Ether Authority

SMART CONTRACT

Security Audit Report

Project: Staked Aave Token
Website: aave.com
Platform: Ethereum
Language: Solidity
Date: April 24th, 2024

Table of contents

`

THIS IS A SECURITY AUDIT REPORT DOCUMENT AND WHICH MAY CONTAIN INFORMATION WHICH IS CONFIDENTIAL. WHICH INCLUDES ANY POTENTIAL VULNERABILITIES AND MALICIOUS CODES WHICH CAN BE USED TO EXPLOIT THE SOFTWARE. THIS MUST BE REFERRED INTERNALLY AND ONLY SHOULD BE MADE AVAILABLE TO THE PUBLIC AFTER ISSUES ARE RESOLVED.

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.

Email: audit@EtherAuthority.io

Introduction

As part of EtherAuthority's community smart contracts audit initiatives, the smart contracts of Staked Aave Token from aave.com were audited. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on April 24th, 2024.

The purpose of this audit was to address the following:

- Ensure that all claimed functions exist and function correctly.
- Identify any security vulnerabilities that may be present in the smart contract.

Project Background

- The StakedTokenV3 contract is designed to enhance the staking functionality, offering better flexibility, security, and management of staked tokens, including handling slashing events, managing roles, and maintaining accurate exchange rates.
- Staked Aave Contracts handle multiple contracts, and all contracts have different functions.
  - **AaveDistributionManager**: This contract is an accounting contract for managing multiple staking distributions.
  - **StakedAaveV3**: The StakedTokenV3 is a staked token that uses the AAVE token.
  - **StakedTokenV2**: This contract allows the stakement of Aave tokens, tokenization of positions, and receiving rewards, inheriting from a distribution manager contract.
  - **GovernancePowerWithSnapshot**: This contract is ERC20 and includes snapshots of balances on transfer-related actions.
  - **RoleManager**: This contract is a generic role manager for managing slashing and cooldown admin in StakedAaveV3.
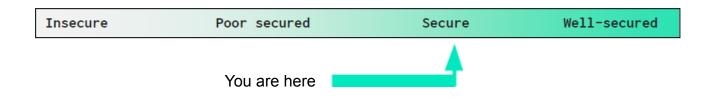
Audit scope

Name	Code Review and Security Analysis Report for Staked Aave Token Smart Contracts
Platform	Ethereum
File	StakedAaveV3.sol
Smart Contract Code	0x0fe58fe1caa69951dc924a8c222be19013b89476
Audit Date	April 24th, 2024

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1: StakedAaveV3.sol **Claim helper owner control:** ● Claim rewards and stake on behalf. **StakedTokenV3.sol** ● Emission managers can configure assets. ● Cooldown on Behalf Of address can be set by the claim helper owner. ● Redeem On Behalf address and amount can be set by the claim helper owner. ● Claim rewards on Behalf address and amount can be set by the claim helper owner. ● Claim Rewards And Redeem On Behalf address and amount can be set by the claim helper owner. ● Slash address ad amount set by the slashing admin. ● Settle slashing by the slashing admin. ● Maximum Slashable percentage can be set by the slashing admin.	**YES, This is valid.**

Audit Summary

According to the standard audit assessment, the Customer`s solidity-based smart contracts are **"Secured"**. Also, these contracts contain owner control, which does not make them fully decentralized.

Insecure	Poor secured	Secure	Well-secured

You are here →

We used various tools like Slither, Solhint, and Remix IDE. At the same time, this finding solidity-based critical analysis of the manual audit.

All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the Audit overview section, The General overview is presented in the AS-IS section and all identified issues can be found in the Audit overview section.

We found 0 critical, 0 high, 0 medium, 0 low, and 5 very low level issues.

Investors Advice: The Technic Overview of the general tract does not guarantee the ethical nature of the project. Any owner-controlled functions should be executed by the owner with responsibility. All stores,/users are advised to do their due diligence before investing in the project.

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.

Email: audit@EtherAuthority.io

Technical Quick Stats

Main Category	Subcategory	Result
Contract Programming	The solidity version is not specified	Passed
	The solidity version is too old	Moderated
	Integer overflow/underflow	Passed
	Function input parameters lack check	Passed
	Function input parameters check bypass	Passed
	Function access control lacks management	Passed
	Critical operation lacks event log	Passed
	Human/contract checks bypass	Passed
	Random number generation/use vulnerability	N/A
	Fallback function misuse	Passed
	Race condition	Passed
	Logical vulnerability	Passed
	Features claimed	Passed
	Other programming issues	Moderated
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Passed
	Unused code	Passed
Gas Optimization	"Out of Gas" Issue	Passed
	High consumption 'for/while' loop	Passed
	High consumption 'storage' storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage is not set	Passed
	"Short Address" Attack	Passed
	"Double Spend" Attack	Passed

Overall Audit Result: PASSED

Business Risk Analysis

Category	Result
🟢 Buy Tax	0%
🟢 Sell Tax	0%
🟢 Cannot Buy	No
🟢 Cannot Sell	No
🟢 Max Tax	0%
🟢 Modify Tax	Not Detected
🟢 Fee Check	No
🟢 Is Honeypot	Not Detected
🟢 Trading Cooldown	Detected
🟢 Can Pause Trade?	No
🟢 Pause Transfer?	Not Detected
🟢 Max Tax?	No
🟢 Is it Anti-whale?	Not Detected
🟢 Is Anti-bot?	Not Detected
🟢 Is it a Blacklist?	Not Detected
🟢 Blacklist Check	No
🟢 Can Mint?	No
🟢 Is it Proxy?	Yes
🟢 Can Take Ownership?	No
🟢 Hidden Owner?	No
🟢 Self Destruction?	Not Detected
🟢 Auditor Confidence	High

Overall Audit Result: PASSED

Code Quality

This audit scope has 1 smart contract file. Smart contracts contain Libraries, Smart contracts, inherits, and Interfaces. This is a compact and well-written smart contract.

The libraries in Staked Aave Token are part of its logical algorithm. A library is a different type of smart contract that contains reusable code. Once deployed on the blockchain (only once), it is assigned a specific address and its properties/methods can be reused many times by other contracts in the Staked Aave Token.

The EtherAuthority team has no scenario and unit test scripts, which would have helped to determine the integrity of the code in an automated way.

Code parts are not well commented on in the smart contracts. Ethereum's NatSpec commenting style is recommended.

Documentation

We were given a Staked Aave Token smart contract code in the form of an [Etherscan](#) web link.

As mentioned above, code parts are not well commented on. but the logic is straightforward. So it is easy to quickly understand the programming flow as well as complex code logic. Comments are very helpful in understanding the overall architecture of the protocol.

Use of Dependencies

As per our observation, the libraries used in this smart contract infrastructure that are based on well known industry standard open source projects.

Apart from libraries, its functions are not used in external smart contract calls.

AS-IS overview

StakedAaveV3.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Missing-zero-address-validation	Refer Audit Findings
2	REVISION	write	Passed	No Issue
3	initialize	external	initializer	No Issue
4	claimRewardsAndStake	external	Passed	No Issue
5	claimRewardsAndStakeOnBehalf	external	access only Claim Helper	No Issue
6	_afterTokenTransfer	internal	Passed	No Issue
7	_updateDiscountDistribution	internal	Passed	No Issue
8	onlySlashingAdmin	modifier	Passed	No Issue
9	onlyClaimHelper	modifier	Passed	No Issue
10	onlyCooldownAdmin	modifier	Passed	No Issue
11	REVISION	write	Passed	No Issue
12	getRevision	internal	Passed	No Issue
13	initialize	external	Passed	No Issue
14	_initialize	internal	Passed	No Issue
15	configureAssets	external	Passed	No Issue
16	previewStake	read	Passed	No Issue
17	stake	external	Passed	No Issue
18	stakeWithPermit	external	Passed	No Issue
19	cooldown	external	Passed	No Issue
20	cooldownOnBehalfOf	external	access only Claim Helper	No Issue
21	_cooldown	internal	Passed	No Issue
22	redeem	external	Passed	No Issue
23	redeemOnBehalf	external	access only Claim Helper	No Issue
24	claimRewards	external	Passed	No Issue
25	claimRewardsOnBehalf	external	access only Claim Helper	No Issue
26	claimRewardsAndRedeem	external	Passed	No Issue
27	claimRewardsAndRedeemOnBehalf	external	claim	No Issue
28	getExchangeRate	read	Passed	No Issue
29	previewRedeem	read	Passed	No Issue
30	slash	external	access only Slashing Admin	No Issue
31	returnFunds	external	Passed	No Issue
32	settleSlashing	external	access only Slashing Admin	No Issue

33	setMaxSlashablePercentage	external	access only Slashing Admin	No Issue
34	getMaxSlashablePercentage	external	Passed	No Issue
35	setCooldownSeconds	external	access only Cool down Admin	No Issue
36	getCooldownSeconds	external	Passed	No Issue
37	COOLDOWN_SECONDS	external	Function Overriding Issue	Refer Audit Findings
38	_setMaxSlashablePercentage	internal	Passed	No Issue
39	_setCooldownSeconds	internal	Passed	No Issue
40	_claimRewards	internal	Passed	No Issue
41	_claimRewardsAndStakeOnBehalf	internal	Passed	No Issue
42	_stake	internal	Passed	No Issue
43	_redeem	internal	Passed	No Issue
44	_updateExchangeRate	internal	Passed	No Issue
45	_getExchangeRate	internal	Passed	No Issue
46	_transfer	internal	Passed	No Issue
47	_afterTokenTransfer	internal	Passed	No Issue
48	_getDelegationState	internal	Passed	No Issue
49	_getBalance	internal	Passed	No Issue
50	getPowerCurrent	read	Passed	No Issue
51	_setDelegationState	internal	Passed	No Issue
52	_incrementNonces	internal	Passed	No Issue
53	_getDomainSeparator	internal	Passed	No Issue
54	DOMAIN_SEPARATOR	read	Passed	No Issue
55	EIP712_REVISION	external	Passed	No Issue
56	stake	external	Passed	No Issue
57	redeem	external	Passed	No Issue
58	cooldown	external	Passed	No Issue
59	claimRewards	external	Passed	No Issue
60	getTotalRewardsBalance	external	Passed	No Issue
61	permit	external	Passed	No Issue
62	_updateCurrentUnclaimedRewards	internal	Passed	No Issue
63	_getDomainSeparator	internal	Passed	No Issue
64	_getDelegationState	internal	Passed	No Issue
65	_getBalance	internal	Passed	No Issue
66	_incrementNonces	internal	Passed	No Issue
67	_setDelegationState	internal	Passed	No Issue
68	delegateByType	external	Passed	No Issue
69	delegate	external	Passed	No Issue
70	getDelegateeByType	external	Passed	No Issue
71	getDelegates	external	Passed	No Issue
72	getPowerCurrent	read	Passed	No Issue
73	getPowersCurrent	external	Passed	No Issue
74	metaDelegateByType	external	Passed	No Issue
75	metaDelegate	external	Passed	No Issue

76	_governancePowerTransferByType	internal	Passed	No Issue
77	delegationChangeOnTransfer	internal	Passed	No Issue
78	_getDelegatedPowerByType	internal	Passed	No Issue
79	_getDelegateeByType	internal	Passed	No Issue
80	_updateDelegateeByType	internal	Passed	No Issue
81	_updateDelegationModeByType	internal	Passed	No Issue
82	_delegateByType	internal	Passed	No Issue
83	onlyRoleAdmin	modifier	Passed	No Issue
84	onlyPendingRoleAdmin	modifier	Passed	No Issue
85	getAdmin	read	Passed	No Issue
86	getPendingAdmin	read	Passed	No Issue
87	setPendingAdmin	write	access only Role Admin	No Issue
88	claimRoleAdmin	external	access only Pending Role Admin	No Issue
89	_initAdmins	internal	Passed	No Issue
90	_mint	internal	Passed	No Issue
91	_burn	internal	Passed	No Issue
92	_configureAssets	internal	Passed	No Issue
93	_updateAssetStateInternal	internal	Passed	No Issue
94	_updateUserAssetInternal	internal	Passed	No Issue
95	_claimRewards	internal	Passed	No Issue
96	_getUnclaimedRewards	internal	Passed	No Issue
97	_getRewards	internal	Passed	No Issue
98	_getAssetIndex	internal	Passed	No Issue
99	getUserAssetData	read	Passed	No Issue

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to token loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to tokens lose
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have significant impact on execution
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can be ignored.

Audit Findings

Critical Severity

No Critical severity vulnerabilities were found.

High Severity

No high-severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

No low-severity vulnerabilities were found.

Very Low / Informational / Best practices:

(1) Use the latest solidity version: **StakedTokenV3.sol**

```
pragma solidity ^0.8.0;
```

Use the latest solidity version while contract deployment to prevent any compiler version-level bugs.

Resolution: Please use 0.8.21 which is the latest version.

(2) Missing-zero-address-validation: **AaveDistributionManager.sol**

```
constructor(address emissionManager, uint256 distributionDuration) {
    DISTRIBUTION_END = block.timestamp + distributionDuration;
    EMISSION_MANAGER = emissionManager;
}
```

Detects missing zero address validation. The owner can transfer ownership without specifying the new owner, so the owner may lose ownership of the contract.

Resolution: We suggest first Checking that the address is not zero.

(3) Explicit Visibility for State Variables Warning: **PercentageMath.sol**

```
uint256 constant PERCENTAGE_FACTOR = 1e4; //percentage plus two decimals
uint256 constant HALF_PERCENT = PERCENTAGE_FACTOR / 2;
```

The warning is related to the visibility of state variables in your Solidity code.

Resolution: We recommend updating the code to explicitly mark the visibility of state variables using the internal or public keyword, depending on the intended visibility.

(4) Warning: SPDX license identifier: **RoleManager.sol**

Warning: SPDX license identifier not provided in the source file.

Resolution: Add SPDX-License-Identifier.

(5) Function Overriding Issue: **StakedAaveV3.sol**

The derived contract lacks the necessary function override for "COOLDOWN_SECONDS," which is defined in two or more base classes with the same name and parameter types.

Resolution: To resolve this issue, add the "override" specifier to the function "COOLDOWN_SECONDS" in the derived contract to indicate the intended override and clarify the function's source.

Centralization

This smart contract has some functions which can be executed by the Admin (Owner) only. If the admin wallet's private key would be compromised, then it would create trouble. Following are Admin functions:

StakedAaveV3.sol

- claimRewardsAndStakeOnBehalf: Claim rewards and stake on behalf of the claim helper owner.

StakedTokenV3.sol

- configureAssets: The emissions manager can configure assets.
- cooldownOnBehalfOf: Cooldown on Behalf of the address can be set by the claim helper owner.
- redeemOnBehalf: The redeemOnBehalf address and amount can be set by the claim helper owner.
- claimRewardsOnBehalf: Claim rewards on Behalf address and amount can be set by the claim helper owner.
- claimRewardsAndRedeemOnBehalf: Claim Rewards And Redeem On Behalf address and amount can be set by the claim helper owner.
- slash: Slash address and the amount set by the slashing admin.
- settleSlashing: Settle slashing by the slashing admin.
- setMaxSlashablePercentage: The maximum slashable percentage can be set by the slashing admin.
- setCooldownSeconds: Cooldown seconds can be set by the cooldown admin.

RoleManager.sol

- setPendingAdmin: A new admin address can be set by the current admin.
- claimRoleAdmin: Claim role assigned by the Pending Role Admin.

To make the smart contract 100% decentralized, we suggest renouncing ownership of the smart contract once its function is completed.

Conclusion

We were given a contract code in the form of Etherscan web links. And we have used all possible tests based on given objects as files. We observed 5 informational issues in the smart contracts. And those issues are not critical. So, **it's good to go for the production**.

Since possible test cases can be unlimited for such smart contracts protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover the maximum possible test cases to scan everything.

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. Smart Contract's high-level description of functionality was presented in the As-is overview section of the report.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

Security state of the reviewed smart contract, based on standard audit procedure scope, is **"Secured".**

Our Methodology

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our security audits are to improve the quality of systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

Manual Code Review:

In manually reviewing all of the code, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

Vulnerability Analysis:

Our audit techniques included manual code analysis, user interface interaction, and white box penetration testing. We look at the project's website to get a high-level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. While we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation.

Documenting Results:

We follow a conservative, transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even though we have not yet verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, and then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this we analyze the feasibility of an attack in a live system.

Suggested Solutions:

We search for immediate mitigations that live deployments can take, and finally we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinized by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our report, and before the details are made public.

Disclaimers

EtherAuthority.io Disclaimer

EtherAuthority team has analyzed this smart contract in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment and functionality (performing the intended functions).

Due to the fact that the total number of test cases is unlimited, the audit makes no statements or warranties on the security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bug-free status, or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only. We also suggest conducting a bug bounty program to confirm the high level of security of this smart contract.

Technical Disclaimer

Smart contracts are deployed and executed on the blockchain platform. The platform, its programming language, and other software related to the smart contract can have their own vulnerabilities that can lead to hacks. Thus, the audit can't guarantee explicit security of the audited smart contracts.

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.

Email: audit@EtherAuthority.io

Appendix

Code Flow Diagram - Staked Aave Token

StakedAaveV3 Diagram

Slither Results Log

Slither is a Solidity static analysis framework that uses vulnerability detectors, displays contract details, and provides an API for writing custom analyses. It helps developers identify vulnerabilities, improve code comprehension, and prototype custom analyses quickly. The analysis includes a report with warnings and errors, allowing developers to quickly prototype and fix issues.

We did the analysis of the project altogether. Below are the results.

Slither Log >> StakedAaveV3.sol

```
StakedTokenV3.constructor(IERC20,IERC20,uint256,address,address,uint128).decimals (StakedAaveV3.sol#1825) shadows:
        - ERC20.decimals() (StakedAaveV3.sol#1076-1078) (function)
        - IERC20Metadata.decimals() (StakedAaveV3.sol#34) (function)
StakedTokenV3.previewStake(uint256).assets (StakedAaveV3.sol#1884) shadows:
        - AaveDistributionManager.assets (StakedAaveV3.sol#829) (state variable)
StakedTokenV3.returnFunds(uint256).assets (StakedAaveV3.sol#2007) shadows:
        - AaveDistributionManager.assets (StakedAaveV3.sol#829) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

Reentrancy in StakedTokenV3._claimRewards(address,address,uint256) (StakedAaveV3.sol#2066-2087):
        External calls:
        - REWARD_TOKEN.safeTransferFrom(REWARDS_VAULT,to,amountToClaim) (StakedAaveV3.sol#2084)
        Event emitted after the call(s):
        - RewardsClaimed(from,to,amountToClaim) (StakedAaveV3.sol#2085)
Reentrancy in StakedTokenV3._claimRewardsAndStakeOnBehalf(address,address,uint256) (StakedAaveV3.sol#2089-2111):
        External calls:
        - _claimRewards(from,address(this),amountToClaim) (StakedAaveV3.sol#2106)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (StakedAaveV3.sol#788-791)
                - (success,returndata) = target.call{value: value}(data) (StakedAaveV3.sol#625)
                - REWARD_TOKEN.safeTransferFrom(REWARDS_VAULT,to,amountToClaim) (StakedAaveV3.sol#2084)
        - _stake(address(this),to,amountToClaim) (StakedAaveV3.sol#2107)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (StakedAaveV3.sol#788-791)
                - (success,returndata) = target.call{value: value}(data) (StakedAaveV3.sol#625)
                - STAKED_TOKEN.safeTransferFrom(from,address(this),amount) (StakedAaveV3.sol#2137)
        External calls sending eth:
        - _claimRewards(from,address(this),amountToClaim) (StakedAaveV3.sol#2106)
                - (success,returndata) = target.call{value: value}(data) (StakedAaveV3.sol#625)
        - _stake(address(this),to,amountToClaim) (StakedAaveV3.sol#2107)
                - (success,returndata) = target.call{value: value}(data) (StakedAaveV3.sol#625)
        Event emitted after the call(s):
        - AssetIndexUpdated(underlyingAsset,newIndex) (StakedAaveV3.sol#888)
                - _stake(address(this),to,amountToClaim) (StakedAaveV3.sol#2107)
        - RewardsAccrued(to,accruedRewards) (StakedAaveV3.sol#2132)
                - _stake(address(this),to,amountToClaim) (StakedAaveV3.sol#2107)
        - Staked(from,to,amount,sharesToMint) (StakedAaveV3.sol#2141)
                - _stake(address(this),to,amountToClaim) (StakedAaveV3.sol#2107)
```

```
        - Staked(from,to,amount,sharesToMint) (StakedAaveV3.sol#2141)
                - _stake(address(this),to,amountToClaim) (StakedAaveV3.sol#2107)
        - Transfer(address(0),account,amount) (StakedAaveV3.sol#1163)
                - _stake(address(this),to,amountToClaim) (StakedAaveV3.sol#2107)
        - UserIndexUpdated(user,asset,newIndex) (StakedAaveV3.sol#914)
                - _stake(address(this),to,amountToClaim) (StakedAaveV3.sol#2107)
Reentrancy in StakedTokenV3._redeem(address,address,uint256) (StakedAaveV3.sol#2144-2191):
        External calls:
        - IERC20(STAKED_TOKEN).safeTransfer(to,underlyingToRedeem) (StakedAaveV3.sol#2188)
        Event emitted after the call(s):
        - Redeem(from,to,underlyingToRedeem,amountToRedeem) (StakedAaveV3.sol#2190)
Reentrancy in StakedTokenV3._stake(address,address,uint256) (StakedAaveV3.sol#2113-2142):
        External calls:
        - STAKED_TOKEN.safeTransferFrom(from,address(this),amount) (StakedAaveV3.sol#2137)
        Event emitted after the call(s):
        - Staked(from,to,amount,sharesToMint) (StakedAaveV3.sol#2141)
        - Transfer(address(0),account,amount) (StakedAaveV3.sol#1163)
                - _mint(to,sharesToMint) (StakedAaveV3.sol#2139)
Reentrancy in StakedTokenV3.claimRewardsAndRedeem(address,uint256,uint256) (StakedAaveV3.sol#1944-1951):
        External calls:
        - _claimRewards(msg.sender,to,claimAmount) (StakedAaveV3.sol#1949)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (StakedAaveV3.sol#788-791)
                - (success,returndata) = target.call{value: value}(data) (StakedAaveV3.sol#625)
                - REWARD_TOKEN.safeTransferFrom(REWARDS_VAULT,to,amountToClaim) (StakedAaveV3.sol#2084)
        - _redeem(msg.sender,to,redeemAmount) (StakedAaveV3.sol#1950)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (StakedAaveV3.sol#788-791)
                - (success,returndata) = target.call{value: value}(data) (StakedAaveV3.sol#625)
                - IERC20(STAKED_TOKEN).safeTransfer(to,underlyingToRedeem) (StakedAaveV3.sol#2188)
        External calls sending eth:
        - _claimRewards(msg.sender,to,claimAmount) (StakedAaveV3.sol#1949)
                - (success,returndata) = target.call{value: value}(data) (StakedAaveV3.sol#625)
        - _redeem(msg.sender,to,redeemAmount) (StakedAaveV3.sol#1950)
                - (success,returndata) = target.call{value: value}(data) (StakedAaveV3.sol#625)
        Event emitted after the call(s):
        - AssetIndexUpdated(underlyingAsset,newIndex) (StakedAaveV3.sol#888)
                - _redeem(msg.sender,to,redeemAmount) (StakedAaveV3.sol#1950)
        - Redeem(from,to,underlyingToRedeem,amountToRedeem) (StakedAaveV3.sol#2190)
```

```
        - RewardsAccrued(user,accruedRewards) (StakedAaveV3.sol#1731)
                - _redeem(from,to,redeemAmount) (StakedAaveV3.sol#1960)
        - Transfer(account,address(0),amount) (StakedAaveV3.sol#1180)
                - _redeem(from,to,redeemAmount) (StakedAaveV3.sol#1960)
        - UserIndexUpdated(user,asset,newIndex) (StakedAaveV3.sol#914)
                - _redeem(from,to,redeemAmount) (StakedAaveV3.sol#1960)
Reentrancy in StakedTokenV3.returnFunds(uint256) (StakedAaveV3.sol#2003-2012):
        External calls:
        - STAKED_TOKEN.safeTransferFrom(msg.sender,address(this),amount) (StakedAaveV3.sol#2010)
        Event emitted after the call(s):
        - FundsReturned(amount) (StakedAaveV3.sol#2011)
Reentrancy in StakedTokenV3.slash(address,uint256) (StakedAaveV3.sol#1976-2001):
        External calls:
        - STAKED_TOKEN.safeTransfer(destination,amount) (StakedAaveV3.sol#1997)
        Event emitted after the call(s):
        - Slashed(destination,amount) (StakedAaveV3.sol#1999)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```

```
AaveDistributionManager._updateAssetStateInternal(address,AaveDistributionManager.AssetData,uint256) (StakedAaveV3.sol#867-894)
 uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp == lastUpdateTimestamp (StakedAaveV3.sol#875)
        - newIndex != oldIndex (StakedAaveV3.sol#886)
AaveDistributionManager._updateUserAssetInternal(address,address,uint256,uint256) (StakedAaveV3.sol#896-918) uses timestamp for
 comparisons
        Dangerous comparisons:
        - userIndex != newIndex (StakedAaveV3.sol#908)
AaveDistributionManager._getAssetIndex(uint256,uint256,uint128,uint256) (StakedAaveV3.sol#976-998) uses timestamp for compariso
ns
        Dangerous comparisons:
        - emissionPerSecond == 0 || totalBalance == 0 || lastUpdateTimestamp == block.timestamp || lastUpdateTimestamp >= DISTR
IBUTION_END (StakedAaveV3.sol#983-986)
        - block.timestamp > DISTRIBUTION_END (StakedAaveV3.sol#991-993)
StakedTokenV2.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (StakedAaveV3.sol#1628-1662) uses timestamp for com
parisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp <= deadline,INVALID_EXPIRATION) (StakedAaveV3.sol#1638)
```

```
StakedTokenV3._stake(address,address,uint256) (StakedAaveV3.sol#2113-2142) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(amount != 0,INVALID_ZERO_AMOUNT) (StakedAaveV3.sol#2119)
        - accruedRewards != 0 (StakedAaveV3.sol#2130)
StakedTokenV3._redeem(address,address,uint256) (StakedAaveV3.sol#2144-2191) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)((block.timestamp > cooldownSnapshot.timestamp + _cooldownSeconds),INSUFFICIENT_COOLDOWN) (Staked
AaveV3.sol#2153-2156)
        - require(bool,string)((block.timestamp - (cooldownSnapshot.timestamp + _cooldownSeconds) <= UNSTAKE_WINDOW),UNSTAKE_WI
NDOW_FINISHED) (StakedAaveV3.sol#2157-2161)
        - require(bool,string)(maxRedeemable != 0,INVALID_ZERO_MAX_REDEEMABLE) (StakedAaveV3.sol#2168)
        - cooldownSnapshot.timestamp != 0 (StakedAaveV3.sol#2178)
        - cooldownSnapshot.amount - amountToRedeem == 0 (StakedAaveV3.sol#2179)
        - (amount > maxRedeemable) (StakedAaveV3.sol#2170)
StakedTokenV3._updateExchangeRate(uint216) (StakedAaveV3.sol#2193-2197) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(newExchangeRate != 0,ZERO_EXCHANGE_RATE) (StakedAaveV3.sol#2194)
StakedTokenV3._transfer(address,address,uint256) (StakedAaveV3.sol#2209-2232) uses timestamp for comparisons
        Dangerous comparisons:
        - balanceOfFrom == amount (StakedAaveV3.sol#2223)
        - balanceOfFrom - amount < previousSenderCooldown.amount (StakedAaveV3.sol#2225)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
```

```
Pragma version^0.8.0 (StakedAaveV3.sol#2) allows old versions
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (StakedAaveV3.sol#591-596):
        - (success) = recipient.call{value: amount}() (StakedAaveV3.sol#594)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (StakedAaveV3.sol#618-627):
        - (success,returndata) = target.call{value: value}(data) (StakedAaveV3.sol#625)
Low level call in Address.functionStaticCall(address,bytes,string) (StakedAaveV3.sol#633-640):
        - (success,returndata) = target.staticcall(data) (StakedAaveV3.sol#638)
Low level call in Address.functionDelegateCall(address,bytes,string) (StakedAaveV3.sol#646-653):
        - (success,returndata) = target.delegatecall(data) (StakedAaveV3.sol#651)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

```
        - GovernancePowerDelegationERC20.getPowerCurrent(address,IGovernancePowerDelegationToken.DelegationType) (StakedAaveV3.
sol#1340-1353)
        - StakedTokenV3.getRevision() (StakedAaveV3.sol#1833-1835)
        - StakedTokenV2.getTotalRewardsBalance(address) (StakedAaveV3.sol#1611-1626)
        - AaveDistributionManager.getUserAssetData(address,address) (StakedAaveV3.sol#1000-1006)
        - ERC20.increaseAllowance(address,uint256) (StakedAaveV3.sol#1115-1119)
        - ERC20.name() (StakedAaveV3.sol#1068-1070)
        - StakedTokenV2.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (StakedAaveV3.sol#1628-1662)
        - StakedTokenV3.previewRedeem(uint256) (StakedAaveV3.sol#1967-1974)
        - StakedTokenV3.previewStake(uint256) (StakedAaveV3.sol#1884-1886)
        - StakedTokenV3.redeem(address,uint256) (StakedAaveV3.sol#1914-1919)
        - StakedTokenV3.redeemOnBehalf(address,address,uint256) (StakedAaveV3.sol#1921-1927)
        - StakedTokenV3.returnFunds(uint256) (StakedAaveV3.sol#2003-2012)
        - StakedTokenV3.setCooldownSeconds(uint256) (StakedAaveV3.sol#2036-2041)
        - StakedTokenV3.setMaxSlashablePercentage(uint256) (StakedAaveV3.sol#2019-2025)
        - RoleManager.setPendingAdmin(uint256,address) (StakedAaveV3.sol#1258-1264)
        - StakedTokenV3.settleSlashing() (StakedAaveV3.sol#2014-2017)
        - StakedTokenV3.slash(address,uint256) (StakedAaveV3.sol#1976-2001)
        - StakedTokenV3.stake(address,uint256) (StakedAaveV3.sol#1888-1893)
        - ERC20.symbol() (StakedAaveV3.sol#1072-1074)
        - ERC20.totalSupply() (StakedAaveV3.sol#1080-1082)
        - GovernancePowerDelegationERC20.totalSupplyAt(uint256) (StakedAaveV3.sol#1369-1371)
        - ERC20.transfer(address,uint256) (StakedAaveV3.sol#1088-1092)
        - ERC20.transferFrom(address,address,uint256) (StakedAaveV3.sol#1104-1113)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions
```

```
PercentageMath.HALF_PERCENT (StakedAaveV3.sol#51) is never used in PercentageMath (StakedAaveV3.sol#49-84)
ERC20._balances (StakedAaveV3.sol#1057) is never used in StakedAaveV3 (StakedAaveV3.sol#2235-2448)
ERC20._totalSupply (StakedAaveV3.sol#1061) is never used in StakedAaveV3 (StakedAaveV3.sol#2235-2448)
StakedTokenV2._propositionPowerSnapshots (StakedAaveV3.sol#1571-1572) is never used in StakedAaveV3 (StakedAaveV3.sol#2235-2448
)
StakedTokenV2._propositionPowerSnapshotsCounts (StakedAaveV3.sol#1573) is never used in StakedAaveV3 (StakedAaveV3.sol#2235-244
8)
StakedTokenV2.EIP712_DOMAIN (StakedAaveV3.sol#1578-1581) is never used in StakedAaveV3 (StakedAaveV3.sol#2235-2448)
StakedTokenV3._cooldownSeconds (StakedAaveV3.sol#1778) is never used in StakedAaveV3 (StakedAaveV3.sol#2235-2448)
StakedTokenV3._maxSlashablePercentage (StakedAaveV3.sol#1779) is never used in StakedAaveV3 (StakedAaveV3.sol#2235-2448)
StakedTokenV3._currentExchangeRate (StakedAaveV3.sol#1780) is never used in StakedAaveV3 (StakedAaveV3.sol#2235-2448)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

ERC20._totalSupply (StakedAaveV3.sol#1061) should be constant
GovernancePowerWithSnapshot._aaveGovernance (StakedAaveV3.sol#1546) should be constant
StakedAaveV3._exchangeRateSnapshotsCount (StakedAaveV3.sol#2238) should be constant
StakedTokenV2.DOMAIN_SEPARATOR (StakedAaveV3.sol#1576) should be constant
StakedTokenV3._cooldownSeconds (StakedAaveV3.sol#1778) should be constant
StakedTokenV3._currentExchangeRate (StakedAaveV3.sol#1780) should be constant
StakedTokenV3._maxSlashablePercentage (StakedAaveV3.sol#1779) should be constant
StakedTokenV3.inPostSlashingPeriod (StakedAaveV3.sol#1781) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
StakedAaveV3.sol analyzed (25 contracts with 84 detectors), 211 result(s) found
```

Solidity Static Analysis

Static code analysis is used to identify many common coding problems before a program is released. It involves examining the code manually or using tools to automate the process. Static code analysis tools can automatically scan the code without executing it.

StakedAaveV3.sol

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in StakedAaveV3._beforeTokenTransfer(address,address,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 2341:2:

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.

more

Pos: 685:12:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.

more

Pos: 2158:9:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.

more

Pos: 847:4:

Constant/View/Pure functions:

StakedTokenV2.claimRewards(address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 1609:2:

Delete from dynamic array:

Using "delete" on an array leaves a gap. The length of the array remains the same. If you want to remove the empty position you need to shift items manually and update the "length" property.

more

Pos: 2224:10:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 2168:4:

Solhint Linter

Linters are the utility tools that analyze the given source code and report programming errors, bugs, and stylistic errors. For the Solidity language, there are some linter tools available that a developer can use to improve the quality of their Solidity contracts.

StakedAaveV3.sol

```
Compiler version ^0.8.0 does not satisfy the ^0.5.8 semver
requirement
Pos: 1:1
Use double quotes for string literals
Pos: 42:1237
Use double quotes for string literals
Pos: 7:1244
Use double quotes for string literals
Pos: 9:1276
Use double quotes for string literals
Pos: 7:1291
Use double quotes for string literals
Pos: 5:1307
Use double quotes for string literals
Pos: 5:1311
Use double quotes for string literals
Pos: 38:1377
Use double quotes for string literals
Pos: 42:1454
Use double quotes for string literals
Pos: 49:1576
Use double quotes for string literals
Pos: 7:1579
Use double quotes for string literals
Pos: 7:1583
Use double quotes for string literals
Pos: 34:1636
Use double quotes for string literals
Pos: 42:1637
Use double quotes for string literals
Pos: 9:1641
Use double quotes for string literals
Pos: 50:1656
Use double quotes for string literals
Pos: 24:1682
Use double quotes for string literals
Pos: 38:1685
Use double quotes for string literals
Pos: 44:1686
Use double quotes for string literals
Pos: 40:1687
Use double quotes for string literals
```

```
Pos: 24:1703
Use double quotes for string literals
Pos: 38:1706
Use double quotes for string literals
Pos: 44:1707
Use double quotes for string literals
Pos: 40:1708
Use double quotes for string literals
Pos: 7:1785
Use double quotes for string literals
Pos: 7:1793
Use double quotes for string literals
Pos: 7:1801
Use double quotes for string literals
Pos: 45:1874
Use double quotes for string literals
Pos: 26:1904
Use double quotes for string literals
Pos: 36:1981
Use double quotes for string literals
Pos: 25:1982
Use double quotes for string literals
Pos: 46:1991
Use double quotes for string literals
Pos: 36:2003
Use double quotes for string literals
Pos: 43:2005
Use double quotes for string literals
Pos: 7:2053
Use double quotes for string literals
Pos: 26:2070
Use double quotes for string literals
Pos: 33:2080
Use double quotes for string literals
Pos: 43:2093
Use double quotes for string literals
Pos: 36:2117
Use double quotes for string literals
Pos: 26:2118
Use double quotes for string literals
Pos: 26:2148
Use double quotes for string literals
Pos: 9:2154
Use double quotes for string literals
Pos: 9:2159
Use double quotes for string literals
Pos: 33:2167
Use double quotes for string literals
Pos: 35:2193
```

Software analysis result:

This software reported many false positive results and some were informational issues. So, those issues can be safely ignored.