Aztec Uniswap TWAP

Security Review

Cantina Managed review by:
Eric Wang, Lead Security Researcher
Jonatas Martins, Associate Security Researcher

December 9, 2025

Contents

1 Introduction

1.1 About Cantina

Cantina is a security services marketplace that connects top security researchers and solutions with clients. Learn more at cantina.xyz

1.2 Disclaimer

Cantina Managed provides a detailed evaluation of the security posture of the code at a particular moment based on the information available at the time of the review. While Cantina Managed endeavors to identify and disclose all potential security issues, it cannot guarantee that every vulnerability will be detected or that the code will be entirely secure against all possible attacks. The assessment is conducted based on the specific commit and version of the code provided. Any subsequent modifications to the code may introduce new vulnerabilities that were absent during the initial review. Therefore, any changes made to the code require a new security review to ensure that the code remains secure. Please be advised that the Cantina Managed security review is not a replacement for continuous security measures such as penetration testing, vulnerability scanning, and regular code reviews.

1.3 Risk assessment

Severity level	Impact: High	Impact: Medium	Impact: Low
Likelihood: high	Critical	High	Medium
Likelihood: medium	High	Medium	Low
Likelihood: low	Medium	Low	Low

1.3.1 Severity Classification

The severity of security issues found during the security review is categorized based on the above table. Critical findings have a high likelihood of being exploited and must be addressed immediately. High findings are almost certain to occur, easy to perform, or not easy but highly incentivized thus must be fixed as soon as possible.

Medium findings are conditionally possible or incentivized but are still relatively likely to occur and should be addressed. Low findings are a rare combination of circumstances to exploit, or offer little to no incentive to exploit but are recommended to be addressed.

Lastly, some findings might represent objective improvements that should be addressed but do not impact the project's overall security (Gas and Informational findings).

2 Security Review Summary

Aztec Labs was founded in 2017, and has a team of +50 leading zero-knowledge cryptographers, engineers, and business experts. Aztec Labs is developing its namesake products: AZTEC (a privacy-first L2 on Ethereum) and NOIR (the universal ZK language).

From Oct 20th to Nov 5th the Cantina team conducted a review of `uniswap-token-launcher`, `ignition-monorepo` and `teegeeee` at commit hashes `e606175e`, `8f2c07db` and `8718df6a` respectively. The team identified a total of **24** issues:

Issues Found

Severity	Count	Fixed	Acknowledged
Critical Risk	0	0	0
High Risk	0	0	0
Medium Risk	3	2	1
Low Risk	11	5	6
Gas Optimizations	0	0	0
Informational	10	4	6
Total	**24**	**11**	**13**

2.1 Scope

The security review had the following components in scope for Aztec - Uniswap Twap Audit on commit hash 65685c19:

```
├── ignition-monorepo
│   └── contracts
│       └── src
│           ├── constants.sol
│           ├── sale
│           │   ├── GenesisSequencerSale.sol
│           │   └── IGenesisSequencerSale.sol
│           ├── soulbound
│           │   ├── IgnitionParticipantSoulbound.sol
│           │   ├── IIgnitionParticipantSoulbound.sol
│           │   └── providers
│           │       ├── AttestationProvider.sol
│           │       ├── IWhitelistProvider.sol
│           │       ├── PredicateProvider.sol
│           │       └── ZKPassportProvider.sol
│           ├── staking
│           │   ├── ATPNonWithdrawableStaker.sol
│           │   ├── ATPWithdrawableAndClaimableStaker.sol
│           │   ├── ATPWithdrawableStaker.sol
│           │   ├── interfaces
│           │   │   ├── IATPNonWithdrawableStaker.sol
│           │   │   ├── IATPWithdrawableAndClaimableStaker.sol
│           │   │   ├── IATPWithdrawableStaker.sol
│           │   │   └── IGovernanceATP.sol
│           │   └── rollup-system-interfaces
│           │       ├── IGovernance.sol
│           │       ├── IGSE.sol
│           │       ├── IRegistry.sol
│           │       └── IStaking.sol
│           ├── staking-registry
│           │   ├── libs
│           │   │   ├── BN254.sol
│           │   │   └── QueueLib.sol
```

```
│           │   └── StakingRegistry.sol
│           └── uniswap-periphery
│               ├── AuctionHook.sol
│               ├── nonces
│               │   └── Nonces.sol
│               └── VirtualAztecToken.sol
├── teegeeee
│   └── src
│       ├── ATPFactory.sol
│       ├── ATPFactoryNonces.sol
│       ├── Nonces.sol
│       ├── Registry.sol
│       ├── atps
│       │   ├── linear
│       │   │   └── IATP.sol
│       │   ├── linear
│       │   │   ├── ILATP.sol
│       │   │   ├── LATP.sol
│       │   │   └── LATPCore.sol
│       │   └── noclaim
│       │       ├── INCATP.sol
│       │       └── NCATP.sol
│       ├── libraries
│       │   └── LockLib.sol
│       ├── Nonces.sol
│       ├── Registry.sol
│       ├── staker
│       │   └── BaseStaker.sol
│       └── token
│           └── Aztec.sol
└── uniswap-token-launcher
    └── src
        ├── distributionContracts
        │   ├── LBPStrategyBasic.sol
        │   └── VirtualLBPStrategyBasic.sol
        ├── distributionStrategies
        │   └── VirtualLBPStrategyFactory.sol
        └── utils
            └── HookBasic.sol
```

3 Findings

3.1 Medium Risk

3.1.1 Non-compliant addresses may receive ATPs from the Genesis Sequencer Sale

Severity: Medium Risk

Context: GenesisSequencerSale.sol#L162-L174

Description: In `GenesisSequencerSale`, the `purchaseAndMintSoulboundToken()` function calls `mintFromSale()` on the `IgnitionParticipantSoulbound` contract to mint a Soulbound token to the caller and then creates an ATP to `_atpBeneficiary`, specified in the arguments.

The code skips the screening check on `_atpBeneficiary`, but note that the screening checks in the Soulbound contract apply only to the caller. Therefore, it is possible that `_atpBeneficiary` is non-compliant but still receives an ATP.

Recommendation: Consider updating the `_internalPurchase()` function to perform an address screening check when `_atpBeneficiary` is not the caller. This relies on the fact that the Soulbound token contract already performs the check on the caller, who receives the Soulbound token.

Aztec: Fixed in `ignition-monorepo` PR 347.

Cantina Managed: Fix verified.

3.1.2 Potential bypass of staking requirements due to failed deposits

Severity: Medium Risk

Context: ATPWithdrawableAndClaimableStaker.sol#L106-L113

Description: As mentioned in the NatSpec of the `moveFundsBackToATP()` function, the staking operation could potentially fail under certain conditions,

```
/*
 * Case in which this is required:
 * - When calling deposit with _moveWithLatestRollup set to true, the staker will enter
 ↪  the deposit queue
 * - If user gets to the front of the queue, but the rollup has been upgraded,
 ↪  _moveWithLatestRollup will be invalid
 * - This will return the funds to the withdrawer (this address)
 */
```

If depositing into the rollup fails, the funds will be returned to the `ATPWithdrawableAndClaimableStaker` contract, but the `$.hasStaked` variable will remain `true`, even though the user did not actually stake. If the user can deliberately trigger a deposit failure or time the deposit to guarantee it fails, they may exploit this to bypass the staking requirements.

Recommendation: Instead of setting the flag in the stake() function, consider one of the following approaches:

- Implement another function on the Staker contract, e.g., `confirmStaked()`, which fetches the state from the rollup and checks whether the user has successfully staked. If so, set the flag to `true`.
- Track and update the staked status only during withdrawal to ensure the user completed the deposit.

Aztec: There are too many reasons that a deposit can fail, since the validator will not be losing funds, we will not fix.

Cantina Managed: Acknowledged.

3.1.3 `currency` should be compared to `poolToken` instead of `token`

Severity: Medium Risk

Context: LBPStrategyBasic.sol#L289-L290

Description: The `LBPStrategyBasic` contract implements a `getPoolToken()` function, which returns the actual token used to create the pool. The two tokens in the created Uniswap V4 pool are `currency` and `poolToken`.

In the `_prepareMigrationData()` function, when calculating liquidity, `currency` is incorrectly compared to `token` rather than `poolToken`, which may yield incorrect results if `currency` and `token` result in a different order.

Recommendation: Consider using `currency < poolToken` instead of `currency < token`.

Aztec: Fixed in `uniswap-token-launcher` commit 3cb057a.

Cantina Managed: Fix verified.

3.2 Low Risk

3.2.1 Missing input validation on Grid Tile IDs

Severity: Low Risk

Context: IgnitionParticipantSoulbound.sol#L376-L378

Description: In the `_internalMint()` function of the `IgnitionParticipantSoulbound` contract, a `_gridTileId` is assigned to the `_soulboundRecipient`.

However, a check `_gridTileId > 0` is missing. Without the check, it is possible to assign grid tile ID 0 to a recipient first, and then assign another ID > 0 to it. This will result in the recipient holding two Soulbound NFT tokens, which can be the same token ID or not.

Recommendation: Consider adding a `_gridTileId > 0` check in the `_internalMint()` function.

Aztec: Fixed in `ignition-monorepo` PR 352.

Cantina Managed: Fix verified. Note that after the changes for Issue "Frontrunning force minting to `_soulboundRecipient`", recipients are now allowed to hold multiple Soulbound NFT tokens, which is acceptable, according to the protocol team.

3.2.2 `TokenLauncher` allows arbitrary factory and strategy input parameters

Severity: Low Risk

Context: TokenLauncher.sol#L35, TokenLauncher.sol#L50

Description: The `TokenLauncher` contract deploys new tokens through the `createToken` function and distributes them through the `distributeToken` function. The issue is that `createToken` accepts an arbitrary factory address as input. A malicious address could cause incorrect addresses to be emitted in events. Similarly, in `distributeToken`, the `strategy` is set through the user-provided `distribution` parameter. This arbitrary input could lead to incorrect event emissions or allow extraction of tokens sent to the contract.

Recommendation: Consider implementing a whitelist for `factory` addresses in `createToken` and another whitelist for `strategy` addresses used in `distributeToken`.

Aztec: Acknowledged.

Cantina Managed: Acknowledged.

3.2.3 Missing deadline check in signature in `VirtualAztecToken`

Severity: Low Risk

Context: VirtualAztecToken.sol#L252

Description: The `setAtpBeneficiaryWithSignature` function lacks a deadline and nonce cancellation mechanism. If a user signs a change to their ATP beneficiary, that signature can be used at any time in the future, even if the user has already changed the beneficiary through `setAtpBeneficiary` function.

Recommendation: Add a deadline check for the signature.

Aztec: Fix in `ignition-monorepo` PR 359.

Cantina Managed: Fix verified.

3.2.4 `VirtualLBPStrategyBasic` missing checks on overridden function

Severity: Low Risk

Context: VirtualLBPStrategyBasic.sol#L74-L79

Description: The `VirtualLBPStrategyBasic` overrides the `_validateMigration` implementation of `LBPStrategyBasic` but is missing important checks. It lacks a check to ensure `currencyAmount` is not zero and fails to call `auction.checkpoint` to get the most updated `currencyRaised` value.

Recommendation: Consider using the overridden function that guarantees the checks are performed and additionally add the check for `isMigrationApproved`.

Aztec: Fixed in `uniswap-token-launcher` PR 2.

Cantina Managed: Fix verified.

3.2.5 Frontrunning force minting to `_soulboundRecipient`

Severity: Low Risk

Context: IgnitionParticipantSoulbound.sol#L419

Description: The `IgnitionParticipantSoulbound` contract mints one NFT per address. IDs 0 and 1 are only minted to users already set in `genesisSequencerMerkleRoot` and `contributorMerkleRoot`, respectively. ID 3 is minted for general users. The minting address is determined by the `_soulboundRecipient` variable, input by the user.

A malicious user can frontrun the minting of ID 0 or 1 to a specific `_soulboundRecipient` address, forcing ID 3 to be minted to that address instead. If the `_soulboundRecipient` address cannot be changed, this causes a DOS for the user and soulbound address.

The likelihood of this attack is low, considering the caller must go through KYC before calling it. However, a malicious user can still execute it.

Recommendation: Consider adding an approval mechanism where the `_soulboundRecipient` address must approve the caller before minting tokens.

Aztec: Fixed in `ignition-monorepo` PR 608.

Cantina Managed: Fix by removing the `require` of not having a `gridTileId` set. With the fix, the `gridTileId[_soulboundRecipient]` will always be replaced with `_gridTileId`, and now it's possible to mint multiple NFTs for the `_soulboundRecipient`.

3.2.6 `ATPWithdrawableAndClaimableStaker` doesn't count deposits made through `ATPNonWithdrawableStaker`

Severity: Low Risk

Context: ATPWithdrawableAndClaimableStaker.sol#L98

Description: The staker contracts are implementation contracts that can be updated through the ATP contracts. When a user stakes through `ATPNonWithdrawableStaker` and later updates to `ATPWithdrawableAndClaimableStaker`, they cannot call `withdrawAllTokensToBeneficiary` unless they withdraw and stake again to set `hasStaked` variable to true.

The sequence of actions:

1. User calls `ATPNonWithdrawableStaker.stake`.
2. Update staker to `ATPWithdrawableAndClaimableStaker`.
3. Initiate and finalize the withdraw process.
4. User calls `ATPWithdrawableAndClaimableStaker.stake`.
5. Initiate and finalize withdraw.
6. User can then withdraw through `ATPWithdrawableAndClaimableStaker.withdrawAllTokensToBeneficiar`

Recommendation: Consider moving the `hasStaked` variable to `ATPNonWithdrawableStaker`.

Aztec: We will be deleting `ATPNonWithdrawableStaker` and not deploying it as mediation.

Cantina Managed: Acknowledged, we agree the issue doesn't happen if `ATPNonWithdrawableStaker` is not deployed.

3.2.7 Regarding the enforcement of staking based on mint amounts

Severity: Low Risk

Context: VirtualAztecToken.sol#L396-L408

Description: According to the `_mintAtp()` function, if a user mints more than `MIN_STAKE_AMOUNT` tokens, they need to stake them into the rollup and therefore will receive an NCATP instead of LATP.

While the contract enforces the creation of NCATP for large mints, note that it is still possible for a user to split the total amount into multiple smaller amounts (each less than `MIN_STAKE_AMOUNT`) so that staking is not needed.

Recommendation: The protocol team is aware of the issue, which the frontend will mitigate.

Aztec: Acknowledged.

Cantina Managed: Acknowledged.

3.2.8 Missing input validation on migrator parameters

Severity: Low Risk

Context: LBPStrategyBasic.sol#L183-L185

Description: Consider adding the following checks for `MigratorParameters`:

1. `operator != address(0)`.
2. `block.number < sweepBlock` and `block.number < migrationBlock`.

Recommendation: Consider implementing the above checks.

Aztec: Acknowledged, forwarded to Uniswap team.

Cantina Managed: Acknowledged.

3.2.9 `getValueAtIndex()` may return out-of-bounds values

Severity: Low Risk

Context: QueueLib.sol#L38-L44

Description: In the `getValueAtIndex()` function of `QueueLib`, the `_index` parameter is not validated to be `_self.first <= _index < _self.last`. Therefore, this function can return old values or uninitialized values.

Recommendation: Consider validating `_index` and reverting with an error if the value is out of bounds.

Aztec: Fixed in `ignition-monorepo` PR 610.

Cantina Managed: Fix verified.

3.2.10 Operator can sweep tokens regardless of whether the migration has occurred

Severity: Low Risk

Context: LBPStrategyBasic.sol#L156-L158

Description: It is expected that the operator of the strategy contract will call `migrate()` after the auction ends, and then call `sweepToken()` after the sweep block to transfer out any remaining tokens, if necessary.

Note that the operator can still call `sweepToken()` even if the migration did not happen, as long as the sweep block has passed.

Recommendation: Consider whether it is necessary to prevent the operator from sweeping the token if the migration has not occurred. If so, there can be a check in `sweepToken()` to ensure the migration has been done, e.g., by setting a flag in `migrate()`.

Aztec: Acknowledged, forwarded to Uniswap team.

Cantina Managed: Acknowledged.

3.2.11 Additional checks to enhance code robustness

Severity: Low Risk

Context: VirtualAztecToken.sol#L211-L217, LBPStrategyBasic.sol#L118

Description:

1. Users call the `VirtualAztecToken.sweepIntoAtp()` function to mint an ATP with a balance of their `$pendingAtpBalances`. A zero balance would make the ATP creation fail due to a check in `LATPCore`. However, the code could be made more robust by skipping the `_mintAtp()` call if the pending balance is zero.
2. The `LBPStrategyBasic.onTokensReceived()` function has no access control. The expected use case is to create the strategy contract and fund it atomically through `TokenLauncher`. Even though a second call to the `onTokensReceived()` function will fail since the call to `initializeDistribution()` will revert, there could be an explicit flag in `onTokensReceived()` that ensures the function can be executed only once.
3. Add a comment in the `GenesisSequencerSale` that explaining that each `_beneficiary` receiving the genesis sale can only create a single NCATP per `_beneficiary` address, in the `_internalPurchase` function.

Recommendation: Consider implementing the above checks.

Aztec: Acknowledged.

Cantina Managed: Acknowledged.

3.3 Informational

3.3.1 Suggestions on NatSpec and code comments

Severity: Informational

Context: GenesisSequencerSale.sol#L19, GenesisSequencerSale.sol#L154, IgnitionParticipantSoulbound.sol#L116-L118, IgnitionParticipantSoulbound.sol#L405, StakingRegistry.sol#L274-L279

Description:

1. GenesisSequencerSale.sol#L19: Update the comments since the created ATPs are not revokable.
2. GenesisSequencerSale.sol#L154: Add NatSpec for the `_gridTileId` parameter.
3. IgnitionParticipantSoulbound.sol#L116-L118: Add NatSpec for the `_gridTileId` parameter.
4. IgnitionParticipantSoulbound.sol#L405: Change `msg.sender` to `_identityAddress` since `_identityAddress` may not necessarily be `msg.sender`. `msg.sender` can be the token sale contract.
5. StakingRegistry.sol#L274-L279: Add NatSpec for the `_providerRewardsRecipient` parameter.
6. GovernanceAcceleratedLock.sol#L34-L42: Consider clarifying in the docs or comments that the lock status, i.e., `lockAccelerated()`, can be changed by the Governance anytime.
7. GovernanceAcceleratedLock.sol#L44: Consider clarifying in the docs or comments that the `relay()` function cannot send native tokens while calling the target function.

Recommendation: Consider implementing the above suggestions.

Aztec: Fixed in `ignition-monorepo` PR 613.

Cantina Managed: Fix verified.

3.3.2 Unused code

Severity: Informational

Context: IIgnitionParticipantSoulbound.sol#L35, NCATP.sol#L13, BaseStaker.sol#L23

Description:

1. BaseStaker.sol#L23: The `UnSupportedOperation()` error is unused.
2. IIgnitionParticipantSoulbound.sol#L35: The `IgnitionParticipantSoulbound__InvalidTokenId()` error is unused.
3. NCATP.sol#L13: The immutable variable `CREATED_AT_TIMESTAMP` is unused.
4. GovernanceAcceleratedLock.sol#L8: The `GovernanceAcceleratedLock__NotGovernance()` error is unused.
5. AuctionHook.sol#L66: The variable `addressScreeningProvider` and the event `AdderssScreeningProviderSet` are not used in the `AuctionHook` contract.

Recommendation: Consider removing the unused code if not needed.

Aztec: Fixed in `ignition-monorepo` PR 612.

Cantina Managed: Fix verified.

3.3.3 Consider returning 0 or revert in `getClaimable()` of NCATP

Severity: Informational

Context: NCATP.sol#L19

Description: The `claim()` function of NCATP is overridden to prevent the holder from claiming without staking. To be consistent, the contract could also override the `getClaimable()` function to return 0 or revert. Currently, the `getClaimable()` function may return a non-zero value after the global lock period ends.

Recommendation: Consider overriding the `getClaimable()` to return 0 or revert to be consistent.

Aztec: Acknowledged.

Cantina Managed: Acknowledged.

3.3.4 Factory mismatch may cause DOS in `VirtualAztecToken`

Severity: Informational

Context: VirtualAztecToken.sol#L396-L408

Description: The factories used in `VirtualAztecToken` for stake amounts and low amounts should be `ATPFactoryNonces` , currently set to `ATPFactory`. Otherwise, they might collide with the salt and DOS the creation. This could happen if a user already receives Aztec tokens through Sales and buys the same amount from Auction.

Recommendation: Consider changing the `IATPFactory` to `IATPFactoryNonces`, explicitly setting it to a nonce factory.

Aztec: Product requirements on the lockup have been updated such that their lockups will be identical, so we are now using the same ATP factory - but NCATP / LATP depending on amount purchased. NCATP forcing people to stake, and LATP just being access to their tokens. Fix in `ignition-monorepo` PR 351.

Cantina Managed: Fix verified.

3.3.5 Misleading comments in `ATPNonWithdrawableStaker`

Severity: Informational

Context: ATPNonWithdrawableStaker.sol#L213-L215

Description: The comment for the `moveFundsBackToATP` function describes an incorrect flow:

```
/*
 * - This leaves the user with two options:
 *   - Call stake again with an updated rollup version
 *   - return the funds back to the atp
 */
```

The user cannot stake directly because the funds are held in the Staker contract, and depositing pulls funds from ATP. The only option is to send the funds back to ATP before staking again.

Recommendation: Fix the comment to reflect that the only available flow is returning tokens to ATP when the deposit fails.

Aztec: Fixed in `ignition-monorepo` PR 611.

Cantina Managed: Fix verified.

3.3.6 Add finalise withdraw from governance in `ATPNonWithdrawableStaker`

Severity: Informational

Context: ATPNonWithdrawableStaker.sol#L179

Description: The `ATPNonWithdrawableStaker` contract has an `initiateWithdrawFromGovernance` function but lacks a corresponding `finaliseWithdrawFromGovernance` function. While users can call the governance contract directly to finalize withdrawals, `WithdrawableStaker` implements similar functionality through its `finaliseWithdraw` function for the rollup.

Recommendation: Consider adding the `finaliseWithdrawFromGovernance` function for consistency.

Aztec: Acknowledged.

Cantina Managed: Acknowledged.

3.3.7 Use `SafeCast` when casting variables to a smaller type

Severity: Informational

Context: VirtualLBPStrategyFactory.sol#L38

Description: In the `initializeDistribution()` function of `VirtualLBPStrategyFactory`, the `totalSupply` variable is explicitly cast from type `uint256` to `uint128`. It is considered best practice to use `SafeCast.toUint128()` instead of an explicit cast when truncation of data is not expected. This also applies to L66.

Recommendation: Consider using the `SafeCast` library functions when casting variables to a smaller type.

Aztec: Acknowledged, forwarded to Uniswap team.

Cantina Managed: Acknowledged.

3.3.8 Use `SafeERC20` for ERC-20 token transfers and approvals

Severity: Informational

Context: VirtualAztecToken.sol#L165

Description: Instead of the raw `transfer()` and `transferFrom()`, it is best practice to use `safeTransfer()` or `safeTransferFrom()` in case the transferred token does not comply with the standard. Also applies for L313, L333, L398, L404.

For `approve()`, the best practice is to replace it with `forceApprove()`. See L145, L239, L255 in `ATPNonWithdrawableStaker.sol`, and L216 in `StakingRegistry.sol`.

Using `SafeERC20` functions is not needed if the token is guaranteed to comply with the ERC20 standard.

Recommendation: Consider using `SafeERC20` functions if necessary.

Aztec: Acknowledged, Aztec token complies with the ERC20 standard.

Cantina Managed: Acknowledged.

3.3.9 Staking duration for NCATP holders is not enforced

Severity: Informational

Context: ATPWithdrawableAndClaimableStaker.sol#L98

Description: Holders of NCATP are required to stake their tokens into the rollup to withdraw them after a withdrawal timestamp. Note that the contracts do not enforce the user's staking period. In theory, users can stake and unstake immediately, only waiting for the exit delay. Even though, they will still need to wait until the withdrawal timestamp to withdraw the locked tokens.

Recommendation: After discussing with the protocol team, it is clarified that there are no hard requirements on the staking duration, but users are incentivized to stake for a long enough period.

Aztec: Acknowledged.

Cantina Managed: Acknowledged.

3.3.10 [AZTEC] Additional changes during the review

Severity: Informational

Context: AuctionHook.sol#L113-L120, ATPFactory.sol#L97

Description: During the review, the Aztec team provided additional fixes that were reviewed. The following changes were included:

- teegeeee PR 103: Reduced deployment size by refactoring the implementation to use libraries.
- ignition-monorepo PR 353: Simplified ETH amount checking and updated logic to reflect the new auction behavior, which no longer passes the `Q96` amount.

Cantina Managed: These changes have been reviewed and verified.