AMPHOR AUDIT REPORT

OXORIO

APRIL 12, 2024

CONTENTS

1 INTRO

1.1 DISCLAIMER

The audit makes no assertions or warranties about the utility of the code, its security, the suitability of the business model, investment advice, endorsement of the platform or its products, the regulatory regime for the business model, or any other statements about the fitness of the contracts for their intended purposes, or their bug-free status. The audit documentation is for discussion purposes only.

The information presented in this report is confidential and privileged. If you are reading this report, you agree to maintain its confidentiality and refrain from copying, disclosing, or disseminating it without the client's consent. If you are not the intended recipient of this document, please be advised that any disclosure, copying, or dissemination of its content is strictly prohibited.

1.2 ABOUT OXORIO

Oxorio is a prominent audit and consulting firm in the blockchain industry, offering top-tier security audits and consulting to organizations worldwide. The company's expertise stems from its active involvement in designing and deploying multiple blockchain projects, wherein it developed and analyzed smart contracts.

With a team of more than six dedicated blockchain specialists, Oxorio maintains a strong commitment to excellence and client satisfaction. Its contributions to several blockchain projects reflect the company's innovation and influence in the industry. Oxorio's comprehensive approach and deep blockchain understanding make it a trusted partner for organizations in the sector.

Contact details:

- oxor.io
- ping@oxor.io
- Github
- Linkedin
- Twitter

1.3 SECURITY ASSESSMENT METHODOLOGY

Several auditors work on this audit, each independently checking the provided source code according to the security assessment methodology described below:

1. Project architecture review

The source code is manually reviewed to find errors and bugs.

2. Code check against known vulnerabilities list

The code is verified against a constantly updated list of known vulnerabilities maintained by the company.

3. Security model architecture and structure check

The project documentation is reviewed and compared with the code, including examining the comments and other technical papers.

4. Cross-check of results by different auditors

The project is typically reviewed by more than two auditors. This is followed by a mutual cross-check process of the audit results.

5. Report consolidation

The audited report is consolidated from multiple auditors.

6. Re-audit of new editions

After the client has reviewed and fixed the issues, these are double-checked. The results are included in a new version of the audit.

7. Final audit report publication

The final audit version is provided to the client and also published on the company's official website.

1.4 FINDINGS CLASSIFICATION

1.4.1 Severity Level Reference

The following severity levels were assigned to the issues described in the report:

- **CRITICAL**: A bug that could lead to asset theft, inaccessible locked funds, or any other fund loss due to unauthorized party transfers.
- **MAJOR**: A bug that could cause a contract failure, with recovery possible only through manual modification of the contract state or replacement.
- **WARNING**: A bug that could break the intended contract logic or expose it to DDoS attacks.
- **INFO**: A minor issue or recommendation reported to or acknowledged by the client's team.

1.4.2 Status Level Reference

Based on the client team's feedback regarding the list of findings discovered by the contractor, the following statuses were assigned to the findings:

- **NEW**: Awaiting feedback from the project team.
- **FIXED**: The recommended fixes have been applied to the project code, and the identified issue no longer affects the project's security.
- **ACKNOWLEDGED**: The project team is aware of this finding. Fixes for this finding are planned. This finding does not affect the overall security of the project.
- **NO ISSUE**: The finding does not affect the overall security of the project and does not violate its operational logic.

1.5 PROJECT OVERVIEW

The Amphor protocol serves as a yield aggregator for users to generate returns by supplying liquidity through ERC4626 Vaults. The yield comes from liquid staking and re-staking services, liquidity provision, and incentivized yield farming on DeFi. The generated yield from Amphor Vaults is transferred back to the users automatically. This process multiplies user's initial investment, leading to increase of value in principal and resulting profits.

1.6 AUDIT SCOPE

The scope of the audit includes the smart contracts at the following files:

- `SyncVault.sol`.
- `AsyncVault.sol`.
- `VaultZapper.sol`.

The audited commit identifier is `fc53ac5d0173757b09a49a0f902c48bc065706fb`.

2 FINDINGS REPORT

OXORIO

2.1 CRITICAL

No critical issues found.

2.2 MAJOR

M-01	Tokens with fee on transfer are not supported
Severity	MAJOR
Status	• NEW

Location

File	Location	Line
SyncVault.sol	contract `SyncVault` > function `_deposit`	497
AsyncVault.sol	contract `AsyncVault` > function `requestDeposit`	572
AsyncVault.sol	contract `AsyncVault` > function `_settle`	1006
AsyncVault.sol	contract `AsyncVault` > function `_settle`	1014
AsyncVault.sol	contract `AsyncVault` > function `_settle`	1018
AsyncVault.sol	contract `AsyncVault` > function `_settle`	1027

Description

In the mentioned locations, there is no check on how many tokens the contract actually received after the `safeTransferFrom`.

Some tokens may charge a transfer fee (e.g., such possibility exists in `USDT`) or, conversely, add some amount to the transfer. Such tokens become a problem for the protocol, as the actual amount will be different from the one specified in the `safeTransferFrom`.

For example, in the `_deposit` function, the protocol mints shares for the user based on the `assets` values passed to the `safeTransferFrom` function. If the actual number of tokens received differs from `assets`, the protocol's logic will be compromised.

Recommendation

We recommend considering the amount received for the transfer rather than relying on the amount specified in the `safeTransferFrom` call.

2.3 WARNING

W-01	`owner` parameter is restricted in `AsyncVault`
Severity	WARNING
Status	• NEW

Location

File	Location	Line
AsyncVault.sol	contract `AsyncVault` > function `initialize`	307

Description

In the `initialize` function of the `AsyncVault` contract, the `owner` parameter is utilized for initializing the contract owner. Later in this function, the `setTreasury` function is invoked, which includes a verification that `msg.sender` must be the `owner` address.

```
function setTreasury(address _treasury) public onlyOwner {
  ...
}
```

As a result, the `owner` parameter must match the `msg.sender` address.

Recommendation

We recommend implementing the capability for unrestricted customization of the owner address during contract initialization.

W-02	Redundant drawdown limit logic in `AsyncVault`
Severity	WARNING
Status	· NEW

Location

File	Location	Line
AsyncVault.sol	contract `AsyncVault` > function `_checkMaxDrawdown`	1125

Description

The function `_checkMaxDrawdown` in the `AsyncVault` contract is utilized to limit the vault's drawdown. It is activated during the settlement process and verifies that the vault's drawdown is less than `_maxDrawdown` percent.

```
function _checkMaxDrawdown(
  uint256 _lastSavedBalance,
  uint256 newSavedBalance
)
  internal
  view
{
  if (
    newSavedBalance
      < _lastSavedBalance.mulDiv(
      BPS_DIVIDER - _maxDrawdown, BPS_DIVIDER, Math.Rounding.Ceil
      )
  ) revert MaxDrawdownReached();
}
```

However, the `owner` can manipulate the `newSavedBalance` parameter by invoking the `settle` function multiple times consecutively, ensuring that each call appears to have a drawdown of less than `_maxDrawdown` percent. Cumulatively, though, the total drawdown could significantly exceed this limit. For instance, ten calls each with a `10` percent drawdown would lead to a cumulative `65` percent drawdown.

Moreover, this limit applies only to the `settle` function (which can only be used by the `owner`), and at the same time, the `owner` can disable the drawdown limit by setting `_maxDrawdown` to `100` percent using the `setMaxDrawdown` function.

Recommendation

We recommend either removing the drawdown limit logic or reimplementing it to address the concerns mentioned above.

W-03	Missing `whenNotPaused` modifier in `AsyncVault`
Severity	WARNING
Status	• NEW

Location

File	Location	Line
AsyncVault.sol	contract `AsyncVault` > function `close`	382

Description

The `close` function of the `AsyncVault` contract lacks the `whenNotPaused` modifier, which is present in most other contract functions. As a result, the owner can close the vault and withdraw all assets even when the vault is paused.

```
function close() external override onlyOwner {
  ...
  lastSavedBalance = totalAssets();
  vaultIsOpen = false;
  _asset.safeTransfer(owner(), lastSavedBalance);
  ...
}
```

Recommendation

We recommend adding the `whenNotPaused` modifier to the `close` function to ensure consistency in how the contract handles paused state.

W-04	Loss of shares in certain conditions in `AsyncVault`
Severity	WARNING
Status	NEW

Description

In the protocol, a complex sequence of events may block some of the shares that have been minted for `claimableSilo` for distribution to users who have requested a deposit. This can be observed by examining the following scenario:

- owner initializes the contract with `bootstrapAmount` equal to `10000`
- owner calls the `close` function, which allows users to execute `requestDeposit`
- Alice executes `requestDeposit` with `assets` equal to `1010`
- Bob executes `requestDeposit` with `assets` equal to `1000`
- owner calls the `open` function with the value `assetReturned` equal to `10001`
- `2009` shares minted for Alice and Bob's deposits (transferred to `claimableSilo`)
- Alice calls `claimDeposit` and receives `1009` shares
- Bob receives only `999`, and one share remains locked on the `claimableSilo` contract, since `1009 + 999 = 2008`

This results in locking a portion of the user's funds, particularly in scenarios involving numerous users and substantial monetary operations.

Recommendation

We recommend revising the logic for distributing shares for deposits so that users receive a fair number of shares if the above is relevant to the protocol.

W-05	Insufficient verification of `bootstrapAmount`, which can lead to an inflationary attack in `SyncVault`
Severity	WARNING
Status	• NEW

Location

File	Location	Line
SyncVault.sol	contract `SyncVault` > function `initialize`	178

Description

In the function `initialize` of the contract `SyncVault`, there is no check to ensure that `bootstrapAmount` is not `0`, which could lead to a potential inflationary attack on protocol users.

Recommendation

We recommend adding a check to ensure that `bootstrapAmount` is greater than `0` to prevent an inflationary attack.

W-06	Identical event parameters for `FeesChanged` in `SyncVault`
Severity	WARNING
Status	• NEW

Location

File	Location	Line
SyncVault.sol	contract `SyncVault` > function `setFee`	241

Description

In the function `setFee` of the contract `SyncVault`, the event `FeesChanged` emits two arguments that are both set to `newFee`. This redundancy occurs because `feesInBps` is assigned the value of `newFee` immediately before the event is emitted, meaning `feesInBps` and `newFee` are guaranteed to be identical at the time of emission. This redundancy could lead to unnecessary consumption of gas and does not provide additional information to event subscribers.

```
feesInBps = newFee;
emit FeesChanged(feesInBps, newFee);
```

Recommendation

We recommend refactoring the `setFee` function to emit `newFee` and `feesInBps` before the assignment `feesInBps = newFee;`.

W-07	No event emission when `_maxDrawdown` is changed in `SyncVault`
Severity	WARNING
Status	• NEW

Location

File	Location	Line
SyncVault.sol	contract `SyncVault` > function `setMaxDrawdown`	244-247

Description

In the function `setMaxDrawdown` of the contract `SyncVault`, no event is emitted when the `_maxDrawdown` parameter is updated. This parameter is critical as it defines the maximum permissible drawdown, affecting the risk management procedures of the vault. The absence of event emission means that external subscribers, such as user interfaces, are not notified of this change, potentially leading to the use of outdated or incorrect information.

Recommendation

We recommend introducing an event emission upon the update of `_maxDrawdown`.

2.4 INFO

No info issues found.

3 CONCLUSION

The following table contains all the findings identified during the audit, grouped by statuses and severity levels:

Severity	NEW
CRITICAL	0
MAJOR	1
WARNING	7
INFO	0
Total	8

THANK YOU FOR CHOOSING