities and should not be considered exhaustive.

The review and accompanying report are presented on an "as-is" and "as-available" basis, without any express or implied warranties.

Furthermore, this report neither endorses any specific project or team nor assures the complete security of the project.

1.3 Risk Classification

SEVERITY LEVEL	IMPACT: HIGH	IMPACT: MEDIUM	IMPACT: LOW
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

2

Executive Summary

2.1 About Treehouse Finance

Treehouse is a decentralized application that introduces Treehouse Assets (tAssets) and Decentralized Offered Rates (DOR), new primitives that enable fixed income products in digital assets.

Users who deposit ETH or liquid staking tokens (LST) into the protocol receive tETH and contribute to the convergence of fragmented on-chain ETH rates.

tETH also enhances the cryptoeconomic security of DOR, a consensus mechanism for benchmark rate setting.

2.2 Scope

The engagement involved a review of the following targets:

Target	boring-vault-svm
Repository	https://github.com/Veda-Labs/boring-vault-svm
Commit Hash	`ea1e9036856accfaaf2767835230547fb59530a0`
Files	`boring-onchain-queue/*` `boring-vault-svm/*`

2.3 Audit Timeline

March 24, 2025	Audit start
April 2, 2025	Audit end
April 9, 2025	Draft Report published

2.4 Issues Found

SEVERITY	COUNT
Critical Risk	0
High Risk	0
Medium Risk	1
Low Risk	10
Informational	10
Total Issues	**21**

3

Findings Summary

ID	Description	Status
M-1	CPI digest restrictions can be circumvented using upgradeable programs	Resolved
L-1	Division before multiplication precision loss	Resolved
L-2	Users cannot withdraw assets from vault if withdraw_authority is not a queue	Acknowledged
L-3	First one to invoke permissionless initialization can set authorities	Resolved
L-4	Share token mint not explicitly created using Token-2022	Resolved
L-5	Vault wind-downs may cause fees to become stuck if multiple sub-accounts are used	Acknowledged
L-6	Fee updates apply retroactively	Acknowledged
L-7	Exchange rate volatility may affect fee compounding frequency	Acknowledged
L-8	Unnecessary pause check when updating CPI digest accounts	Resolved
L-9	update_cpi_digest() does not validate expected_size	Resolved
L-10	validate_associated_token_accounts() should use InvalidAssociatedTokenAccount as the error code	Resolved
I-1	Limited usability of view_cpi_digest instruction	Resolved
I-2	Use of outdated switchboard-on-demand library	Resolved
I-3	Exchange rate does not reflect owed fees	Acknowledged
I-4	Performance fees incentivize delayed exchange rate updates	Acknowledged
I-5	Asset account not checked in update_asset_data instruction	Resolved

ID	Description	Status
I-6	The update_cpi_digest instruction does not allow to update the CPI digest's properties	Resolved
I-7	Base asset mint not checked in deploy instruction	Resolved
I-8	Project relies on vulnerable crate dependencies	Acknowledged
I-9	get_rate_in_quote() will not work with SOL as the quote asset	Resolved
I-10	Share token mint could benefit from associated metadata	Resolved

4

Findings

4.1 Medium Risk

A total of 1 medium risk findings were identified.

[M-1] CPI digest restrictions can be circumvented using upgradeable programs

SEVERITY: Medium	IMPACT: Low
STATUS: Resolved	LIKELIHOOD: Low

Target

- programs/boring-vault-svm/src/lib.rs#L289-L297
- programs/boring-vault-svm/src/lib.rs#L845-L861

Description:

The CPI digest verification mechanism is intended to whitelist specific operations on a vault, e.g. transferring assets from the deposit sub-account to the withdrawal sub-account in the simplest case.

This mechanism is based on the assumption that a whitelisted instruction's underlying program is immutable. However, neither the vault's `manage` instruction nor its `update_cpi_digest` instruction verifies the program's immutability. Consequently, any CPI digest restrictions could be bypassed by upgrading an initially genuine underlying program after whitelisting, allowing potentially malicious operations on a vault.

Recommendations:

It is recommended to verify within the `update_cpi_digest` instruction that the instruction's program is owned by the non-upgradeable version of the `BPFLoader`, or has its upgrade authority set to `None`, or is any of Solana's built-in programs.

Treehouse: Resolved with @b0e92dbef81... by adding the`IngestInstructionDataSize` operator.

Zenith: Verified

4.2 Low Risk

A total of 10 low risk findings were identified.

[L-1] Division before multiplication precision loss

SEVERITY: Low	IMPACT: Low
STATUS: Resolved	LIKELIHOOD: Low

Target

- programs/boring-vault-svm/src/lib.rs#L715-L720
- programs/boring-vault-svm/src/utils/teller.rs#L310-L324
- programs/boring-vault-svm/src/utils/teller.rs#L443-L454

Description:

In the above instances, division before multiplication or unnecessary division by the inverse is performed, leading to a potential precision loss.

Recommendations:

It is recommended to restructure the above instances such that multiplications are performed before divisions and divisions by the inverse are replaced by straightforward multiplications.

Treehouse: Resolved with @06db56c4705...

Zenith: Verified

[L-2] Users cannot withdraw assets from vault if `withdraw_authority` is not a queue

SEVERITY: Low	IMPACT: Low
STATUS: Acknowledged	LIKELIHOOD: Low

Target

- programs/boring-vault-svm/src/lib.rs#L1478
- programs/boring-vault-svm/src/lib.rs#L1532-L1539

Description:

In the vault's `withdraw` instruction, the `withdraw_authority` (if set) cannot withdraw assets on behalf of a user, but has to own the shares itself, i.e. has to be the authority of the `user_shares` token account.

This is designed to be used with the `boring_onchain_queue` program, where users transfer their shares to the `queue` on a withdrawal request. This `queue` is configured as the `withdraw_authority` to facilitate the withdrawal.

However, the `deploy` as well as the `set_withdraw_authority` instructions impose no restrictions on the `withdraw_authority` which leaves users unable to withdraw in case the `withdraw_authority` was set to any authority different from the respective `queue` account of the `boring_onchain_queue` program.

Please note that this issue does not persist in case of permissionless withdrawals where the `withdraw_authority` is set to the zero account.

Recommendations:

It is recommended to validate the given `withdraw_authority` in the `deploy` and `set_withdraw_authority` instructions to ensure that only the respective `queue` account of the `boring_onchain_queue` program can be set as the `withdraw_authority`.

Treehouse: Acknowledged. This is a possible configuration mistake an admin could make when setting up a vault, but if they do make this mistake the correction is fairly easy, they just need to change the withdraw authority.

Zenith: Acknowledged as a remediable configuration mistake.

[L-3] First one to invoke permissionless initialization can set authorities

SEVERITY: Low	IMPACT: Low
STATUS: Resolved	LIKELIHOOD: Low

Target

- programs/boring-onchain-queue/src/lib.rs#L475-L489
- programs/boring-vault-svm/src/lib.rs#L1172-L1186

Description:

The permissionless nature (no `signer` restriction) of the `initialize` instructions and their respective `Initialize` account contexts allows the first caller after deployment to set the `authority` in the program configs.

Recommendations:

It is recommended to require co-signing of the `initialize` instructions using the programs' keypairs, which should only be known to the deployer. This can be achieved by modifying the `Initialize` account contexts as follows:

```
pub struct Initialize<'info> {
    #[account(mut)]
    pub signer: Signer<'info>,

    #[account(address = crate::ID)]
    pub program: Signer<'info>

    #[account(
        init,
        payer = signer,
        space = 8 + std::mem::size_of::<ProgramConfig>(),
        seeds = [BASE_SEED_CONFIG],
        bump,
    )]
    pub config: Account<'info, ProgramConfig>,
```

```
    pub system_program: Program<'info, System>,
}
```

Treehouse: Resolved with @5680c49bb4...

Zenith: Verified.

[L-4] Share token mint not explicitly created using Token-2022

SEVERITY: Low	IMPACT: Low
STATUS: Resolved	LIKELIHOOD: Low

Target

- programs/boring-vault-svm/src/lib.rs#L1210-L1219

Description:

The mint of the vault's share token (`share_mint`) is not explicitly created with the Token-2022 program although the related `user_shares` and `queue_shares` ATAs are explicitly handled using the Token-2022 program in all instances.

Recommendations:

It is recommended to adapt the `share_mint` initialization as follows:

```
    /// The mint of the share token.
    #[account(
        init,
        payer = signer,
        mint::token_program = token_program_2022,
        mint::decimals = base_asset.decimals,
        mint::authority = boring_vault_state.key(),
        seeds = [BASE_SEED_SHARE_TOKEN, boring_vault_state.key().as_ref()],
        bump,
    )]
    pub share_mint: InterfaceAccount<'info, Mint>,

    pub token_program_2022: Program<'info, Token2022>,
```

Treehouse: Resolved with @01dd9c33dcb...

Zenith: Verified

[L-5] Vault wind-downs may cause fees to become stuck if multiple sub-accounts are used

SEVERITY: Low	IMPACT: Low
STATUS: Acknowledged	LIKELIHOOD: Low

Target

- programs/boring-vault-svm/src/lib.rs#L580-L590

Description:

Fees are collected in _each_ sub-account PDA of each vault, but during calls to `claim_fees_in_base()`, only one of the vault's sub-accounts can be specified as the source ATA of the fees. If multiple sub-accounts have been used, it is likely that the fees will likewise be distributed over multiple sub-accounts. When it comes time to claim the fees, there may not be enough of the base asset present in a _single_ sub-account if the vault has been wound down, and all users have redeemed their shares.

Recommendations:

Modify `claim_fees_in_base()` to take in an amount to claim, rather than requiring the claiming of all fees.

Treehouse: Acknowledged.

Users are only benefitted from this, and really it comes down to mis-management on the strategists part. Fees will be regularly collected so even if a vault is wound down and this does happen, the loss to the strategist will be minimal compared to the fees they collected overtime. With this in mind it is best to keep the code simpler, and the function signature simpler so it is easier for strategists to collect fees.

[L-6] Fee updates apply retroactively

SEVERITY: Low	IMPACT: Low
STATUS: Acknowledged	LIKELIHOOD: Low

Target

- programs/boring-vault-svm/src/lib.rs#L494-L523

Description:

When an admin calls `set_fees()` the new fee rates are set immediately, without checkpointing the fee compounding for the prior period. The next time the `exchange_rate_provider` calls `update_exchange_rate()`, the new fee will apply to the period of time since the last call to `update_exchange_rate()`, which may be a long time ago

Recommendations:

Modify `set_fees()` to compound the fees at the current rate (as if `update_exchange_rate()` had been called with its previous value) prior to updating the fees. This may affect the compounding rate, however.

Treehouse: Acknowledged.

[L-7] Exchange rate volatility may affect fee compounding frequency

SEVERITY: Low	IMPACT: Low
STATUS: Acknowledged	LIKELIHOOD: Low

Target

- programs/boring-vault-svm/src/lib.rs#L673-L685

Description:

As a safety mechanism, if the `exchange_rate_provider`'s update in too soon, or if the change as compared to the prior value is too large or too small, the protocol is automatically paused for manual inspection. When it is later unpaused, there is no code that performs the protocol fee compounding that would have occurred had the protocol not been paused. This means that if there are multiple such events in a row, the protocol fee compounding will not match the predicted rate. Rather than using continuous compounding, the protocol uses discrete compounding, so any changes in the frequency of non-pausing calls to `update_exchange_rate()`, will cause the stated interest rate not to match the publicly stated one.

Recommendations:

Document the expected compounding behavior during pauses and how this affects the interest rate, along with the period minimums and high water mark

Treehouse: Acknowledged.

[L-8] Unnecessary pause check when updating CPI digest accounts

SEVERITY: Low	IMPACT: Low
STATUS: Resolved	LIKELIHOOD: Low

Target

- programs/boring-vault-svm/src/lib.rs#L1559

Description:

One of the constraints in the `UpdateCpiDigest` account struct causes requests made while the vault is paused. If there is a market event that causes the exchange rate to change such that the vault becomes paused, it may be advantageous to be able to create or disable CPI digests while the market is still paused, to ensure that there is no automation that was forgotten to be disabled, that may perform out-dated operations.

Recommendations:

Remove the constraint requiring that the vault is unpaused

Treehouse: Resolved with @d8fb374aaef...

Zenith: Verified.

[L-9] `update_cpi_digest()` does not validate `expected_size`

SEVERITY: Low	IMPACT: Low
STATUS: Resolved	LIKELIHOOD: Low

Target

- programs/boring-vault-svm/src/lib.rs#L283-L285

Description:

The function documentation for `update_cpi_digest()` states that it does not validate that the input `Operators` hash to the right ID, presumably to save CUs. It does not comment on whether `expected_size` is validated. The `apply_operators()` function itself does a check against the maximum length, and it is a waste of transaction fees and CU for the check not to be done in `update_cpi_digest()` instead

Recommendations:

Move the maximum `expected_size` validation from `apply_operators()` to `update_cpi_digest()`, or update the comment to reflect that it is not checked in `update_cpi_digest()`

Treehouse: Resolved with @b0e92dbef81...

Zenith: Verified.

[L-10] `validate_associated_token_accounts()` should use `InvalidAssociatedTokenAccount` as the error code

SEVERITY: Low	IMPACT: Low
STATUS: Resolved	LIKELIHOOD: Low

Target

- programs/boring-onchain-queue/src/utils/utils.rs#L74-L77
- programs/boring-onchain-queue/src/utils/validate.rs#L15-L18
- programs/boring-vault-svm/src/utils/teller.rs#L91-L98

Description:

The `validate_associated_token_accounts()` functions currently return an `InvalidTokenAccount` error code if the associated token account that the user provides does not match the one expected, given the intended owner. If a user provides an ATA owned by the right owner, but with the wrong token program, they'll get an `InvalidTokenAccount` even though they provided the correct token account.

Recommendations:

Use `InvalidAssociatedTokenAccount` for the flagged cases instead, since the validation is of the ATAs, not of the owners themselves, which are validated elsewhere.

Treehouse: Resolved with @9477732b718...

Zenith: Verified.

4.3 Informational

A total of 10 informational findings were identified.

[I-1] Limited usability of `view_cpi_digest` instruction

SEVERITY: Informational	IMPACT: Informational
STATUS: Resolved	LIKELIHOOD: Low

Target

- programs/boring-vault-svm/src/lib.rs#L1091-L1104

Description:

The `view_cpi_digest` instruction is intended to compute and return the expected digest for a given instruction with the specified operators applied. However, in order to return the desired CPI digest, the instruction already expects the correct hash data size to be passed. This might become a usability limitation since the `expected_size` is most likely still unknown at this point.

Recommendations:

It is recommended to alter the `view_cpi_digest` instruction such that it returns the CPI digest as well as the hash data size.

Treehouse: Resolved with @b0e92dbef8... by removing the expected_size check.

Zenith: Verified.

[I-2] Use of outdated `switchboard-on-demand` library

SEVERITY: Informational	IMPACT: Informational
STATUS: Resolved	LIKELIHOOD: Low

Target

- programs/boring-vault-svm/Cargo.toml#L25I

Description:

The protocol depends on `v0.2.2` of the `switchboard-on-demand` library although there already is the substantially newer `v0.3.5` available.

Recommendations:

It is recommended to upgrade to a newer version of the `switchboard-on-demand` library, if possible concerning compatibility.

Treehouse: Resolved with @15c9aa6c09...

Zenith: Verified.

[I-3] Exchange rate does not reflect owed fees

SEVERITY: Informational	IMPACT: Informational
STATUS: Acknowledged	LIKELIHOOD: Low

Target

- programs/boring-vault-svm/src/lib.rs#L631-L786

Description:

The exchange rate effectively establishes a relation between a vault's shares and its contained/withdrawable assets. On each exchange rate update, the owed fees (platform and performance fees) are computed and can be claimed from a vault. This reduces the amount of a vault's base assets, effectively reducing the exchange rate.

Recommendations:

It is recommended to ensure that a vault's owed fees are reflected in its current exchange rate, or update the documentation accordingly in case they already are.

Treehouse: Acknowledged. If fees are being collected regularly, then the pending fees should have a negligible impact on share price, but if for some reason fees are piling up and the admin is not collecting it is expected the strategist will adjust the exchange rate they provide to account for this. A comment has been added to call this out.

[I-4] Performance fees incentivize delayed exchange rate updates

SEVERITY: Informational	IMPACT: Informational
STATUS: Acknowledged	LIKELIHOOD: Low

Target

- programs/boring-vault-svm/src/lib.rs#L725-L750

Description:

A vault's performance fees are proportional to `(new_exchange_rate - high_water_mark) * share_supply`, whereby the `high_water_mark` is set to `new_exchange_rate` every time the performance fees are calculated on invocation of `update_exchange_rate`.

Assuming a growing share supply and an exchange rate provider who directly or indirectly profits from the performance fees, they are incentivized to delay exchange rate updates to apply `(new_exchange_rate - high_water_mark)` at the greatest possible `share_supply`.

Recommendations:

It is recommended to utilize the average share supply since the last performance computation on exchange rate update for a fairer performance fee computation, i.e. `(curr_share_supply + prev_share_supply) / 2`.

Treehouse: Acknowledged. It's more likely that the share supply decreases over time leading to the opposite incentive.

[I-5] Asset account not checked in `update_asset_data` instruction

SEVERITY: Informational	IMPACT: Informational
STATUS: Resolved	LIKELIHOOD: Low

Target

- programs/boring-vault-svm/src/lib.rs#L1303-L1304

Description:

The `asset` account of the `update_asset_data` instruction is expected to be checked within the instruction according to the comment in the `UpdateAssetData` context. However, the current implementation neglects to validate whether the `asset` is a zero account (native SOL) or a SPL-Token / Token-2022 mint.

Recommendations:

It is recommended to implement the intended checks of the `asset` account.

Treehouse: Resolved with @2e92f1c4c0...

Zenith: Verified.

[I-6] The `update_cpi_digest` instruction does not allow to update the CPI digest's properties

SEVERITY: Informational	IMPACT: Informational
STATUS: Resolved	LIKELIHOOD: Low

Target

- programs/boring-vault-svm/src/lib.rs#L1564-L1574

Description:

The `cpi_digest` account of the `update_cpi_digest` instruction is created with Anchor's `init` constraint. Therefore, the CPI digest's properties such as `operators` and `expected_size` cannot be updated.

Recommendations:

Typically, a change of the `operators` and `expected_size` leads to a change of the `cpi_digest` itself, eliminating the need to update its properties. Therefore, it is recommended to rename the instruction from `update_cpi_digest` to `initialize_cpi_digest`.

Treehouse: Resolved with @0761b9e205...

Zenith: Verified.

[I-7] Base asset mint not checked in `deploy` instruction

SEVERITY: Informational	IMPACT: Informational
STATUS: Resolved	LIKELIHOOD: Low

Target

- programs/boring-vault-svm/src/lib.rs#L1221-L1222

Description:

The `base_asset` mint account of the `deploy` instruction is expected to be checked within the instruction according to the comment in the `Deploy` context. However, the current implementation neglects to perform further custom validation of the `base_asset` mint, e.g. potential whitelist checks.

Recommendations:

It is recommended to implement the intended check of the `base_asset` mint or remove the comment.

Treehouse: Resolved with @47c701a18b...

Zenith: Verified.

[I-8] Project relies on vulnerable crate dependencies

SEVERITY: Informational	IMPACT: Informational
STATUS: Acknowledged	LIKELIHOOD: Low

Target

- Cargo.lock

Description:

The following vulnerable create dependencies were identified through `cargo audit`:

```
Crate:     curve25519-dalek
Version:   3.2.1
Title:     Timing variability in `curve25519-dalek`'s
    `Scalar29::sub`/`Scalar52::sub`
Date:      2024-06-18
ID:        RUSTSEC-2024-0344
URL:       https://rustsec.org/advisories/RUSTSEC-2024-0344
Solution:  Upgrade to ⩾4.1.3

Crate:     ed25519-dalek
Version:   1.0.1
Title:     Double Public Key Signing Function Oracle Attack on
    `ed25519-dalek`
Date:      2022-06-11
ID:        RUSTSEC-2022-0093
URL:       https://rustsec.org/advisories/RUSTSEC-2022-0093
Solution:  Upgrade to ⩾2

Crate:     ring
Version:   0.17.8
Title:     Some AES functions may panic when overflow checking is enabled.
Date:      2025-03-06
ID:        RUSTSEC-2025-0009
URL:       https://rustsec.org/advisories/RUSTSEC-2025-0009
Solution:  Upgrade to ⩾0.17.12
```

Recommendations:

It is recommended to upgrade the affected dependencies if applicable and viable concerning compatibility.

Treehouse: Acknowledged.

[I-9] `get_rate_in_quote()` will not work with SOL as the quote asset

SEVERITY: Informational	IMPACT: Informational
STATUS: Resolved	LIKELIHOOD: Low

Target

- programs/boring-vault-svm/src/lib.rs#L1847-L1848

Description:

The `deposit_sol()` instruction allows one to deposit native SOL to the vault, but there is no way to check the exchange rate because the `get_rate_in_quote()` instruction requires a `Mint` account, whereas the protocol uses the blank address (whose account is not owned by either token program) as the SOL address. Since the protocol does not support withdrawing SOL, the missing functionality in `get_rate_in_quote()` has no impact on the protocol itself, but adding it may help others to integrate and simplify their calculations.

Recommendations:

Consider modifying the function to support the blank address

Treehouse: Documented the behaviour with this commit

Zenith: Verified.

[I-10] Share token mint could benefit from associated metadata

SEVERITY: Informational	IMPACT: Informational
STATUS: Resolved	LIKELIHOOD: Low

Target

- programs/boring-vault-svm/src/lib.rs#L1211-L1219

Description:

Although the mint of the vault's share token (`share_mint`) is intended to be created with the Token-2022 program which supports extensions, no extensions are utilized.

Recommendations:

It is recommended to utilize the `metadata_pointer` extension to attach `TokenMetadata` (name, symbol, URI, etc.) to the vault's share mint.

Treehouse: Resolved with @93a0b481112... and @83ecdb1ab57...

Zenith: Verified