OpenZeppelin | security

Aave aToken Vault Audit

March 1, 2023

This security assessment was prepared by OpenZeppelin.

Table of Contents

Summary

Type	DeFi
Timeline	From 2023-02-06 To 2022-02-10
Languages	Solidity

Total Issues	8 (5 resolved, 1 partially resolved)
Critical Severity Issues	0 (0 resolved)
High Severity Issues	0 (0 resolved)
Medium Severity Issues	1 (1 resolved)
Low Severity Issues	1 (0 resolved)
Notes & Additional Information	6 (4 resolved, 1 partially resolved)

Scope

We audited the aave/wrapped-atoken-vault repository at the `f39678dd1279ff9d8bcc13dfa6fd0eaf75037b69` commit.

In scope were the following contracts:

```
src
├── ATokenVault.sol
├── ATokenVaultStorage.sol
├── interfaces
│   └── IATokenVault.sol
└── libraries
    ├── Constants.sol
    └── MetaTxHelpers.sol
```

System Overview

The `ATokenVault` is an ERC-4626 vault with an underlying ERC-20 token that is supported by the Aave-v3 pool. Users can deposit/withdraw the underlying token to the vault to mint/burn vault shares. Under the hood, the vault calls the Aave-v3 pool to `supply`/`withdraw` the corresponding underlying tokens, and consequently mint/burn the equivalent amount of `aToken` to its balance. The vault has no privilege to withdraw any underlying assets on its own nor change any default user configuration settings with respect to the Aave-v3 pool. Additionally, the vault accepts deposits/withdrawals in `aToken` directly. The share accounting is performed with respect to the `aToken` balance, and not the underlying ERC-20 token balance. The vault accrues yield in `aToken` balance and takes a fee cut before any state-updating contract interaction. The vault's access to underlying tokens is restricted by the corresponding Aave-v3 pool states and relevant liquidity positions, and such conditions may affect the deposit and withdrawal limits.

Privileged Roles

The vault has an `owner` role that can set the fee up to 100% of the accrued interest, withdraw accumulated fees and claim any relevant Aave-v3 rewards. The `owner` can also rescue any ERC-20 tokens (other than the vault `aToken`) to any address from this vault.

Findings

Here we present our findings.

Medium Severity

M-01 Some functions do not conform to EIP-4626 specifications

The following functions do not conform to the EIP-4626 specifications:

- The `maxWithdraw` function in EIP-4626 should specify the amount of the underlying asset, in this case the ERC-20 token instead of the corresponding aToken, that can be withdrawn from the owner's balance through a `withdraw` call. The implementation is required to satisfy both the global and user-specific limits and MUST return 0 if disabled (even temporarily). However, the `maxWithdraw` function inherited from the base implementation only gives the limit in terms of the aToken amount, not the underlying token amount. In the case where `withdraw` from the pool would fail due to the pool configuration (e.g., the pool is paused), withdrawing from the vault would also fail. In such cases, the `maxWithdraw` should return 0 instead.
- The `previewWithdraw` in EIP-4626 should allow a user to simulate the effects of their withdrawal at the current block, given current on-chain conditions. However the `previewWithdraw` inherited from the base implementation does not account for the corresponding Aave-v3 pool states and liquidity position in that block. This applies similarly to the `previewDeposit`, `previewMint` and `previewRedeem` functions.

Consider incorporating the pool states and liquidity positions into the affected functions.

***Update**: Resolved in pull request #61 at commits 04dda15 and ce4e07b.*

Low Severity

L-01 Deposits of fee-on-transfer tokens will fail

When depositing underlying tokens to the vault, an amount is transferred from the depositor to the vault and the same amount is then supplied to the pool. If the amount of underlying tokens that the vault receives is less than what is specified in the transfer (for example, if the underlying token is a fee-on transfer type), then supplying to the pool will revert and the deposit operation will fail.

Consider only supplying the actual amount transferred to the vault and updating the stored vault balance by the actual amount transferred.

Update: *Acknowledged, not resolved. The Aave team stated:*

> *This is intended due to the fact that the Aave Protocol does not support fee-on-transfer tokens either. In case the Aave Protocol does, we could upgrade the vault implementation with custom logic to fit tokens with this behavior.*

Notes & Additional Information

N-01 Files specifying outdated Solidity versions

Throughout the codebase there are `pragma` statements that use an outdated version of Solidity. For instance:

- The `pragma` statement on line 2 of `ATokenVault.sol`
- The `pragma` statement on line 2 of `ATokenVaultStorage.sol`
- The `pragma` statement on line 2 of `IATokenVault.sol`
- The `pragma` statement on line 1 of `Constants.sol`
- The `pragma` statement on line 2 of `MetaTxHelpers.sol`

Consider taking advantage of the latest Solidity version to improve the overall readability and security of the codebase. Regardless of which version of Solidity is used, consider pinning the version consistently throughout the codebase to prevent opening bugs due to incompatible future releases.

Update*: Partially resolved in pull request #61 at commit a5dd31c. The Aave team stated:*

> *We opted for softening the Solidity version for the sake of composability and using version 0.8.10.*

N-02 Inconsistent use of override keyword

In the vault contract, the `deposit` function signature is given with the `override` keyword, but not in the corresponding vault interface.

Consider adding the `override` keyword to the `deposit` function signature in the interface to remain consistent.

***Update**: Resolved in pull request #61 at commit ef03a62.*

N-03 Lack of indexed parameters

Within `IATokenVault.sol`, some events do not have their parameters indexed. For instance:

- line 38
- line 55

Consider indexing event parameters to avoid hindering the task of off-chain services searching and filtering for specific events.

***Update**: Resolved in pull request #61 at commit ef101cc.*

N-04 Lack of SPDX license identifier

Within `Constants.sol` there is no SPDX license identifier.

To avoid legal issues regarding copyright and follow best practices, consider adding SPDX license identifiers to files as suggested by the Solidity documentation.

***Update**: Resolved in pull request #61 at commit cb28356.*

N-05 `require` statement with multiple conditions

Within `MetaTxHelpers.sol` there is a `require` statement on line 31 that requires multiple conditions to be satisfied.

To simplify the codebase and raise the most helpful error messages for failing `require` statements, consider employing a single `require` statement per condition.

***Update:** Acknowledged, not resolved. The Aave team stated:*

> *Given that both conditions result in the same consequence (e.g., invalid signature), we consider that separating the check into two `require` statements is not worth it.*

N-06 Various improvements on doc-strings

There are several opportunities for doc-string improvements. For example, in the `ATokenVault`:

- There is an extra comma on Line 47.
- The comment on Line 70 is misleading as the initial deposit is in the underlying tokens instead of the 'underlying atokens'.
- Many functions allow third parties to use a valid signature to call, for example, `depositWithSig`, `depositATokensWithSig`, etc. These functions require the signer of the message to have already approved the appropriate amount in either the underlying token or the aToken. Consider appropriately adding this requirement in the doc-string for relevant functions.

***Update**: Resolved in pull request #61 at commit 52d187b.*

Conclusion

One medium-severity and one low-severity issue were identified, as well as various notes to improve code quality and reduce trust requirements for users. Our main recommendations are to modify some functionalities of the vault to conform to ERC-4626 standards and to handle fee-on-transfer tokens to reduce the possibility of unexpected failures when interacting with the vault contract.

Appendix

Monitoring Recommendations

While audits help in identifying potential security risks, the Aave team is encouraged to also incorporate automated monitoring of on-chain contract activity into their operations. Ongoing monitoring of deployed contracts helps in identifying potential threats and issues affecting the production environment.

It is recommended to incorporate off-chain monitoring for calls to privileged functions in the vault (i.e., `setFee`, `withdrawFees`, `claimRewards`, and `emergencyRescue`). In the rare case of keys being stolen, monitoring activity from owners can detect any unexpected activity and alert the protocol owners of the stolen keys.

The functionality of the vault depends on the state of the reserve of the underlying token in the Aave-v3 pool. If the reserve is paused, users will be unable to withdraw from the reserve and thus unable to redeem their vault shares. It is recommended to monitor the configuration of reserves that have an active corresponding vault to avoid any inadvertent state change affecting the usage of the vault.