yAudit

November 25, 2025

Prepared for
HAI

Audited by
Adriro
HHK

haiVELO V2 review

Smart Contract Security Assessment

Contents

1 Review Summary

1.1 Protocol Overview

haiVELO V2 allows users to use veVELO NFT to mint haiVELO token, while the V1 only allowed VELO tokens. User from haiVELO V1 can easily migrate to the new version.

1.2 Audit Scope

This audit covers one smart contract totaling approximately 120 lines of code across 1.5 days of review.

`core/src/contracts/tokens/WrappedTokenV2.sol`

1.3 Risk Assessment Framework

1.3.1 Severity Classification

Severity	Description	Potential Impact
Critical	Immediate threat to user funds or protocol integrity	Direct loss of funds, protocol compromise
High	Significant security risk requiring urgent attention	Potential fund loss, major functionality disruption
Medium	Important issue that should be addressed	Limited fund risk, functionality concerns
Low	Minor issue with minimal impact	Best practice violations, minor inefficiencies
Undetermined	Findings whose impact could not be fully assessed within the time constraints of the engagement. These issues may range from low to critical severity, and although their exact consequences remain uncertain, they present a sufficient potential risk to warrant attention and remediation.	Varies based on actual severity
Gas	Findings that can improve the gas efficiency of the contracts.	Reduced transaction costs
Informational	Code quality and best practice recommendations	Improved maintainability and readability

Table 1: severity classification

1.4 Key Findings

Breakdown of Finding Impacts

Impact Level	Count
Critical	0
High	0
Medium	0
Low	0
Informational	5

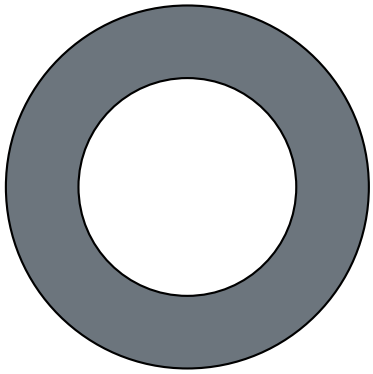

Figure 1: Distribution of security findings by impact level

1.5 Overall Assessment

The contract is simple and well organized, the team was prompt in fixing all issues. Some concerns were raised regarding the usage of haiVELO as collateral and if the peg to VELO always holds.

2 Audit Overview

2.1 Project Information

Protocol Name: HAI
Repository: https://github.com/hai-on-op/core/
Commit Hash: 25f3a1a571d71a1e49599b6f07521517ad146ecb
Commit URL:
https://github.com/hai-on-op/core/commit/25f3a1a571d71a1e49599b6f07521517ad146ecb

2.2 Audit Team

Adriro, HHK

2.3 Audit Timeline

The audit was conducted from August 15 to July 16, 2025.

2.4 Audit Resources

Code repositories and a Hackmd describing the v2.

Category	Mark	Description
Access Control	Good	No access control mechanisms are needed for this protocol's design.
Mathematics	Good	The protocol uses only basic mathematical operations.
Complexity	Good	The codebase is straightforward and easy to understand.
Libraries	Good	The project uses battle-tested OpenZeppelin libraries.
Decentralization	Low	The backing of haiVELO is stored on a Safe multisig, introducing centralization risks.
Code Stability	Good	The codebase remained stable throughout the audit period.
Documentation	Good	NatSpec documentation is present along with existing developer documentation available on the website.
Monitoring	Average	Events are emitted when needed, though a few minor gaps were identified.
Testing and verification	Good	Comprehensive test coverage is implemented.

Table 2: Code Evaluation Matrix

2.5 Critical Findings

None.

2.6 High Findings

None.

2.7 Medium Findings

None.

2.8 Low Findings

None.

2.9 Gas Savings Findings

2.9.1 Burn address can be constant

The `BURN_ADDRESS` is known at compilation time and can be defined as a constant.

Technical Details

- WrappedTokenV2.sol#L51

Impact

Gas savings.

Recommendation

Change the variable to a constant.

Developer Response

Fixed in be1e037bd0b571bcb940d789ecbbe471e073b363.

2.10 Informational Findings

2.10.1 Incorrect account argument in events

Technical Details

The events should notify about the user depositing or migrating, but are instead emitted with the receiver.
As an example, in the `WrappedTokenV2Deposit` event, the `_account` argument refers to the *user depositing the base tokens*.

```
20:   /**
21:    * @notice Emitted when a user deposits tokens and mints wrapped tokens
22:    * @param _account Address of the user depositing the base tokens
23:    * @param _wad Amount of tokens deposited
24:    */
25:   event WrappedTokenV2Deposit(address indexed _account, uint256 _wad);
```

However, the event is then emitted with the `_account` parameter, the receiver, which may differ from the caller.

```
100:     emit WrappedTokenV2Deposit(_account, _wad);
```

A second event should be emitting the amount of Velo per NFT locked, but is emitting the sum of all Velo locked instead.
The interface:

```
27:  /**
28:   * @notice Emitted when a user deposits a veNFT and mints wrapped tokens
29:   * @param _account Address of the user depositing the base tokens
30:   * @param _tokenId ID of the veNFT being deposited
31:   * @param _wad Amount of locked velo in veNFT being deposited
32:   */
33:  event WrappedTokenV2NFTDeposit(address indexed _account, uint256 _tokenId, uint256
_wad);
```

However, inside the function, it uses the sum of all NFTs locked instead of the amount per tokenId.

```
110: _balance += uint256(uint128(BASE_TOKEN_NFT.locked(_tokenIds[i]).amount));
...
119: emit WrappedTokenV2NFTDeposit(_account, _tokenIds[i], _balance);
```

Impact

Informational.

Recommendation

Change event arguments to `msg.sender`, or change the associated documentation to note that this address is the token recipient and not the originator of the action.

Developer Response

Fixed in 0fde26277968341029a57ac77fb8596153996312 & 757e12b42236c4f8859853fee113f1f22780c25e & 937ab3752c43f08ab6ae1d173b78b04d7732f7f2.

2.10.2 Inaccurate supply on NFT transfer callback

The `depositNFTs()` function transfers the tokens before minting haiVELO, potentially leading to an incorrect supply if read from the NFT transfer callback.

Technical Details

WrappedTokenV2.sol#L104

Impact

Informational.

Recommendation

Take into account this desynchronization in future manager implementations. Another alternative would be to follow a CEI approach and mint before transferring.

Developer Response

Fixed in 64d57eb84b67bb088e221a6158a54740e64ef6ca.

2.10.3 haiVELO backing relies on trusted multisig custody

Technical Details

haiVELO tokens maintain a 1:1 backing with VELO tokens through either pure VELO tokens or veVELO NFTs. However, during the minting process, the backing VELO tokens are transferred to the `baseTokenManager`, which is a Safe multisig controlled by the HAI team. This introduces a trust assumption where any party with access to the Safe could mint additional haiVELO tokens without proper backing, breaking the 1:1 peg.

Impact

Informational.

Recommendation

Store the backing VELO tokens and veVELO NFTs in an immutable smart contract that enforces the 1:1 backing mechanically, removing the need for trusted custodians.

Developer Response

Acknowledged. We are in the process of writing a contract to handle this.

2.10.4 Validate duplicates in tokens array

Even though the implementation should fail on transferring a duplicate token ID, the `depositNFTs()` function could validate that there are no repeated IDs in the `_tokenIds` array.

Technical Details

WrappedTokenV2.sol#L104

Impact

Informational.

Recommendation

Validate token IDs are not repeated in the `_tokenIds` array.

```
    uint256 _balance = 0;
    for (uint256 i = 0; i < _tokenIds.length; i++) {
      _balance += uint256(uint128(BASE_TOKEN_NFT.locked(_tokenIds[i]).amount));
+     if (i > 0 && _tokenIds[i-1] >= _tokenIds[i]) {
+       revert WrappedTokenV2_DuplicateTokenId();
+     }
    }
```

Developer Response

Fixed in 74b2ffcfbc3e9ceda0a63fc5e21a707676bc60ca.

2.10.5 V1 total supply isn't adjusted when migrating

The implementation of `migrateV1toV2()` transfers V1 tokens to a burn address without adjusting the supply, while new V2 tokens are minted.

Technical Details

WrappedTokenV2.sol#L130

Impact

Informational.

Recommendation

Ensure no component depends on the total supply of the old token, and that the burn address balance is accounted for when calculating the circulating supply.

Developer Response

Acknowledged. No fix needed.

2.11 Final Remarks

The protocol features a well-organized and straightforward codebase. The development team was responsive and promptly addressed all identified issues.
The auditors raised concerns regarding the use of haiVELO as collateral, particularly around the sustainability of its peg to VELO.