SALUS SECURITY

AUG 2025

CODE SECURITY ASSESSMENT

DEFIHOMI

Overview

Project Summary

- Name: DefiHomi
- Platform: EVM-compatible chains
- Language: Solidity
- Repository:
  - https://github.com/Danbaba1882/defihomi-contracts
- Audit Range: See Appendix - 1

Project Dashboard

Application Summary

Name	DefiHomi
Version	v2
Type	Solidity
Dates	Aug 25 2025
Logs	Aug 18 2025; Aug 25 2025

Vulnerability Summary

Total High-Severity issues	0
Total Medium-Severity issues	0
Total Low-Severity issues	4
Total informational issues	6
Total	10

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	Missing Docstrings	Low	Code Quality	Resolved
2	Missing State Change Validation	Low	Configuration	Resolved
3	Centralization risk	Low	Centralization	Mitigated
4	Lack of testing	Low	Configuration	Acknowledged
5	Variables Initialized With Their Default Values	Informational	Code Quality	Resolved
6	Lack of Security Contact	Informational	Configuration	Resolved
7	Missing Named Parameters in Mapping	Informational	Code Quality	Resolved
8	Duplicate Imports	Informational	Redundancy	Resolved
9	Missing two-step transfer ownership pattern	Informational	Business Logic	Resolved
10	Use of floating pragma	Informational	Configuration	Resolved

2.2 Notable Findings

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

1. Missing Docstrings	
Severity: Low	Category: Code Quality
Target: - contracts/DefiHomi.sol	

Description

The DefiHomi.sol file currently has very limited documentation. For instance, the DefiHomi contract, its functions, state variables, and custom errors lack comprehensive documentation.

Consider thoroughly documenting all contracts and their functions (and their parameters) that are part of any contract's public API. Functions implementing sensitive functionality, even if not public, should be clearly documented as well. The same approach should be applied to custom errors and events, including their parameters. When writing docstrings, consider following the Ethereum Natural Specification Format (NatSpec).

Recommendation

It is recommended that comprehensive documentation be added to the contract.

Status

This issue has been resolved by the team at commit 227d1b2.

2. Missing State Change Validation	
Severity: Low	Category: Configuration
Target: - contracts/DefiHomi.sol	

Description

Throughout the codebase, multiple instances of functions that do not verify whether the new value actually differs from the existing one before updating were identified:

- The `` `restrictAddress` `` function in DefiHomi.sol
- The `` `unrestrictAddress` `` function in DefiHomi.sol
- The `` `setRestrictionsBatch` `` function in DefiHomi.sol

Recommendation

Consider adding validation checks that revert the transaction if the input value matches the existing value.

Status

This issue has been resolved by the team at commit 227d1b2.

3. Centralization risk	
Severity: Low	Category: Centralization
Target: - contracts/DefiHomi.sol	

Description

The `DefiHomi` contract contains a privileged account. This privileged account can restrict the transfer functionality of a specific account and control whether the contract is paused.

If the private key of the owner address is compromised, an attacker could arbitrarily restrict normal users' transfer functions or directly halt the operation of the contract.

If the privileged accounts are plain EOA accounts, this can be worrisome and pose a risk to the other users.

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 mitigated by the team by using a multisig account at commit 227d1b2.

4. Lack of testing	
Severity: Low	Category: Configuration
Target: - test/test.ts	

Description

Currently, although test files are present, the test coverage is low. Only the Token name is tested for expected results, while other functionalities are not verified against expected behavior.

Contract security and dependability are at risk because defects or regressions might be undiscovered.

Recommendation

It is recommended to increase the test coverage.

Status

This issue has been acknowledged by the team.

2.3 Informational Findings

5. Variables Initialized With Their Default Values	
Severity: Informational	Category: Code Quality
Target: - contracts/DefiHomi.sol	

Description

contracts/DefiHomi.sol:L38

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

The above variable `i` was initialized using its default value.

Recommendation

To avoid wasting gas, consider not initializing variables with their default values.

Status

This issue has been resolved by the team at commit 227d1b2.

6. Lack of Security Contact	
Severity: Informational	Category: Configuration
Target: - contracts/DefiHomi.sol	

Description

Providing a specific security contact (such as an email or ENS name) within a smart contract significantly simplifies the process for individuals to communicate if they identify a vulnerability in the code. This practice is quite beneficial as it permits the code owners to dictate the communication channel for vulnerability disclosure, eliminating the risk of miscommunication or failure to report due to a lack of knowledge on how to do so. In addition, if the contract incorporates third-party libraries and a bug surfaces in those, it becomes easier for their maintainers to contact the appropriate person about the problem and provide mitigation instructions.

The ``DefiHomi`` contract does not have a security contact.

Recommendation

Consider adding a NatSpec comment containing a security contact above each contract definition. Using the ``@custom:security-contact`` convention is recommended as it has been adopted by the OpenZeppelin Wizard and the ethereum-lists.

Status

This issue has been resolved by the team at commit 227d1b2.

7. Missing Named Parameters in Mapping	
Severity: Informational	Category: Code Quality
Target: - contracts/DefiHomi.sol	

Description

Since Solidity 0.8.18, developers can utilize named parameters in mappings. This means that mappings can take the form of `` `mapping(KeyType KeyName? => ValueType ValueName?)` ``. This updated syntax provides a more transparent representation of a mapping's purpose.

Recommendation

Consider adding at least one named parameter to the key type in the aforementioned mapping in order to improve the readability and maintainability of the codebase.

Status

This issue has been resolved by the team at commit 227d1b2.

8. Duplicate Imports	
Severity: Informational	Category: Redundancy
Target: - contracts/DefiHomi.sol	

Description

In the `DefiHomi` contract, a duplicate import instance was found:

The `DefiHomi.sol` file imports `ERC20.sol` which is already imported as a result of importing `ERC20Pausable.sol`.

Recommendation

It is recommended to remove duplicate import statements and explicitly import ERC20 while importing ERC20Pausable to improve the overall clarity and readability of the code base.

Status

This issue has been resolved by the team at commit 227d1b2.

9. Missing two-step transfer ownership pattern	
Severity: Informational	Category: Business logic
Target: - contracts/DefiHomi.sol	

Description

The `DefiHomi` contract inherits from the `Ownable` contract. This contract does not implement a two-step process for transferring ownership. Thus, ownership of the contract can easily be lost when making a mistake in transferring ownership.

Recommendation

Consider using the Ownable2Step contract from OpenZeppelin instead.

Status

This issue has been resolved by the team at commit 227d1b2.

10. Use of floating pragma	
Severity: Informational	Category: Configuration
Target: - contracts/DefiHomi.sol	

Description

```
pragma solidity ^0.8.27;
```

The `DefiHomi` contract uses a floating compiler version `^0.8.27`.

Using a floating pragma `^0.8.27` statement is discouraged, as code may compile to different bytecodes with different compiler versions. Use a locked pragma statement to get a deterministic bytecode. Also use the latest Solidity version to get all the compiler features, bug fixes and optimizations.

Recommendation

It is recommended to use a locked Solidity version throughout the project. It is also recommended to use the most stable and up-to-date version.

Status

This issue has been resolved by the team at commit 227d1b2.

Appendix

Appendix 1 - Files in Scope

This audit covered the following files in commit 394bc30:

File	SHA-1 hash
contracts/DefiHomi.sol	6f91728e9023d11f5a5d12fc967b16a66537a4dc