Xmplified × PESSIMISTIC

SECURITY ANALYSIS

by Pessimistic

This report is public

October 23, 2024

ABSTRACT

In this report, we consider the security of smart contracts of Amplified Strategy Vault project. Our task is to find and describe security issues in the smart contracts of the platform.

DISCLAIMER

The audit does not give any warranties on the security of the code. A single audit cannot be considered enough. We always recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts. Besides, a security audit is not investment advice.

SUMMARY

In this report, we considered the security of Amplified Strategy Vault smart contracts. We described the audit process in the section below.

The audit showed two critical issues: Exchange rate not accounted, Inaccurate shares calculation. The audit also revealed several issues of medium severity: Burning unspent shares, Possible frontrun, Redeem may return zero, Vault earnings not accounted, Incorrect upgrade of vaults. Moreover, several low-severity issues were found.

After the initial audit, the developers provided a new version of the code. All critical and medium issues were either resolved or addressed. However, four new medium severity issues were identified during the recheck.

After that, the developers provided us with a new version of the code. In this update, they fixed several medium issues. M02 issue was partially fixed. Several new issues were added with this update as well: Incorrect re-weighting after emergency exit medium issue and Use safeTransfer, Discrepancies in conversions, and Deposit ignoring maxDeposit low-severity issues.

Later the developer fixed most of the discovered issues. We verified that all the fixes are effective and do not produce any additional problems.

GENERAL RECOMMENDATIONS

We recommend implementing CI to run tests, calculate code coverage, and analyze code with linters and security tools.

PROJECT OVERVIEW

Project description

For the audit, we were provided with Amplified Strategy Vault project on a private GitHub repository, commit 81771d108877acb5454810269693788866 1ba561.

The scope of the audit included

- **src/strategy/StrategyVault.sol**;
- **src/strategy/StrategyVaultFactory.sol**;
- **src/strategy/base/StrategyManager.sol**.

The documentation for the project included this link and `README.md`.

All 84 tests pass successfully. The code coverage is 96%.

The total LOC of audited sources is 592.

Codebase update #1

After the initial audit, the developers provided an updated version of the code: commit 3f5d204c1158fb53eaedcc98aecaa2664a2c5838. In that update, most of the issues were fixed. Four new medium issues were found: Inflation attack, Possible DOS, Ceiling during mint and withdraw calls, Incorrect total assets value.

All 85 tests pass successfully. The code coverage is 96%.

Codebase update #2

After the previous codebase update, the developers provided us with a new version of the code, commit e58f4e7fa4a6cb1cc93eefc53748ac093122fc90. The update included fixes for M02, M07, M06, M08, and M09 issues. The developers added a function to remove broken strategies from the vault. One new medium and three new low-severity issues were found during the recheck: M10, L06, L07, and L08. M02 issue was partially fixed.

All 104 tests pass successfully. The code coverage increased to 98%.

AUDIT PROCESS

We started the audit on September 23, 2024 and finished on September 26, 2024.

We inspected the materials provided for the audit. Then, we contacted the developers for an introduction to the project. After a discussion, we performed preliminary research and specified those parts of the code and logic that require additional attention during the audit:

- Whether the conversion between shares and assets is accurate;
- Whether there is no possibility to profit from frontrunning users;
- Whether the vault correctly implments **EIP-4626** standard.

We manually analyzed all the contracts within the scope of the audit and checked their logic. Additionally, we scanned the project with the following tools:

- Static analyzer Slither;
- Our plugin Slitherin with an extended set of rules;
- Semgrep rules for smart contracts. We also sent the results to the developers in the text file.

We ran tests and calculated the code coverage.

We combined in a private report all the verified issues we found during the manual audit or discovered by automated tools.

After the initial audit, we discussed the results with the developers. On October 3, 2024 the developers provided us with an updated version of the code.

We reviewed the updated codebase and scanned it with the following tools:

- Static analyzer Slither;
- Our plugin Slitherin with an extended set of rules.

The developers fixed most of the issues mentioned in the initial report. Four new medium issues were added after the recheck and one note was removed. In this version of the code, all tests pass. The code coverage is 96%. The number of tests increased.

After that, we updated the report.

Following the report, the developers made updates to the codebase. As a result, they fixed four medium severity issues, and one issue was fixed partially. One new medium and three low-severity issues were introduced with this updated.

In addition to this, we ran the static analyzer Slither with our plugin Slitherin and verified their outputs.

After reviewing the fixes, we updated the report.

After the second codebase update, the developers introduced minor fixes and resolved most of the remaining issues.

Finally, we updated the report.

MANUAL ANALYSIS

The contracts were completely manually analyzed, their logic was checked. Besides, the results of the automated analysis were manually verified. All the confirmed issues are described below.

Critical issues

Critical issues seriously endanger project security. They can lead to loss of funds or other catastrophic consequences. The contracts should not be deployed before these issues are fixed.

C01. Exchange rate not accounted (fixed)

The vault does not account for an exchange rate between shares and assets. This allows a new user to deposit assets into the vault and withdraw them, receiving a portion of the rewards that were earned prior to their deposit. Consider revising the share minting logic in the `_deposit` function of the **StrategyVault** contract to ensure fair reward distribution.

> The issue has been fixed and is not present in the latest version of the code.

C02. Inaccurate shares calculation (fixed)

The shares of strategies may have different decimals compared to the shares of the vault. However, they are currently summed together in the `previewWithdraw` function, which could lead to incorrect behavior. Additionally, in the `_preview_redeem` function, vault shares are used for the `previewRedeem` call to strategies, which might be inaccurate as strategies may not mint shares with the same exchange rate as the vault. Consider the accounting exchange rate between shares of the vault and shares of the strategies in the `previewWithdraw` and `previewRedeem` functions of the **StrategyVault** contract.

> The issues have been fixed and are not present in the latest version of the code.

Medium severity issues

Medium severity issues can influence project operation in the current implementation. Bugs, loss of potential income, and other non-critical failures fall into this category, as well as potential problems related to incorrect system management. We highly recommend addressing them.

M01. Burning unspent shares (fixed)

The code does not revert if the amount of shares to be redeemed exceeds `strategy.maxRedeem` value. It might be better for a user to revert in such cases and call several separate `redeem` transactions to redeem all of the shares. Moreover, the vault will burn unspent vault shares provided by the user, so part of the assets in strategies will not be redeemable. The issue can be found in line 249 in the `_redeem_strategy` function of the **StrategyManager** contract.

> The issue has been fixed and is not present in the latest version of the code.

M02. Possible frontrun (fixed)

During the `_rebalanceStrategy`, `redeem`, `mint`, `deposit`, or `withdraw` of tokens, potential swaps may occur within the strategy. Since there is no option for the caller to specify a minimum asset amount, they may be vulnerable to frontrunning. Consider adding some slippage protection in these functions of the **StrategyVault** and **StrategyManager** contracts.

> With the second codebase update developers added slippage protected versions for `redeem`, `mint`, `deposit`, and `withdraw` functions. However, `_rebalanceStrategy` still does not implement checks correctly.

> The issue has been fixed with commit d1190ed34a394b6a407b509169ac1ad1068a271e and is not present in the latest version of the code.

M03. Redeem may return zero (fixed)

It should be an expected behaviour if call to `redeem` of a strategy returns zero. It can happen due to strategy losses and should be considered a valid behaviour as long as `previewRedeem` returned the same result. Consider not reverting in such cases or enforcing a flag that will revert on zero redeems at line 253 in `_redeem_strategy` function of the **StrategyManager** contract.

> The issue has been fixed and is not present in the latest version of the code.

M04. Vault earnings not accounted (fixed)

The conversion functions currently return values based on a 1:1 conversion between shares and assets. However, they do not account for the vault's earnings, which should be used during the conversion to ensure an accurate exchange rate in the `convertToAssets` and `convertToShares` functions of the **StrategyVault** contract.

> The issues have been fixed and are not present in the latest version of the code.

M05. Incorrect upgrade of vaults (fixed)

In the `upgradeStrategyVaultsAndCall` function of the **StrategyVaultFactory** contract, the `vaults` parameter is not utilized in the loop for the upgrade. Instead, the first `vault.length` vaults from the `strategyVaults` array are upgraded. Consider replacing `strategyVaults[i]` with `vaults[i]` at line 166 of the **StrategyVaultFactory** contract.

> The issue has been fixed and is not present in the latest version of the code.

M06. Inflation attack (fixed)

The current implementation of the vault is vulnerable to an inflation attack. To mitigate this, consider adding virtual shares to the vault and ensuring that minted shares are not zero. Otherwise, such a scenario is possible:

- For simplicity, assume the vault has only one strategy;
- A hacker deposits `1` asset and receives `1` share in return;
- When a large depositor attempts to deposit into the vault, the hacker can directly transfer assets to the underlying strategy, inflating the value of `totalAssets` in the **StrategyVault**;
- As a result, the depositor receives zero shares, and the hacker can redeem their `1` share for the entire vault's assets.

Zero mint check is required to prevent this.

- Moreover, without proper amount of virtual shares it is still possible to steal money of the depositors.

We recommend adding `1,000` virtual shares to significantly reduce the risk of this attack.

> The issue has been fixed and is not present in the latest version of the code.

M07. Possible DOS (fixed)

Strategies may start reverting on any call made to them, preventing users from opting out of the vault if one of the strategies always reverts. Consider implementing a function that allows interactions with strategies while ignoring reverts. Additionally, it is important to be able to rebalance strategies in such scenarios. We recommend addressing this case as well.

> The issue has been fixed and is not present in the latest version of the code.

> Developers introduced `emergencyRemoveStrategy` function to remove vulnerable strategies with the second codebase update. However, incorrect rebalancing was implemeted to resolve the issue. We described it in detail in M10 issue.

M08. Ceiling during mint and withdraw calls (fixed)

The `previewMint` and `previewWithdraw` functions of the **StrategyVault** should round up the required amount of assets to ensure accurate conversion between shares and assets. Without this, there is a potential to get more assets for the same amount of shares or mint the same amount of shares for less assets.

> The issues have been fixed and are not present in the latest version of the code.

M09. Incorrect total assets value (fixed)

In the current implementation, the vault calculates its `totalAssets` by summing the `totalAssets` of all the underlying strategies. However, since other depositors may also contribute to these strategies, the vault's `totalAssets` should instead be based on the value of its shares in these strategies.

> The issue has been fixed and is not present in the latest version of the code.

M10. Incorrect re-weighting after emergency exit (fixed)

It is possible to remove reverting strategies using `emergencyRemoveStrategy` function of the **StrategyManager** contract. However, the reweighting mechanism does not reweight strategies correctly at lines 182-186. Consider calculating new weights as: `weight + removeWeight * weight / (TOTAL_WEIGHT - removeWeight)`.

> The issue has been fixed with commit d1190ed34a394b6a407b509169ac1ad1068a271e and is not present in the latest version of the code.

Low severity issues

Low severity issues do not directly affect project operation. However, they might lead to various problems in future versions of the code. We recommend fixing them or explaining why the team has chosen a particular option.

L01. Better totalAssets accountment (fixed)

The function `totalAssets` of the **StrategyVault** contract should account for any compounding returns generated from yield. According to the ERC4626 standard, `totalAssets` should reflect the true value of the vault's holdings, including any accrued yield or compounding. Currently, the function does not capture these earnings, which could result in an inaccurate representation of the vault's total assets. Consider updating the logic to comply with the **ERC4626** standard and include yield compounding in the calculation.

> The issue has been fixed and is not present in the latest version of the code.

L02. Dependency management

Consider avoiding the direct copying of external sources into the project folder. We recommend managing dependencies externally using **npm** or **yarn** package managers instead.

L03. Gas optimization (fixed)

WETH can be used directly instead of retrieving `_asset()` from storage, as the function already verifies that `_asset() == WETH` at line 57 in `receive` function of the **StrategyVault** contract.

> The issue has been fixed and is not present in the latest version of the code.

L04. Storage read in loops (fixed)

The `strategyVaultImplementation` variable is read repeatedly in several loops within the **StrategyVaultFactory** contract. To reduce gas consumption, consider reading it once into a local variable before the loops.

> The issues have been fixed and are not present in the latest version of the code.

L05. Hash does not match documented preimage (fixed)

The value of `STRATEGY_MANAGER_STORAGE_LOCATION` at line 41 in the **StrategyManager** contract differs from the one that can be obtained from the comment above.

> The issue has been fixed and is not present in the latest version of the code.

L06. Use safeTransfer (fixed)

Consider using `safeTransfer` function to safely transfer tokens at line 167 of the **StrategyManager** contract as some tokens might not return `bool` value as a result of the transfer.

> The issue has been fixed with commit d1190ed34a394b6a407b509169ac1ad1068a271e and is not present in the latest version of the code.

L07. Discrepancies in conversions (fixed)

`_convertToShares` function of **StrategyManager** contract adds `1` to `_totalAssets` during conversion. At the same time, `_convertToAssets` does not contain such an addition.

> The issue has been fixed with commit d1190ed34a394b6a407b509169ac1ad1068a271e and is not present in the latest version of the code.

L08. Deposit ignoring maxDeposit (commented)

The vault with underlying **WETH** token allows to make deposits using the native token through `receive` method. However, `receive` does not validate that the deposited amount is less than `maxDeposit` result, which allows to bypass the same check in the `deposit` function using native deposit.

> `Comment from the developers:`
> By contract design `maxDeposit` pure function always returns `type(uint256).max` value. `receive` and `deposit` functions' input value argument has `uint256` type (32-bytes). So we don't need check the deposit asset amount exeeds `maxDeposit`. This check is already covered by Solidity Compiler and EVM.

Notes

N01. Project roles

In the current implementation, the system relies heavily on the owner role, which has the following powers:

- The `owner` of the **StrategyVaultFactory** can upgrade, change the vault implementations, and make vaults immutable;
- The `owner` of the **StrategyVault** can pause/unpause the contract and rebalance assets within strategies.

> The developers have informed us that the owner of these contracts will be a multi-signature wallet.

This analysis was performed by Pessimistic:

Yhtyyar Sahatov, Security Engineer
Oleg Bobrov, Security Engineer
Konstantin Zherebtsov, Business Development Lead
Irina Vikhareva, Project Manager
Alexander Seleznev, CEO

October 23, 2024