SALUS SECURITY

SEP 2025

# CODE SECURITY ASSESSMENT

ZEROBASE

# Overview

## Project Summary

- Name: ZeroBase - V2
- Platform: EVM-compatible chains
- Language: Solidity
- Repository:
    - https://github.com/ZeroBase-Pro/ZKFi
- Audit Range: See Appendix - 1

# Project Dashboard

## Application Summary

| | |
|---|---|
| **Name** | ZeroBase - V2 |
| **Version** | v2 |
| **Type** | Solidity |
| **Dates** | Sep 16 2025 |
| **Logs** | Aug 22 2025; Sep 16 2025 |

## Vulnerability Summary

| | |
|---|---|
| **Total High-Severity issues** | 0 |
| **Total Medium-Severity issues** | 0 |
| **Total Low-Severity issues** | 3 |
| **Total informational issues** | 2 |
| **Total** | 5 |

## Contact

E-mail: support@salusec.io

# Risk Level Description

| | |
|---|---|
| **High Risk** | The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for clients' reputations or serious financial implications for clients and users. |
| **Medium Risk** | The issue puts a subset of users' sensitive information at risk, would be detrimental to the client's reputation if exploited, or is reasonably likely to lead to a moderate financial impact. |
| **Low Risk** | The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client's business circumstances. |
| **Informational** | The issue does not pose an immediate risk, but is relevant to security best practices or defense in depth. |

# Content

# Introduction

## 1.1 About SALUS

At Salus Security, we are in the business of trust.

We are dedicated to tackling the toughest security challenges facing the industry today. By building foundational trust in technology and infrastructure through security, we help clients to lead their respective industries and unlock their full Web3 potential.

Our team of security experts employ industry-leading proof-of-concept (PoC) methodology for demonstrating smart contract vulnerabilities, coupled with advanced red teaming capabilities and a stereoscopic vulnerability detection service, to deliver comprehensive security assessments that allow clients to stay ahead of the curve.

In addition to smart contract audits and red teaming, our Rapid Detection Service for smart contracts aims to make security accessible to all. This high calibre, yet cost-efficient, security tool has been designed to support a wide range of business needs including investment due diligence, security and code quality assessments, and code optimisation.

We are reachable on Telegram (https://t.me/salusec), Twitter (https://twitter.com/salus_sec), or Email (support@salusec.io).

## 1.2 Audit Breakdown

The objective was to evaluate the repository for security-related issues, code quality, and adherence to specifications and best practices. Possible issues we looked for included (but are not limited to):

- Risky external calls
- Integer overflow/underflow
- Transaction-ordering dependence
- Timestamp dependence
- Access control
- Call stack limits and mishandled exceptions
- Number rounding errors
- Centralization of power
- Logical oversights and denial of service
- Business logic specification
- Code clones, functionality duplication

## 1.3 Disclaimer

Note that this security audit is not designed to replace functional tests required before any software release and does not give any warranties on finding all possible security issues with the given smart contract(s) or blockchain software, i.e., the evaluation result does not guarantee the nonexistence of any further findings of security issues.

# Findings

## 2.1 Summary of Findings

| ID | Title | Severity | Category | Status |
|---|---|---|---|---|
| 1 | Centralization risk | Low | Centralization | Acknowledged |
| 2 | The deprecated admin still retains the pause permission | Low | Business Logic | Acknowledged |
| 3 | Lack of remove supported token function | Low | Business Logic | Acknowledged |
| 4 | Unnecessary lastRewardUpdateTime update | Informational | Redundancy | Acknowledged |
| 5 | Gas optimization suggestions | Informational | Gas Optimization | Acknowledged |

## 2.2 Notable Findings

Significant flaws that impact system confidentiality, integrity, or availability are listed below.

### 1. Centralization risk

| Severity: Low | Category: Centralization |
| --- | --- |
| Target:<br>- V2/src/WithdrawVault.sol<br>- V2/src/Vault.sol | |

### Description

In the Vault contract, there exists one privileged role, `DEFAULT_ADMIN_ROLE`. The role has the authority to execute some key functions such as `setRewardRate`, `setPenaltyRate` and `setCeffu`, etc.

If the role's private key is compromised, an attacker could trigger these functions to steal remaining funds in the vault.

V2/src/Vault.sol: L714-L720

```
function setCeffu(address _newCeffu) external onlyRole(DEFAULT_ADMIN_ROLE) {
    Utils.CheckIsZeroAddress(_newCeffu);
    require(_newCeffu != ceffu);

    emit UpdateCeffu(ceffu, _newCeffu);
    ceffu = _newCeffu;
}
```

### Recommendation

We recommend transferring privileged accounts to multi-sig accounts with timelock governors for enhanced security. This ensures that no single person has full control over the accounts and that any changes must be authorized by multiple parties.

### Status

This issue has been acknowledged by the team.

## 2. The deprecated admin still retains the pause permission

| Severity: Low | Category: Business Logic |
| --- | --- |
| Target:<br>- V2/src/WithdrawVault.sol | |

## Description

The changeAdmin function updates the contract admin but does not reassign the `PAUSER_ROLE`. This can leave the previous admin with `pause/unpause` privileges, resulting in inconsistent access control and potential abuse.

V2/src/WithdrawVault.sol: L24-L41, L83-L87

```
constructor(address[] memory tokens, address admin, address bot, address _ceffu) {
    ...
    // Grant admin roles
    _grantRole(DEFAULT_ADMIN_ROLE, admin);
    _grantRole(PAUSER_ROLE, admin);
    _grantRole(BOT_ROLE, bot);
}
function changeAdmin(address _admin) external onlyRole(DEFAULT_ADMIN_ROLE) {
    require(_admin != address(0), "Admin address cannot be zero");
    _revokeRole(DEFAULT_ADMIN_ROLE, msg.sender);
    _grantRole(DEFAULT_ADMIN_ROLE, _admin);
}
```

## Recommendation

Add a revoke for the `PAUSER_ROLE` in the `changAdmin` function.

## Status

This issue has been acknowledged by the team.

### 3. Lack of remove supported token function

| Severity: Low | Category: Business Logic |
| --- | --- |
| Target:<br>- V2/src/WithdrawVault.sol<br>- V2/src/Vault.sol | |

## Description

The contract lacks a mechanism to remove supported tokens. Once a token has been added, it cannot be removed from the system even if it has a security vulnerability or needs to be delisted, which increases the overall security risk for the contract.

## Recommendation

Implement an admin function that allows for the removal or deactivation of tokens.

## Status

This issue has been acknowledged by the team.

# 2.3 Informational Findings

### 4. Unnecessary lastRewardUpdateTime update

| Severity: Informational | Category: Redundancy |
| --- | --- |
| Target:<br>- V2/src/Vault.sol | |

## Description

When users want to transfer their shares, we will update both `from` and `to` accounts' reward state.

The problem here is that in the function `_updateRewardState`, we will update this account's lastRewardUpdateTime. But we will update this again in the function `_assetsInfoUpdate`. This update is redundant.

V2/src/Vault.sol: L592-L615

```
function _assetsInfoUpdate(address token, address from, address to, uint256 amount,
uint256 tokenBefore) internal{
    _updateRewardState(from, token);
    _updateRewardState(to, token);
    ...
    assetsInfoTo.lastRewardUpdateTime = block.timestamp ;
}
```

V2/src/Vault.sol: L494-L512

```
function _updateRewardState(address _user, address _token) internal {
    AssetsInfo storage assetsInfo = userAssetsInfo[_user][_token];
    ...
    newAccumulatedRewardForAll = _getClaimableRewards(address(this), _token);
    // This user's rewards will be recorded into the accumulatedReward.
    assetsInfo.accumulatedReward = newAccumulatedReward;
    // update the last reward update time.
    assetsInfo.lastRewardUpdateTime = block.timestamp;

    _lastRewardUpdatedTime[_token] = block.timestamp;
    totalRewardsAmountByToken[_token] = newAccumulatedRewardForAll;
}
```

## Recommendation

Suggest removing the unnecessary lastRewardUpdateTime update.

## Status

This issue has been acknowledged by the team.

## 5. Gas optimization suggestions

| Severity: Informational | Category: Gas Optimization |
| --- | --- |
| Target:<br>- V2/src/Vault.sol | |

## Description

**Finding 1:** The function `transferOrTransferFrom` is declared as `public`, but there is no indication that it needs to be called internally by the contract.

Using `public` instead of `external` introduces unnecessary overhead, since Solidity will copy input arguments into memory. Declaring the function as `external` can save a small amount of gas on external calls.

V2/src/Vault.sol: L548-L569

```
function transferOrTransferFrom(address token, address from, address to, uint256 amount)
public returns (bool) {
      ...
}
```

**Finding 2:** Memory reading saves more gas than storage reading multiple times when the state is not changed. So caching the storage variables in memory and using the memory instead of storage reading is effective. Cache array length outside of the loop can save gas.

V2/src/Vault.sol:L278, L320

```
for(uint256 i = 0; i < pendingClaimQueueIDs.length; i++) {
```

## Recommendation

Consider using the above suggestions to save gas.

## Status

This issue has been acknowledged by the team.

# Appendix

## Appendix 1 - Files in Scope

This audit covered the following files in commit 3391353:

| File | SHA-1 hash |
|---|---|
| V2/src/IVault.sol | c0805d4be6a97c36cf2cf8ae30462abdd6fa0c7d |
| V2/src/IzkToken.sol | 59f56ae27e26003e909bf8e944375549703ee61a |
| V2/src/IWithdrawVault.sol | cc903f6d1e58f87a8f358c60bfe79e83abcd419d |
| V2/src/utils.sol | 787b1f5b70928dd2a6767541f6628777c2361b2f |
| V2/src/Vault.sol | e14530676bb327831d3361611062afcda4b88317 |
| V2/src/WithdrawVault.sol | b0874a98d72ab698177344e2fc6ed28cff60d3d4 |
| V2/src/zkToken.sol | 825d0ecd48564f77477d89295c20c7cb9d24d832 |