gkrastenov

Smart Contract Security Review

Yield.Meme

April 27 2024

gkrastenov.com gkrastenov gkrastenov

Table of Contents

1 About gkrastenov **2**

2 Disclaimer **2**

3 About Yield.Meme **2**

4 Risk classification **2**
4.1 Impact . . . 2
4.2 Likelihood . . . 2
4.3 Actions required by severity level . . . 3

5 Executive summary **4**

6 Findings **5**
6.1 High risk . . . 5
6.1.1 User can claim more tokens than expected . . . 5
6.2 Medium risk . . . 5
6.2.1 At some point the getAmountsOut function will always revert . . . 5
6.2.2 Lack of access control in swap function . . . 6
6.2.3 Possible future DDoS because of many reward tokens . . . 6
6.3 Low risk . . . 7
6.3.1 After adding the new token, the last harvest time is not set . . . 7
6.3.2 The protocol does not support reward tokens with more than 18 decimals . . . 7
6.4 Informational . . . 7
6.4.1 The getUsdValue function is redundant . . . 7

1 About gkrastenov

Georgi Krastenov, known as gkrastenov, is an independent smart contract security researcher and former smart contract engineer at Nexo. Having conducted over 15 solo smart contract security reviews and discovered numerous vulnerabilities in various protocols, he does his best to contribute to the blockchain ecosystem and its protocols by dedicating time and effort to security research and reviews. Check his previous work here or reach out on Twitter/X or Telegram @gkrastenov.

2 Disclaimer

Audits are a time, resource, and expertise bound effort where trained experts evaluate smart contracts using a combination of automated and manual techniques to identify as many vulnerabilities as possible. Audits can show the presence of vulnerabilities **but not their absence**.

3 About Yield.Meme

The Yield.Meme protocol allows users to deposit stablecoins into Vaults where they can earn yields in popular memecoins. Users' staked assets are deployed in various strategies, generating yield. This yield is then bulk-converted into memecoins, efficiently saving on gas fees and facilitating a Dollar Cost Averaging (DCA) strategy for acquiring memecoins.

4 Risk classification

Severity	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

4.1 Impact

- **High** - leads to a significant loss of assets in the protocol or significantly harms a group of users.
- **Medium** - only a small amount of funds can be lost or a functionality of the protocol is affected.
- **Low** - any kind of unexpected behaviour that's not so critical.

4.2 Likelihood

- **High** - direct attack vector; the cost is relatively low to the amount of funds that can be lost.
- **Medium** - only conditionally incentivized attack vector, but still relatively likely.
- **Low** - too many or too unlikely assumptions; provides little or no incentive.

4.3 Actions required by severity level

- **Critical** - client **must** fix the issue.
- **High** - client **must** fix the issue.
- **Medium** - client **should** fix the issue.
- **Low** - client **could** fix the issue.

5 Executive summary

Overview

Project Name	Yield.Meme
Repository	https://github.com/cuddlechamp/contracts/tree/main
Commit hash	f874587a87781e8219a07ab98a76059a5ab488f7
Review Commit hash	a7c96734bdb85eba488220156f0dd766bcc3c8ce
Documentation	https://docs.yield.meme/
Methods	Manual review

Scope

contracts/AaveERC4626Vault.sol
contracts/BaseVault.sol
contracts/Swapper.sol

Timeline

April 15, 2024	Audit kick-off
April 21, 2024	Preliminary report
April 27, 2024	Mitigation review

Issues Found

Severity	**Count**	**Fixed**	**Acknowledged**
Critical Risk	0	0	0
High Risk	1	1	0
Medium Risk	3	1	2
Low Risk	2	1	1
Informational	1	1	0
Total	**7**	**4**	**3**

6 Findings

6.1 High risk

6.1.1 User can claim more tokens than expected

Severity: *High risk*

Context: BaseVault.sol#L238

Description: Users can claim more tokens by changing the order of the reward tokens. The protocol expects that users will claim their tokens in the exact order as in the `rewardTokens` array, but this is not always true and some users can benefit from that.

Example: Suppose Alice holds 10% of the total shares and the reward tokens are as follows:

Token A, price = $0.02, and the planned distribution is 100K tokens. Token B, price = $0.04, and the planned distribution is 100K tokens. Token C, price = $0.01, and the planned distribution is 100K tokens.

For the three reward tokens, `A`, `B`, and `C`, Alice expects to claim a total of `$700` `(10K tokenA * $0.02 + 10K tokenB * $0.04 + 10K tokenC * $0.01)`. However, she can gain more reward if she changes the order of the tokens and executes a few transactions:

The Attack Vector:

1. In transaction 1, she calls the `claim` function with tokens `A` and `C`. The reward is equal to `$600` `(10K tokenA * $0.02 + 10K tokenB * $0.04)`.
2. In transaction 2, she calls the `claim` function with tokens `A` and `B`. The reward is equal to `$400` `(0 tokenA * $0.02 + 10K tokenB * $0.04)`.
3. As a result, Alice claims `20K tokenB` and `10K tokenA` for a total sum of `$1000`, which is more than expected.

Recommendation: Make the following changes:

```
-IERC20(rewardTokens[i]).transfer(owner, claimable_);
+IERC20(tokens[i]).transfer(owner, claimable_);
```

Resolution and Client comment: Resolved. Fixed at `e2c4384a08c5d3b56041444a34923061ff7e1ecf` commit.

6.2 Medium risk

6.2.1 At some point the getAmountsOut function will always revert

Severity: *Medium risk*

Context: Swapper.sol#L178

Description: When the `getAmountsOut` function is called, it returns an array with the expected amounts after swapping the tokens. In the case where `vault.tokenPerMille(i)= 0`, the `distributedBaseTokens` will be equal to `0`, which will cause the transaction to revert. The problem arises because in the `quoteExactInput` function in the `UniswapV3StaticQuoter` contract, if the input argument is equal to `0`, the transaction will revert.

```
        } else if (config.venue == SwapVenue.UNISWAPV3) {
            IUniswapV3StaticQuoter quoter = IUniswapV3StaticQuoter(
                quoters[SwapVenue.UNISWAPV3]
            );
            //@audit-issue revert if amount = 0
            return quoter.quoteExactInput(config.path, amount);
```

```
    /// @inheritdoc IUniswapV3PoolActions
    function swap(
        address recipient,
        bool zeroForOne,
        int256 amountSpecified, // 0
        uint160 sqrtPriceLimitX96,
        bytes calldata data
    ) external override noDelegateCall returns (int256 amount0, int256 amount1) {
        require(amountSpecified != 0, 'AS'); //@audit-issue revert
```

Recommendation: When `distributedBaseTokens` equals `0`, directly set `amounts[i] = 0`.

Resolution and Client comment: Resolved. Fixed at `b1e2fac659d7077596192b06f38d136c71cd7f2c` commit.

6.2.2 Lack of access control in swap function

Severity: *Medium risk*

Context: Swapper.sol#L110

Description: The `swap` function in the `Swapper` contract lacks access control. A malicious user (a smart contract that inherits and implements the `Vault` interface) can call this function and transfer all funds in the contract to themselves. The `Swapper` contract is expected to not handle any funds, but if reward tokens or asset tokens are sent to it, they can easily be transferred by anyone.

Recommendation: Allow only legitimate vaults (mUSDT, mUSDC, and mDAI) to call the swap function.

Resolution and Client comment: Acknowledge. Currently, a central registry is not created for the vaults.

6.2.3 Possible future DDoS because of many reward tokens

Severity: *Medium risk*

Context: BaseVault.sol#L116

Description: Currently, the reward tokens can only be added and not removed. There are many places in the codebase with for loops iterating over all tokens, which leads to significant gas usage. While this might not be a problem for some L2 networks but for other networks where the maximum block gas limit is specified and more tokens are added in the future, this could lead to a DDoS of the protocol.

Recommendation: Remove the reward tokens when they are not supported anymore.

Resolution and Client comment: Acknowledge.

6.3 Low risk

6.3.1 After adding the new token, the last harvest time is not set

Severity: *Low risk*

Context: BaseVault.sol#L286-L305

Description: When a new reward token is added, the setting of the last harvest time `rewards[token].lastHarvest = uint32(block.timestamp)` is missing. This is not problematic and will not lead to significant issues, but the initial planned distribution of the new token would be exactly 3 days.

Recommendation: After adding a new token, update the last harvest time.

Resolution and Client comment: Resolved. Fixed at `a7c96734bdb85eba488220156f0dd766bcc3c8ce` commit.

6.3.2 The protocol does not support reward tokens with more than 18 decimals

Severity: *Low risk*

Context: BaseVault.sol#L299-L303

Description: Currently, the protocol does not support reward tokens with decimals greater than 18. If a token with decimal above 18 is added in the future, it will lead to incorrect calculation of the claimed rewards.

```
uint256 decimals = IERC20Metadata(token).decimals();

if (decimals < 18) {
    tokenPrecisions[token] = 10 ** (18 - decimals);
} else {
    tokenPrecisions[token] = 1;
}
```

Some of the meme tokens have a very large supply and sometimes the developer changes the tokens' decimals. Therefore, it is not unexpected for a future popular meme token to have more than 18 decimals.

Recommendation: Avoid adding reward tokens with decimals greater than 18 or ensure that their token precision is not equal to 1.

Resolution and Client comment: Acknowledge. Token with decimal above 18 does not need more precision, the precision multiplier is then set to zero which should not affect tokens with more than 18 decimals.

6.4 Informational

6.4.1 The getUsdValue function is redundant

Severity: *Informational*

Context: Swapper.sol#L242-L245

Description: The `getUsdValue` function in the `Swapper` contract is redundant. The function is never used and has an empty implementation.

Recommendation: Remove or implement the function with the expected logic.

Resolution and Client comment: Resolved. Fixed at `c69a45689bc1f52951677fcd50401e3d3ffc24b1` commit.