ENIGMA DARK

Securing the Shadows

Managed Security Review

Alto Rewards

September 11, 2025

Contents

Summary

Enigma Dark
Enigma Dark is a web3 security firm leveraging the best talent in the space to secure all kinds of blockchain protocols and decentralized apps. Our team comprises experts who have honed their skills at some of the best auditing companies in the industry. With a proven track record as highly skilled white-hats, they bring a wealth of experience and a deep understanding of the technology and the ecosystem.

Learn more about us at enigmadark.com

Alto
Alto is a decentralized credit protocol built around DUSD. Isolated markets, a single stablecoin, and partial liquidations keep risk contained while giving you clear, sustainable ways to grow capital.

Engagement Overview

Over the course of 1 week, the Enigma Dark team conducted a security review of the Alto project. The review was performed by Kiki.

The following repositories were reviewed at the specified commits:

Repository	Commit
altomoney/v1	`9184b9df000e231b606161cf16be2cd08a723ccb`

Risk Classification

Severity	Description
High	Exploitable, causing loss or manipulation of assets or data.
Medium	Risk of future exploits that may or may not impact the smart contract execution.
Low	Minor code errors that may or may not impact the smart contract execution.
Informational	Non-critical observations or suggestions for improving code quality, readability, or best practices.

Vulnerability Summary

Severity	Count	Fixed	Acknowledged
High	0	0	0
Medium	0	0	0
Low	2	1	1
Informational	2	2	0

Findings

Index	Issue Title	Status
L-01	Lock Rewards With No Discount Due To Truncation	Acknowledged
L-02	Unwhitelisted Referrals Can Re-enable Status	Fixed
I-01	Implicit Merkle Root Acceptance Via Zero Value	Fixed
I-02	Missing NatSpec For AdaptersWhitelist Array	Fixed

Detailed Findings

High Risk

No issues found.

Medium Risk

No issues found.

Low Risk

L-01 - Lock Rewards With No Discount Due To Truncation

Severity: Low Risk

Context:

- contracts/rewards/AltoRewardsDistributor.sol#L144

Technical Details:
In the `claimRewardTokens` function, when a user opts to lock their rewards by setting `lockPurchasedRewards` to true, they receive an additional `lockBonusDiscount`. The total discount is calculated by summing the base discount, adapter discounts, and lock bonus discount. However, when this total exceeds `MAX_DISCOUNT` (0.5 ether), the function truncates it to `MAX_DISCOUNT` without considering the composition of the discount. This creates a scenario where if the base discount plus adapter discounts already exceed `MAX_DISCOUNT`, the `lockBonusDiscount` provides no actual benefit despite the user still being required to lock their tokens. For example, if base discount is 0.4 ether, adapter discounts are 0.1 ether, and lock bonus is 0.1 ether, the total 0.6 ether gets truncated to 0.5 ether, effectively nullifying the lock bonus while still forcing token lockup.

Impact:
Users lock their reward tokens for an extended period without receiving the promised discount benefit.

Recommendation:
Add validation to check if `discount` minus `lockBonusDiscount` exceeds `MAX_DISCOUNT` and revert if true, preventing users from locking tokens without discount benefit.

Developer Response: Acknowledged. The `MAX_DISCOUNT` will rarely be reached as the team will be shaping the discounts.

L-02 - Unwhitelisted Referrals Can Re-enable Status

Severity: Low Risk

Context:

- contracts/rewards/adapters/AltoReferralWhitelistAdapter.sol#L147

Technical Details:
In the `AltoReferralWhitelistAdapter` contract, addresses removed from the whitelist through `setReferralWhitelist` can re-enable their referral status by calling `activateReferralWhitelist` if the merkle root has not been updated. The `_verifyProof` check only validates against the current merkle root, which remains valid for previously whitelisted addresses until explicitly updated. This creates a window where removed referrals can bypass the administrative action of being unwhitelisted by re-activating their status using their old merkle proof. It also means that removing an address cannot be instant and persistent as changing the merkle root has to timelock on finalizing the update.

Impact:
Removed referrals maintain the ability to reinstate their whitelist status, undermining administrative control.

Recommendation:
Update merkle root when removing referrals and reapply setReferralWhitelist with false after the merkle update to ensure complete removal.

Developer Response: Fixed at commit `4cf1809`.

Informational

I-01 - Implicit Merkle Root Acceptance Via Zero Value

Severity: Informational

Context:

- contracts/rewards/MerkleRootManager.sol#L74
- contracts/rewards/MerkleRootManager.sol#L96

Technical Details:
The `acceptPendingRoot` and `revokePendingRoot` functions in the `MerkleRootManager` contract allow implicit acceptance of any pending root by passing bytes32(0). The validation checks in both functions use a pattern that bypasses explicit verification when a zero value is provided: `merkleRootToAccept != bytes32(0) && merkleRootToAccept != pendingMerkleRoot.merkleRoot`. Since the pending root is public and verifiable, this implicit acceptance through zero values undermines the explicit verification intent of these functions. The same issue exists in the revocation function with identical logic.

Impact:
Administrators can accidentally accept or revoke incorrect merkle roots by passing zero values, bypassing the intended explicit verification mechanism.

Recommendation:
Modify both functions to require explicit verification by reverting if the provided merkle root is zero or does not match the pending root.

Developer Response: Fixed at commit `ecb1b66`.

I-02 - Missing NatSpec For AdaptersWhitelist Array

Severity: Informational

Context:

- contracts/rewards/AltoRewardsDistributor.sol#L58

Technical Details:
The `adaptersWhitelist` array in the `AltoRewardsDistributor` contract lacks NatSpec documentation. This array stores addresses of whitelisted adapters that can provide bonus reward information for users. While other state variables in the contract have proper NatSpec documentation using `@inheritdoc` or `@notice` tags, this important array lacks any documentation about its purpose, contents, or usage.

Impact:
Lack immediate clarity about the array, reducing code readability.

Recommendation:
Add `documentation to describe the array's purpose, contents,

Developer Response: Fixed at commit `dac6a1e`.

Disclaimer

This report does not endorse or critique any specific project or team. It does not assess the economic value or viability of any product or asset developed by parties engaging Enigma Dark for security assessments. We do not provide warranties regarding the bug-free nature of analyzed technology or make judgments on its business model, proprietors, or legal compliance.

This report is not intended for investment decisions or project participation guidance. Enigma Dark aims to improve code quality and mitigate risks associated with blockchain technology and cryptographic tokens through rigorous assessments.

Blockchain technology and cryptographic assets inherently involve significant risks. Each entity is responsible for conducting their own due diligence and maintaining security measures. Our assessments aim to reduce vulnerabilities but do not guarantee the security or functionality of the technologies analyzed.

This security engagement does not guarantee against a hack. It is a review of the codebase during a specific period of time. Enigma Dark makes no warranties regarding the security of the code and does not warrant that the code is free from defects. By deploying or using the code, the project and users of the contracts agree to use the code at their own risk. Any modifications to the code will require a new security review.