Kilo - Vault2
Audit Report

Tue Apr 30 2024

contact@scalebit.xyz https://twitter.com/scalebit_

ScaleBit

Kilo - Vault2 Audit Report

1 Executive Summary

1.1 Project Information

Description	A user-friendly perpetual DEX
Type	DeFi
Auditors	ScaleBit
Timeline	Wed Apr 17 2024 - Tue Apr 30 2024
Languages	Solidity
Platform	Manta
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/KiloExPerp/kilo-vault2.git
Commits	d5cfa33078a7872ece6841ee04e7eba9db9e9a21 2d4875d8b758c72fe20948d43cedc34ce6f45768 62bebd37e40f5194603021b013a2b3ac1a3ea091

1.2 Files in Scope

The following are the SHA1 hashes of the original reviewed files.

ID	File	SHA-1 Hash
OOGU	contracts/access/OperatorOwnerGovernableUpgradeable.sol	f0edeac24f9b0dbbe54c3d98f2b0c8f00fc45da0
OGU	contracts/access/OwnerGovernableUpgradeable.sol	3d4e40d9e82b4b1d5fef28527577b95b85841e43
VSR	contracts/core/VaultStakeReward.sol	3feea012f605eee01bd424e391038ac28f4e5e5b
KTLDND	contracts/vaultv2/KTokenLockedDepositNftDesign.sol	b16398356c1a587eab62ad5252ac4034514559d9
KTLDN	contracts/vaultv2/KTokenLockedDepositNft.sol	c3e930f119749e68216f7520e9059a0ca2816a55
KTOPF	contracts/vaultv2/KTokenOpenPnlFeed.sol	8eb722f5f07b88a5021ad581b8514408e609f55b

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	11	10	1
Informational	3	2	1
Minor	5	5	0
Medium	1	1	0
Major	2	2	0
Critical	0	0	0

1.4 ScaleBit Audit Breakdown

ScaleBit aims to assess repositories for security-related issues, code quality, and compliance with specifications and best practices. Possible issues our team looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Integer overflow/underflow
- Number of rounding errors
- Unchecked External Call
- Unchecked CALL Return Values
- Functionality Checks
- Reentrancy
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic issues
- Gas usage
- Fallback function usage
- tx.origin authentication
- Replay attacks
- Coding style issues

1.5 Methodology

The security team adopted the **"Testing and Automated Analysis"**, **"Code Review"** and **"Formal Verification"** strategy to perform a complete security test on the code in a way that is closest to the real attack. The main entrance and scope of security testing are stated in the conventions in the "Audit Objective", which can expand to contexts beyond the scope according to the actual testing needs. The main types of this security audit include:

(1) Testing and Automated Analysis

Items to check: state consistency / failure rollback / unit testing / value overflows / parameter verification / unhandled errors / boundary checking / coding specifications.

(2) Code Review

The code scope is illustrated in section 1.2.

(3) Audit Process

- Carry out relevant security tests on the testnet or the mainnet;
- If there are any questions during the audit process, communicate with the code owner in time. The code owners should actively cooperate (this might include providing the latest stable source code, relevant deployment scripts or methods, transaction signature scripts, exchange docking schemes, etc.);
- The necessary information during the audit process will be well documented for both the audit team and the code owner in a timely manner.

2 Summary

This report has been commissioned by Kilo to identify any potential issues and vulnerabilities in the source code of the Kilo - Vault2 smart contract, as well as any contract dependencies that were not part of an officially recognized library. In this audit, we have utilized various techniques, including manual code review and static analysis, to identify potential vulnerabilities and security issues.

During the audit, we identified 11 issues of varying severity, listed below.

ID	Title	Severity	Status
KTL-1	Replace `_safeMint()` with `_mint()`	Medium	Fixed
KTO-1	`forceNewEpoch()` can Disrupt the Execution Plan of the Operator when Executed during `makeOpenPnlRequest()`	Major	Fixed
KTO-2	`makeOpenPnlRequest()` Lacks Access Control	Major	Fixed
VSR-1	Missing check `depositId`	Minor	Fixed
VSR-2	Lack of Parameter Validation in `initializeV2` Function	Minor	Fixed
VSR-3	Redundant Constants in `VaultStakeReward` Contract	Minor	Fixed
VSR-4	The Implementation of the Withdraw Epochs Timelock is incorrect	Minor	Fixed
VSR-5	Redundant Inheritance	Informational	Acknowledged
VSR-6	Unused Function Parameter	Informational	Fixed

VSR-7	Inconsistent Comment Style and Non-English Comments	Informational	Fixed
KTL1-1	Incomplete Functionality in `numberToRoundedString()`	Minor	Fixed

3 Participant Process

Here are the relevant actors with their respective abilities within the Kilo - Vault2 Smart Contract :

Gov

- The Gov can invoke the `setGov` function to set a new `Gov` .
- The Gov can invoke the `setOwner` function to set a new `owner` .

Owner

- The owner can call the `setOperator` function to set a new `operator` .
- The owner can call the `updateRequestsStart` , `updateRequestsEvery` , and `updateRequestsCount` functions to respectively update `requestsStart` , `requestsEvery` , and `requestsCount` .
- The owner can call the `updatePnlHandler` function to update the `PnlHandler` address.
- The owner can use the `updateOpenTradesPnlFeed` function to update the address of the `OpenTradesPnlFeed` contract.
- The owner can invoke the `updateMaxAccOpenPnlDelta` function to adjust the `maxAccOpenPnlDelta` parameter.
- The owner can utilize the `updateMaxDailyAccPnlDelta` function to adjust the `maxDailyAccPnlDelta` parameter.
- The owner can invoke the `updateWithdrawLockThresholdsP` function to update the `withdrawLockThresholdsP` .
- The owner can invoke the `updateMaxSupplyIncreaseDailyP` function to adjust the `maxSupplyIncreaseDailyP` .
- The owner can use the `updateLossesBurnP` function to update the `lossesBurnP` .
- The owner can call the `updateMaxDiscountP` function to adjust the `maxDiscountP` .
- The owner can utilize the `updateMaxDiscountThresholdP` function to update the `maxDiscountThresholdP` .

TokenManager

- KTokenManager can call the `updateDesign` function to update the `design` .

- KTokenManager can call the `updateDesignDecimals` function to update the `designDecimals`.

KToken

- KToken can call the `mint` and `burn` functions to mint and burn NFTs.

User

- Users can call the `forceNewEpoch` function to force update a new epoch.
- Users can call the `makeOpenPnlRequest` function to collect `openPnlValue` over multiple calls, initiating a new epoch after the fifth call.
- Users can call the `deposit` function to deposit assets into the contract, converting them to shares and scaling variables accordingly.
- Users can invoke the `mint` function to mint shares for a specified receiver, converting them from assets and scaling variables accordingly.
- Users can use the `withdraw` function to withdraw assets from the contract, converting shares to assets and scaling variables accordingly.
- Users can call the `redeem` function to redeem shares for assets, scaling variables accordingly.
- Users can call `makeWithdrawRequest` to request a withdrawal of shares for a specified owner.
- Users can invoke `cancelWithdrawRequest` to cancel a previous withdrawal request for a specified amount of shares.
- Users can utilize `depositWithDiscountAndLock` to deposit assets with a discount and lock them for a specified duration.
- Users can call `mintWithDiscountAndLock` to mint shares with a discount and lock them for a specified duration.
- Users can invoke `unlockDeposit` to unlock a previously locked deposit, transferring the locked shares to a specified receiver after the lock duration has passed.
- Users can call `distributeReward` to evenly distribute a reward to all stakers of the vault, updating accumulated rewards per token and total rewards accordingly.
- Users can call `receiveAssets` to receive assets, updating accumulated PnL per token, daily accumulated PnL delta, total liability, and total closed PnL accordingly. Additionally, it depletes a portion of assets if there are accumulated losses.

PnlHandler

- The pnlHandler can call `sendAssets` to send assets, updating accumulated PnL per token, daily accumulated PnL delta, total liability, and total closed PnL accordingly.

OpenTradesPnlFeed

- OpenTradesPnlFeed can call the `updateAccPnlPerTokenUsed` function to update the accumulated PnL per token used based on new positive open PnL values.

4 Findings

KTL-1 Replace `_safeMint()` with `_mint()`

Severity: Medium

Status: Fixed

Code Location:

contracts/vaultv2/KTokenLockedDepositNft.sol#52

Descriptions:

In the `KTokenLockedDepositNft.mint()` function, the protocol utilizes `_safeMint()` to mint NFTs for users.

```
function mint(address to, uint tokenId) external onlyKToken {
    _safeMint(to, tokenId);
}
```

However, as indicated by the following code, `_safeMint()` carries a reentrancy risk. Within `_safeMint()` , the protocol calls `IERC721Receiver(to).onERC721Received()` , which presents a risk of reentrancy.

```
function _safeMint(address to, uint256 tokenId, bytes memory data) internal virtual {
    _mint(to, tokenId);
    ERC721Utils.checkOnERC721Received(_msgSender(), address(0), to, tokenId, data);
}
```

```

function checkOnERC721Received(
        address operator,
        address from,
        address to,
        uint256 tokenId,
        bytes memory data
    ) internal {
        if (to.code.length > 0) {
            try IERC721Receiver(to).onERC721Received(operator, from, tokenId, data) returns
(bytes4 retval) {
                if (retval != IERC721Receiver.onERC721Received.selector) {
```

```
            // Token rejected
            revert IERC721Errors.ERC721InvalidReceiver(to);
        }
```

Suggestion:

It is recommended to use `_mint()` instead.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

KTO-1 `forceNewEpoch()` can Disrupt the Execution Plan of the Operator when Executed during `makeOpenPnlRequest()`

Severity: Major

Status: Fixed

Code Location:

contracts/vaultv2/KTokenOpenPnlFeed.sol#112-117

Descriptions:

The `KTokenOpenPnlFeed.makeOpenPnlRequest()` function updates the open PNL value, and each invocation updates `nextEpochValuesLastRequest` to `block.timestamp`. The next call must satisfy the condition `block.timestamp - nextEpochValuesLastRequest >= requestsEvery`, and invoking it 5 times will start a new epoch.

```
 function makeOpenPnlRequest(
      uint requestId,
      int openPnlValue // collateral.precision (assets)
   ) external onlyOperator {
      bool firstRequest = nextEpochValuesLastRequest == 0;

      if(firstRequest && block.timestamp - kToken.currentEpochStart() >= requestsStart) {
//just mark we are starting to makeOpenPnlRequest
         nextEpochValuesRequestCount = 1;
         nextEpochValuesLastRequest = block.timestamp;
      } else if(!firstRequest && block.timestamp - nextEpochValuesLastRequest >=
requestsEvery) { //collect openPnlValue four times
         if (nextEpochValuesRequestCount <= requestsCount) {
            effectiveOpenPnlRequest(requestId, openPnlValue);
         }
         if(nextEpochValues.length >= requestsCount) {
            startNewEpoch();
         }
      } else {
         revert("TOO_EARLY");
      }
   }
```

Anyone can call `forceNewEpoch()` when `block.timestamp - kToken.currentEpochStart() >= requestsStart + requestsEvery * requestsCount`.

```
function forceNewEpoch() external {
    require(block.timestamp - kToken.currentEpochStart()
       >= requestsStart + requestsEvery * requestsCount,"TOO_EARLY");
    uint newEpoch = startNewEpoch();
    emit NewEpochForced(newEpoch);
 }
```

The issue here is that while `makeOpenPnlRequest()` updates the open PNL value with each call, thus updating the timestamp, it requires setting the open PNL value 5 times to start a new epoch. On the other hand, `forceNewEpoch()` can be called as long as `block.timestamp - kToken.currentEpochStart() >= requestsStart + requestsEvery * requestsCount`, potentially leading to its execution earlier than the last invocation of `makeOpenPnlRequest()`. In this scenario, a malicious actor could disrupt the operator's plan by calling `forceNewEpoch()` before the last invocation of `makeOpenPnlRequest()` reaches its execution time, thus prematurely initiating a new epoch.

Suggestion:

It is recommended to add permission control to `forceNewEpoch()`.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

KTO-2 `makeOpenPnlRequest()` Lacks Access Control

Severity: Major

Status: Fixed

Code Location:

contracts/vaultv2/KTokenOpenPnlFeed.sol#123-142

Descriptions:

Calling the function `KTokenOpenPnlFeed.makeOpenPnlRequest()` five times triggers the initiation of a new epoch. During the second and fifth calls, the protocol invokes `effectiveOpenPnlRequest()` to update the `nextEpochValues`.

```
 } else if(!firstRequest && block.timestamp - nextEpochValuesLastRequest >=
requestsEvery) { //collect openPnlValue four times
        if (nextEpochValuesRequestCount <= requestsCount) {
            effectiveOpenPnlRequest(requestId, openPnlValue);
        }
```

Upon the fifth call, the protocol invokes `startNewEpoch()` to commence a new epoch.

```
    if(nextEpochValues.length >= requestsCount) {
            startNewEpoch();
        }
```

Within `startNewEpoch()`, the protocol calculates `newEpochOpenPnl` based on `nextEpochValues` and passes it as an argument to `kToken.updateAccPnlPerTokenUsed()` function, consequently updating `accPnlPerToken`, `accPnlPerTokenUsed`, `totalLiability`, and `currentEpochPositiveOpenPnl`. However, m `akeOpenPnlRequest()` lacks permission control, allowing anyone to call it and manipulate `newEpochOpenPnl` by passing a value for `openPnlValue`, thereby affecting `accPnlPerToken`, `accPnlPerTokenUsed`, `totalLiability`, and `currentEpochPositiveOpenPnl`.

```
  function makeOpenPnlRequest(
      uint requestId,
      int openPnlValue // collateral.precision (assets)
  ) external {
    bool firstRequest = nextEpochValuesLastRequest == 0;
```

Suggestion:

It is recommended to add access controls to `makeOpenPnlRequest()`.

Resolution:

This issue has been fixed. The client restricted it to only Operator can call it.

VSR-1 Missing check `depositId`

Severity: Minor

Status: Fixed

Code Location:

contracts/core/VaultStakeReward.sol#641-671

Descriptions:

In the `VaultStakeReward.unlockDeposit()` function, the protocol operates on the corresponding data based on the depositId. However, the protocol does not check the `depositId`, causing transactions beyond `lockedDepositsCount` to fail.

```
function unlockDeposit(uint depositId, address receiver) external {
      LockedDeposit storage d = lockedDeposits[depositId];

      address sender = _msgSender();
      address owner = lockedDepositNft.ownerOf(depositId);

      require(
          owner == sender ||
          lockedDepositNft.getApproved(depositId) == sender ||
          lockedDepositNft.isApprovedForAll(owner, sender),
          "NOT_ALLOWED"
      );
      require(block.timestamp >= d.atTimestamp + d.lockDuration, "NOT_UNLOCKED");

      int accPnlDelta = int(d.assetsDiscount.mulDiv(PRECISION, totalSupply(),
MathUpgradeable.Rounding.Up));
```

Suggestion:

It is recommended to check `depositId < lockedDepositsCount`.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

VSR-2 Lack of Parameter Validation in `initializeV2` Function

Severity: Minor

Status: Fixed

Code Location:

contracts/core/VaultStakeReward.sol#203

Descriptions:

The parameter `_MIN_LOCK_DURATION` in the `initializeV2` function lacks validation, which should be restricted to not exceed `MAX_LOCK_DURATION`. Otherwise, it will result in a validation failure in `validDiscount`, and further modification will not be possible.

Suggestion:

It is recommended to implement parameter validation for `_MIN_LOCK_DURATION` in the `initializeV2` function to ensure it does not exceed `MAX_LOCK_DURATION`.

Resolution:

This issue has been fixed. The client has added validation.

VSR-3 Redundant Constants in `VaultStakeReward` Contract

Severity: Minor

Status: Fixed

Code Location:

contracts/core/VaultStakeReward.sol#63

Descriptions:

There are unused constants `PRECISION_2` in the `VaultStakeReward` contract. it is unnecessary and may cause confusion to readers.

Suggestion:

It is recommended to remove the unused constants `PRECISION_2` from the `VaultStakeReward` contract to improve code clarity and avoid confusion.

Resolution:

This issue has been fixed. The client has removed this constant.

VSR-4 The Implementation of the Withdraw Epochs Timelock is incorrect

Severity: Minor

Status: Fixed

Code Location:

contracts/core/VaultStakeReward.sol#366

Descriptions:

According to the comment, when the CR is greater than or equal to 110% but less than or equal to 120%, it should wait for 2 epochs.

```
//    CR>120%  wait 1Epochs
//    CR=110%-120%  wait 2Epochs
//    CR<110%  wait 3Epochs
```

However, the implemented logic in the protocol is to wait for 2 epochs when `overCollatP` is greater than 110% but less than or equal to 120%.

```
    return
        overCollatP > withdrawLockThresholdsP[1]
            ? WITHDRAW_EPOCHS_LOCKS[2]
            : (overCollatP > withdrawLockThresholdsP[0] ? WITHDRAW_EPOCHS_LOCKS[1] :
WITHDRAW_EPOCHS_LOCKS[0]);
```

Suggestion:

It is recommended to modify the code to check `overCollatP >= withdrawLockThresholdsP[0]` .

Resolution:

This issue has been fixed. The client has updated the comments.

VSR-5 Redundant Inheritance

Severity: Informational

Status: Acknowledged

Code Location:

contracts/core/VaultStakeReward.sol#16

Descriptions:

There is redundancy in the contract inheritance relationship, ERC20Upgradeable is already inherited in ERC4626Upgradeable, and there is no longer a need to inherit the ERC20Upgradeable contract in VaultStakeReward.

```
contract VaultStakeReward is OwnerGovernableUpgradeable,
ReentrancyGuardUpgradeable, PausableUpgradeable, ERC20Upgradeable,
ERC4626Upgradeable {
```

Suggestion:

It is recommended to remove the VaultStakeReward inheritance statement for ERC20Upgradeable and the rewrite relationship between the transfer,decimals and transferFrom functions.

Resolution:

Customer response: Since the agent contract has been deployed, changing the inheritance may change the variable slot, so it can't be modified for the time being, and it will be modified if it is redeployed later.

VSR-6 Unused Function Parameter

Severity: Informational

Status: Fixed

Code Location:

contracts/core/VaultStakeReward.sol#688

Descriptions:

In the `sendAssets` function, `receiver` is passed as a function parameter but not used in the function.

Suggestion:

It is recommended to remove the unused variable if there's no further design.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

VSR-7 Inconsistent Comment Style and Non-English Comments

Severity: Informational

Status: Fixed

Code Location:

contracts/core/VaultStakeReward.sol#83

Descriptions:

Non-English comments are present in the code, which is inconsistent with the overall comment style.

```
uint public shareToAssetsPrice; // PRECISION     kUSDT price
```

Suggestion:

It is recommended to maintain a uniform comment style throughout the codebase and use English comments for consistency.

Resolution:

This issue has been fixed. The client has updated the comments.

KTL1-1 Incomplete Functionality in `numberToRoundedString()`

Severity: Minor

Status: Fixed

Code Location:

contracts/vaultv2/KTokenLockedDepositNftDesign.sol#80

Descriptions:

The functionality of the function `numberToRoundedString()` appears to be incomplete. The function currently only returns the integer part of the number as a string and does not consider the specified output decimal places.

```
// Returns readable string of int part of number passed
// TODO: make it return the string with decimals = 'outputDecimals'
function numberToRoundedString(
    uint number,
    uint8 inputDecimals,
    uint8 outputDecimals
) public pure returns (string memory) {
    outputDecimals = 0; // silence warning
    return Strings.toString(number / (10 ** inputDecimals));
}
```

Suggestion:

It is recommended to complete the functionality of the `numberToRoundedString()` function by implementing the logic to return a string representation with the specified number of decimal places.

Resolution:

This issue has been fixed. The client has removed information about `outputDecimals`.

Appendix 1

Issue Level

- **Informational** issues are often recommendations to improve the style of the code or to optimize code that does not affect the overall functionality.
- **Minor** issues are general suggestions relevant to best practices and readability. They don't post any direct risk. Developers are encouraged to fix them.
- **Medium** issues are non-exploitable problems and not security vulnerabilities. They should be fixed unless there is a specific reason not to.
- **Major** issues are security vulnerabilities. They put a portion of users' sensitive information at risk, and often are not directly exploitable. All major issues should be fixed.
- **Critical** issues are directly exploitable security vulnerabilities. They put users' sensitive information at risk. All critical issues should be fixed.

Issue Status

- **Fixed:** The issue has been resolved.
- **Partially Fixed:** The issue has been partially resolved.
- **Acknowledged:** The issue has been acknowledged by the code owner, and the code owner confirms it's as designed, and decides to keep it.

Appendix 2

Disclaimer

This report is based on the scope of materials and documents provided, with a limited review at the time provided. Results may not be complete and do not include all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your own risk. A report does not imply an endorsement of any particular project or team, nor does it guarantee its security. These reports should not be relied upon in any way by any third party, including for the purpose of making any decision to buy or sell products, services, or any other assets. TO THE FULLEST EXTENT PERMITTED BY LAW, WE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, IN CONNECTION WITH THIS REPORT, ITS CONTENT, RELATED SERVICES AND PRODUCTS, AND YOUR USE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NOT INFRINGEMENT.