SOTA LABS
R&D Center of SotaTek

Smart Contract Security Assessment

Final Report

Vemo.network

26 September 2024

Table of contents

1. Overview

This report has been prepared for Vemo.network. Sotatek provides a user-centered examination of smart contracts to look for vulnerabilities, logic errors or other issues from both an internal and external perspective.

A comprehensive examination has been performed, utilizing Cross Referencing, Static Analysis, In-House Security Tools, and line-by-line Manual Review. The auditing process pays special attention to the following considerations:

- Ensuring contract logic meets the specifications and intentions of the

client without exposing the user's funds to risk.

- Testing the smart contracts against both common and uncommon attack vectors.
- Inspecting liquidity and holders' statistics to inform the status to both users and clients when applicable.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Verifying contract functions that allow trusted and/or untrusted actors to mint, lock, pause, and transfer assets.
- Thorough line-by-line manual review of the entire codebase by industry experts.

Summary

Project Name	Vemo network
Type	Defi
Platform	Arbitrum One, Ethereum, Binance Smart Chain
Language	Solidity
Auditors	Sotatek
Timeline	Sep 2, 2024 – Sep 26, 2024
Description	Vemo Network utilizes ERC-721 and ERC-6551 standards to tokenize locked positions

	into NFT Accounts or Smart Vouchers. These tokenized assets enable trading, lending, and fundraising. NFT Accounts function like independent accounts, self-custodying assets and performing operations across chains. Smart Vouchers store assets and vesting details, allowing holders to claim tokens or sell them on secondary markets.

1. Audit Summary

Delivery Date	Sep 26, 2024
Audit Methodology	Static Analysis, Manual Review

2. Vulnerability Summary

Vulnerability	Total	Pending	Resolved	Acknowledged
Critical	0	0	0	0
Major	0	0	0	0
Medium	5	0	0	5
Informational	0	1	0	1

Classification Of Issues

Severity	Description
Critical	Exploits, vulnerabilities, or errors that will certainly or probabilistically lead towards loss of funds, control, or impairment of the contract and its functions. Issues under this

	classification are recommended to be fixed with utmost urgency.
Major	Bugs or issues with that may be subject to exploitation, though their impact is somewhat limited. Issues under this classification are recommended to be fixed as soon as possible
Medium	Effects are minimal in isolation and do not pose a significant danger to the project or its users. Issues under this classification are recommended to be fixed nonetheless.
Informational	Consistency, syntax, or style best practices. Generally, pose a negligible level of risk, if any.

3. Audit Scope

Delivery Date	September 26, 2023
Audit Methodology	Static Analysis, Manual Review

The following files were made available during the review:

File	Interfaces
src/cross-chain/FxChildExecutor.sol	1
src/interfaces/ICollectionDeployer.sol	1
src/interfaces/IDelegationCollection.sol	1
src/interfaces/IExecutionTerm.sol	1
src/interfaces/IWalletFactory.sol	1
src/interfaces/ISandboxExecutor.sol	1
src/interfaces/erc6551/IAccountProxy.sol	1
src/interfaces/erc6551/IERC6551Executable.sol	1
src/interfaces/erc6551/IAccountRegistry.sol	1
src/interfaces/IAccountGuardian.sol	1
src/abstract/Overridable.sol	
src/abstract/Signatory.sol	
src/abstract/execution/ERC6551Executor.sol	

src/abstract/execution/BatchExecutor.sol	
src/abstract/execution/NestedAccountExecutor.sol	
src/abstract/execution/SandboxExecutor.sol	
src/abstract/execution/BaseExecutor.sol	
src/abstract/execution/TokenboundExecutor.sol	
src/abstract/ERC4337Account.sol	
src/abstract/Permissioned.sol	
src/abstract/ERC6551Account.sol	
src/abstract/Lockable.sol	
src/WalletFactory.sol	
src/terms/VePendleTerm.sol	
src/accounts/AccountV3Upgradable.sol	
src/accounts/AccountProxy.sol	
src/accounts/AccountV3.sol	
src/accounts/NFTAccountDelegable.sol	
src/lib/LibSandbox.sol	
src/lib/LibExecutor.sol	
src/lib/OPAddressAliasHelper.sol	
src/lib/IterableMap.sol	
src/CollectionDeployer.sol	
src/AccountGuardian.sol	
src/helpers/VemoDelegationCollection.sol	
src/helpers/VemoWalletCollection.sol	
src/AccountRegistry.sol	
Totals	38

2. Finding

1. VEMO-01 | Mixed solidity versions

Category	Severity	Location	Status
Volatile Code	Medium	Multiple location	Acknowledged

Description

The smart contract codebase uses multiple Solidity compiler versions (0.8.13, 0.8.20, and 0.8.21) across different files. This inconsistency can lead to several problems:

1. **Compatibility Issues**: Different versions may have incompatible features or behaviors, potentially causing unexpected interactions between contracts.
2. **Security Risks**: Older versions might lack important security fixes or optimizations present in newer versions.
3. **Maintenance Challenges**: Managing multiple compiler versions complicates the development and deployment processes.
4. **Audit Complexity**: Inconsistent versions make it harder to conduct a thorough security audit, as each version needs to be considered separately.

Recommendation

Standardize the Solidity compiler version across all smart contracts. We recommend using the most recent stable version (0.8.21 in this case) for all contracts, unless there's a specific reason to use an older version. If upgrading all contracts to the latest version is not immediately feasible, consider the following:

1. Document the reasons for using different versions.
2. Plan a gradual migration to a single, up-to-date version.
3. Ensure thorough testing of inter-contract interactions, especially between contracts using different versions.

2. VEMO-02 | Missing zero checks

Category	Severity	Location	Status
Volatile Code	Medium	Multiple location	Acknowledged

Description

A number of constructors and functions in the codebase do not revert if zero is passed in for a parameter that should not be set to zero. The following parameters are not checked for the zero value

```
src/WalletFactory.sol#76)
  - owner = _owner (src/WalletFactory.sol#95)
  - feeReceiver = _owner (src/WalletFactory.sol#102)
src/WalletFactory.sol#230
  - collectionDeployer = _deployer (src/WalletFactory.sol#231)
src/WalletFactory.sol#252
  - accountRegistry = _accountRegistry (src/WalletFactory.sol#253)
src/WalletFactory.sol#256
  - walletImpl = _walletImpl (src/WalletFactory.sol#257)
src/WalletFactory.sol#260
  - walletProxy = _walletProxy (src/WalletFactory.sol#261)
src/WalletFactory.sol#271
  - feeReceiver = _receiver (src/WalletFactory.sol#272)
```

if zero is passed in for one of those parameters, it will render the contract unusable, leaving its funds locked (and therefore effectively lost) and necessitating an expensive redeployment.

Recommendation

Short term, add zero checks for the parameters mentioned above and for all other parameters for which zero is not an acceptable value.

Long term, comprehensively validate all parameters. Avoid relying solely on the validation performed by front-end code, scripts, or other contracts, as a bug in any of those components could prevent it from performing that validation. Additionally, integrate ***Slither*** into the CI pipeline to automatically detect functions that lack zero check.

3. VEMO-03 | Missing events for onlyAdmin functions

Category	**Severity**	**Location**	**Status**
Volatile Code	Medium	src/WalletFactory	Acknowledged

Description

onlyAdmin functions that change critical contract parameters/addresses/state should emit events and consider adding timelocks so that users and other privileged roles can detect upcoming changes (by offchain monitoring of events) and have the time to react to them.

```
- WalletFactory.setWalletCollection(uint160,address) (src/WalletFactory.sol#224-228)
- WalletFactory.setCollectionDeployer(address) (src/WalletFactory.sol#230-232)
- WalletFactory.setAccountRegistry(address) (src/WalletFactory.sol#252-254)
- WalletFactory.setWalletImpl(address) (src/WalletFactory.sol#256-258)
- WalletFactory.setWalletProxy(address) (src/WalletFactory.sol#260-262)
- WalletFactory.setFeeReceiver(address) (src/WalletFactory.sol#271-273)
- WalletFactory.setFee(uint256,uint256) (src/WalletFactory.sol#275-278)
```

Recommendation

Add events to all possible flows (some flows emit events in callers) and consider adding timelocks to such onlyAdmin functions.

4. VEMO-04 | Unused Import

Category	Severity	Location	Status
Volatile Code	Informational	Multiple location	Acknowledged

Description

Importing a file that is not used in the contract likely indicates a mistake. The import should be removed until it is needed:

```
- src/abstract/ERC6551Account.sol#4
- src/abstract/execution/SandboxExecutor.sol#4
- src/abstract/execution/ERC6551Executor.sol#9
- src/abstract/execution/NestedAccountExecutor.sol#7
```

Recommendation

Remove the unused import. If the import is needed later, it can be added back.

5. VEMO-05 | Dead Code

Category	Severity	Location	Status
Volatile Code	Medium	Several	Acknowledged

Description

Functions that are not used.

- src/abstract/Overridable.sol#97

Recommendation

Remove unused functions.

6. VEMO-06 | Block timestamp

Category	Severity	Location	Status
Volatile Code	Medium	src/abstract/Lockable.sol#47-49	Acknowledged

Description

The usage of block.timestamp. ***block.timestamp*** can be manipulated by miners.

Recommendation

Avoid relying on block.timestamp.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e., incorrect usage of private or deleted.