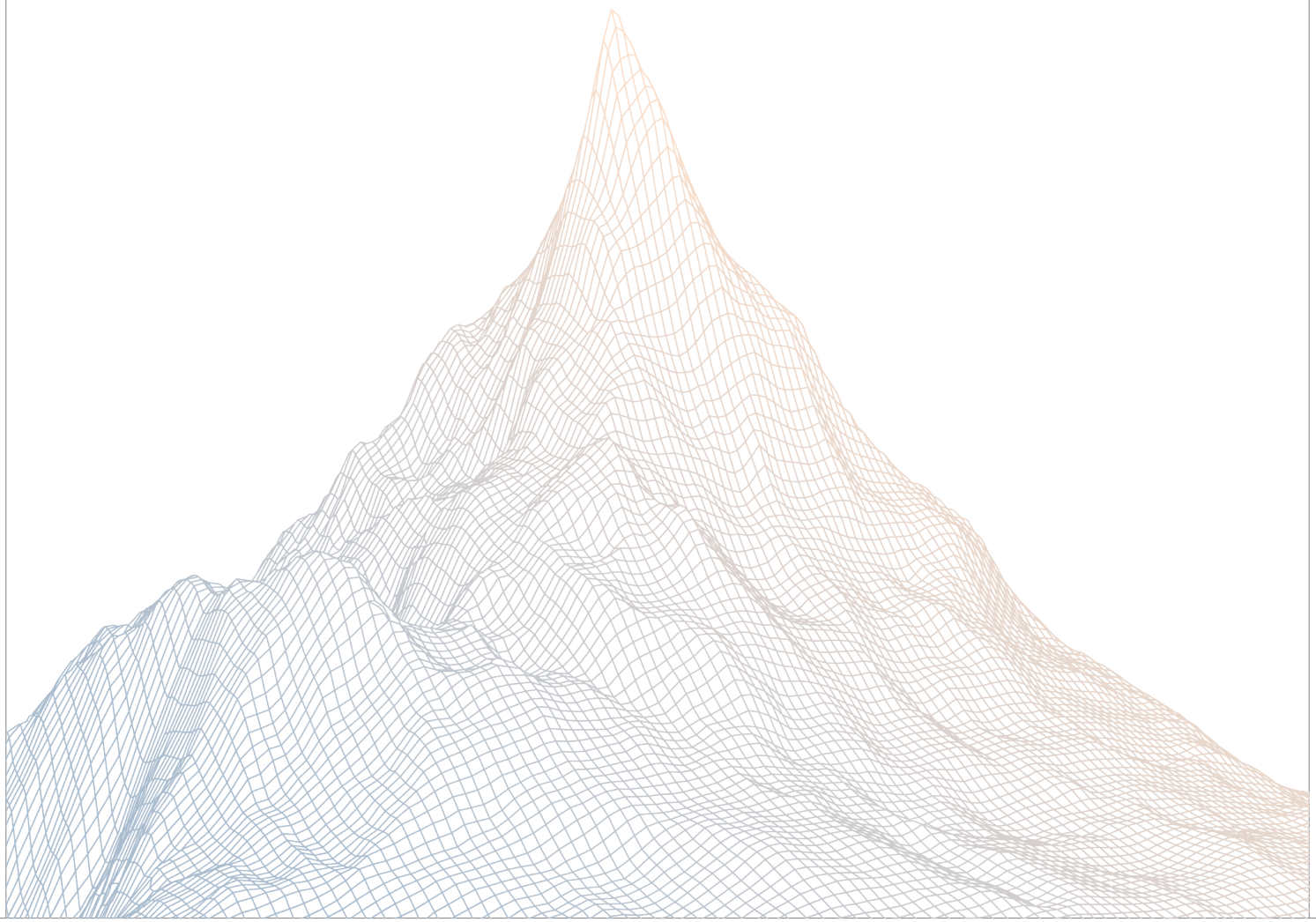
Zenith

MetaDAO

Smart Contract Security Assessment

VERSION 1.1



AUDIT DATES: January 5th to January 8th, 2026

AUDITED BY: IIIIIII
peakbolt

Contents

1 Introduction

1.1 About Zenith

Zenith assembles auditors with proven track records: finding critical vulnerabilities in public audit competitions.

Our audits are carried out by a curated team of the industry's top-performing security researchers, selected for your specific codebase, security needs, and budget.

Learn more about us at https://zenith.security.

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 MetaDAO

A fundraising and governance platform for high-quality founders and their communities.

2.2 Scope

The engagement involved a review of the following targets:

Target	programs
Repository	https://github.com/metaDAOproject/programs
Commit Hash	`df2a33abdfb4ea96ca91d4d9796fde3fcc90a56b`
Files	`programs/conditional_vault/src/**/*.rs`

2.3 Audit Timeline

January 5, 2026	Audit start
January 8, 2026	Audit end
January 14, 2026	Report published

2.4 Issues Found

SEVERITY	COUNT
Critical Risk	0
High Risk	0
Medium Risk	0
Low Risk	2
Informational	5
Total Issues	**7**

3

Findings Summary

ID	Description	Status
L-1	No way to close and reclaim rent for Question, ConditionalVault, and the vault's underlying token account	Acknowledged
L-2	Questions with many outcomes may be unusable	Resolved
I-1	Conditional vaults do not work with account delegates, hampering composability	Acknowledged
I-2	Unused account fields waste rent	Acknowledged
I-3	Misleading comments	Resolved
I-4	Account need not be marked as mutable	Resolved
I-5	Conditional tokens may be redeemed for zero underlying	Acknowledged

4

Findings

4.1 Low Risk

A total of 2 low risk findings were identified.

[L-1] No way to close and reclaim rent for `Question`, `ConditionalVault`, and the vault's underlying token account

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

Target

- src/instructions/initialize_question.rs
- src/instructions/initialize_conditional_vault.rs
- src/instructions/common.rs

Description:

The `Question`, `ConditionalVault`, and the vault's underlying token account never get closed, even though they are no longer useful once all tokens have been redeemed. This means that there is no way to reclaim these accounts' rent.

Recommendations:

Have a counter in each question, tracking the number of open vaults using the question, and close the question's account when the number reaches zero. Also have the vaults track whether all conditional tokens' supplies are zero, and in such cases, transfer any remaining underlying dust to the treasury or deployer ATA, and close the token account as well as the vault account.

MetaDAO: Acknowledged. I don't think the juice is worth the squeeze on this one. There's often conditional token balances for years after completion, so this would just add complexity w/o a huge benefit. It's also nice having this historical data in the onchain state.

[L-2] `Questions` with many outcomes may be unusable

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

Target

- src/instructions/initialize_question.rs
- src/instructions/initialize_conditional_vault.rs

Description:

During question initialization, the number of outcomes is required to be at least two, but there is no upper limit enforced. During `ConditionalVault` initialization, however, there's an implicit cap due to the fact that each outcome requires that a different token account be passed to the instruction, and there is a maximum number of accounts that may be passed in a single transaction. If there are more outcomes than are allowed by the tx's limit, the question will be able to be created, but will never be able to be used to initialize a vault, and will thus be wasted computation and rent.

In addition, the implicit cap currently prevents the `total_numerator` summation during redemption, and the `total_numerator` summation in the invariant check from overflowing when there are many outcomes. If the Solana account limits are ever greatly increase, these sums may become vulnerable to overflows.

Recommendations:

For legacy transactions, the limit is 34 accounts, and for address lookup table transactions, the limit is 64. Since `InitializeConditionalVault` uses 10 accounts, and two extra accounts are needed per outcome, a fair cap may be either 10 or 25.

```
require_gte!(args.num_outcomes, 2, VaultError::InsufficientNumConditions);
require_lte!(args.num_outcomes, 10, VaultError::ExcessiveNumConditions);
```

Also, use `checked_sum()`s during the flagged summations above, to prevent future issues.

MetaDAO: Resolved with `@9fb5b50f5b ...`

Zenith: Verified.

4.2 Informational

A total of 5 informational findings were identified.

[I-1] Conditional vaults do not work with account delegates, hampering composability

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

Target

- src/instructions/common.rs

Description:

Various places throughout the code require that account owners are the authority performing the operation, and that the full account balance is available to be consumed. This means that it is not possible for an account delegate to interact with conditional vaults, since they will not be the token account authority or owner, and the balance available to them may be less than the full balance.

Recommendations:

Consider relying on the SPL Token CPIs to do ownership validation, and change amount checks/burns to be based on the delegate's balance if the caller is a delegate. Also consider allowing minting of conditional tokens to accounts not owned by the minter.

MetaDAO: Acknowledged. Juice not worth the squeeze on this one - I'd rather constrain how you can use the program as much as possible to reduce attack surface and only open it up as there's a user need.

[I-2] Unused account fields waste rent

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

Target

- src/state/conditional_vault.rs

Description:

The `ConditionalVault.decimals` field is written, but isn't actually used anywhere. The same information is available in the underlying's mint, so the use of this field wastes rent, and consumes CUs during deserialization.

Recommendations:

Remove the field from the listed struct, and the code that sets its value.

MetaDAO: Acknowledged. We have decided to keep this as-is until we release a new version of `conditional_vault`.

[I-3] Misleading comments

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

Target

- src/instructions/add_metadata_to_conditional_tokens.rs
- src/state/conditional_vault.rs
- src/events.rs
- src/lib.rs

Description:

The following comments are misleading, which adds unnecessary cognitive load during reviews:

- In the `AddMetadataToConditionalTokens.validate()` function, there are checks for the vault's state, which no longer exists. If the instruction is meant to only be executable while the question remains unresolved, the `question` account should be added to the accounts struct, and the check should be done against `!self.question.is_resolved()`. Otherwise, the commented out code should be removed:

```
pub fn validate(&self) → Result<()> {
    // require!(
    //     self.vault.status == VaultStatus::Active,
    //     VaultError::VaultAlreadySettled
    // );
```

- In the `ConditionalVault.invariant()` function, the comment states that the assets must be greater than the liabilities, but the actual check requires that assets must be *at least* as much as liabilities (gte vs gt):

```
/// Checks that the vault's assets are always greater than its potential
/// liabilities. Should be called anytime you mint or burn conditional
/// tokens.
...
    // if the question isn't resolved, the vault should have more underlying
    // tokens than ANY conditional token mint's supply
```

```
    // if the question is resolved, the vault should have more underlying
    // tokens than the sum of the conditional token mint's supplies
    multiplied
    // by their respective payouts
```

- The `vault` is already a part of the event:

```
// TODO add `vault` to this event
#[event]
pub struct InitializeConditionalVaultEvent {
    pub common: CommonFields,
    pub vault: Pubkey,
```

- The `security_txt!()` attribute may need to update is source release and auditors. The `acknowledgements` field is also confusing since the conditional vault program doesn't use cash flows:

```
source_release: "v0.4",
auditors: "Neodyme (v0.3)",
acknowledgements: "DCF = (CF1 / (1 + r)^1) + (CF2 / (1 + r)^2) + ... (CFn /
    (1 + r)^n)"
```

Recommendations:

Update the comments to reflect the current behavior.

MetaDAO: Resolved with `@9e1859c7b8...` and `@abf6a2454f...`. Acknowledged for `security_txt` acknowledgements - I like using DCF for all of them.

Zenith: Verified.

[I-4] Account need not be marked as mutable

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

Target

- src/instructions/add_metadata_to_conditional_tokens.rs

Description:

The `AddMetadataToConditionalTokens.conditional_token_mint` account need not be mutable since the instruction doesn't modify its state, and the Metaplex program does not expect a writable account.

Recommendations:

Remove the `mut` modifier from the listed account.

MetaDAO: Resolved with `@131053bb2a ...`

Zenith: Verified.

[I-5] Conditional tokens may be redeemed for zero underlying

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

Target

- src/instructions/redeem_tokens.rs

Description:

It is possible for a user holding conditional tokens in a winning outcome, to receive zero tokens due to that outcome's numerator being very small compared to the denominator. In such cases, the user's tokens will be burned, but they'll be transferred zero underlying tokens.

Recommendations:

Prevent redemption if `total_redeemable` is zero

MetaDAO: Acknowledged. This is a feature because it allows us in our UI to just redeem for all possible token accounts, whereas if we reverted we'd need to check