// Private Smart Contract Security Assessment 06.24.2024 - 07.03.2024

Staking-Module

Bonzo

HALBORN

Staking-Module - Bonzo

Prepared by: HALBORN

Last Updated 07/17/2024

Date of Engagement by: June 24th, 2024 - July 3rd, 2024

Summary

100% ⓘ OF ALL REPORTED FINDINGS HAVE BEEN ADDRESSED

ALL FINDINGS	CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
10	2	0	1	2	5

TABLE OF CONTENTS

1. Introduction

`Bonzo` *engaged* `Halborn` *to conduct a security assessment on their smart contracts beginning on June 24th and ending on July 3rd. The security assessment was scoped to the smart contracts provided in the GitHub repository* *Bonzo Staking Module*, *commit hashes and further details can be found in the Scope section of this report.*

The protocol utilizes Aave's Safety Module for staking and rewards distribution in WHBAR on the Hedera network. Stakers of any permitted asset by the emission manager can stake their assets in the Safety Module. The protocol ensures that stakers earn rewards in WHBAR, Hedera's wrapped HBAR token. The Safety Module acts as a backstop in case of a shortfall event, enhancing the protocol's security. This system is designed to incentivize users to contribute to the protocol's safety while earning rewards for their participation.

2. Assessment Summary

The team at Halborn was provided one week for the engagement and assigned one full-time security engineer to check the security of the smart contract. The security engineer is a blockchain and smart-contract security expert with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The purpose of this assessment is to:

- Ensure that smart contract functions operate as intended.
- Identify potential security issues with the smart contracts.

In summary, Halborn identified some security risks that were mostly addressed by the `Bonzo team`.

3. Test Approach And Methodology

`Halborn` performed a combination of manual review of the code and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of the smart contract assessment. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of smart contracts and can quickly identify items that do not follow security best practices.

The following phases and associated tools were used throughout the term of the assessment:

- Research into the architecture, purpose, and use of the platform.
- Smart contract manual code review and walkthrough to identify any logic issues.
- A thorough assessment of safety and usage of critical Solidity variables and functions in scope that could lead to arithmetic related vulnerabilities.
- Manual testing by custom scripts.
- Graphing out functionality and contract logic/connectivity/functions (`solgraph`).
- Static Analysis of security for scoped contract, and imported functions. (`Slither`).
- Local or public testnet deployment (`Brownie, Remix IDE`).

4. RISK METHODOLOGY

Every vulnerability and issue observed by Halborn is ranked based on **two sets** of **Metrics** and a **Severity Coefficient**. This system is inspired by the industry standard Common Vulnerability Scoring System.

The two **Metric sets** are: **Exploitability** and **Impact**. **Exploitability** captures the ease and technical means by which vulnerabilities can be exploited and **Impact** describes the consequences of a successful exploit.

The **Severity Coefficients** is designed to further refine the accuracy of the ranking with two factors: **Reversibility** and **Scope**. These capture the impact of the vulnerability on the environment as well as the number of users and smart contracts affected.

The final score is a value between 0-10 rounded up to 1 decimal place and 10 corresponding to the highest security risk. This provides an objective and accurate rating of the severity of security vulnerabilities in smart contracts.

The system is designed to assist in identifying and prioritizing vulnerabilities based on their level of risk to address the most critical issues in a timely manner.

4.1 EXPLOITABILITY

ATTACK ORIGIN (AO):

Captures whether the attack requires compromising a specific account.

ATTACK COST (AC):

Captures the cost of exploiting the vulnerability incurred by the attacker relative to sending a single transaction on the relevant blockchain. Includes but is not limited to financial and computational cost.

ATTACK COMPLEXITY (AX):

Describes the conditions beyond the attacker's control that must exist in order to exploit the vulnerability. Includes but is not limited to macro situation, available third-party liquidity and regulatory challenges.

METRICS:

EXPLOITABILIY METRIC (M_E)	METRIC VALUE	NUMERICAL VALUE
Attack Origin (AO)	Arbitrary (AO:A) Specific (AO:S)	1 0.2
Attack Cost (AC)	Low (AC:L) Medium (AC:M) High (AC:H)	1 0.67 0.33

EXPLOITABILIY METRIC (M_E)	METRIC VALUE	NUMERICAL VALUE
Attack Complexity (AX)	Low (AX:L) Medium (AX:M) High (AX:H)	1 0.67 0.33

Exploitability E is calculated using the following formula:

$$E = \prod m_e$$

4.2 IMPACT

CONFIDENTIALITY (C):

Measures the impact to the confidentiality of the information resources managed by the contract due to a successfully exploited vulnerability. Confidentiality refers to limiting access to authorized users only.

INTEGRITY (I):

Measures the impact to integrity of a successfully exploited vulnerability. Integrity refers to the trustworthiness and veracity of data stored and/or processed on-chain. Integrity impact directly affecting Deposit or Yield records is excluded.

AVAILABILITY (A):

Measures the impact to the availability of the impacted component resulting from a successfully exploited vulnerability. This metric refers to smart contract features and functionality, not state. Availability impact directly affecting Deposit or Yield is excluded.

DEPOSIT (D):

Measures the impact to the deposits made to the contract by either users or owners.

YIELD (Y):

Measures the impact to the yield generated by the contract for either users or owners.

METRICS:

IMPACT METRIC (M_I)	METRIC VALUE	NUMERICAL VALUE
Confidentiality (C)	None (I:N) Low (I:L) Medium (I:M) High (I:H) Critical (I:C)	0 0.25 0.5 0.75 1

IMPACT METRIC (M_I)	METRIC VALUE	NUMERICAL VALUE
Integrity (I)	None (I:N) Low (I:L) Medium (I:M) High (I:H) Critical (I:C)	0 0.25 0.5 0.75 1
Availability (A)	None (A:N) Low (A:L) Medium (A:M) High (A:H) Critical (A:C)	0 0.25 0.5 0.75 1
Deposit (D)	None (D:N) Low (D:L) Medium (D:M) High (D:H) Critical (D:C)	0 0.25 0.5 0.75 1
Yield (Y)	None (Y:N) Low (Y:L) Medium (Y:M) High (Y:H) Critical (Y:C)	0 0.25 0.5 0.75 1

Impact I is calculated using the following formula:

$$I = max(m_I) + \frac{\sum m_I - max(m_I)}{4}$$

4.3 SEVERITY COEFFICIENT

REVERSIBILITY (R):

Describes the share of the exploited vulnerability effects that can be reversed. For upgradeable contracts, assume the contract private key is available.

SCOPE (S):

Captures whether a vulnerability in one vulnerable contract impacts resources in other contracts.

METRICS:

SEVERITY COEFFICIENT (C)	COEFFICIENT VALUE	NUMERICAL VALUE
Reversibility (r)	None (R:N) Partial (R:P) Full (R:F)	1 0.5 0.25
Scope (s)	Changed (S:C) Unchanged (S:U)	1.25 1

Severity Coefficient C is obtained by the following product:

$$C = rs$$

The Vulnerability Severity Score S is obtained by:

$$S = min(10, EIC * 10)$$

The score is rounded up to 1 decimal places.

SEVERITY	SCORE VALUE RANGE
Critical	9 - 10
High	7 - 8.9
Medium	4.5 - 6.9
Low	2 - 4.4
Informational	0 - 1.9

5. SCOPE

FILES AND REPOSITORY

(a) Repository: bonzo-staking-module

(b) Assessed Commit ID: 0c3d22e

(c) Items in scope:

- contracts/interfaces/IAaveIncentivesController.sol
- contracts/interfaces/IExchangeRate.sol
- contracts/interfaces/IHederaTokenService.sol
- contracts/interfaces/IWHBAR.sol
- contracts/lib/DistributionTypes.sol
- contracts/misc/WHBAR/Bits.sol
- contracts/misc/WHBAR/HederaResponseCodes.sol
- contracts/misc/WHBAR/HederaTokenService.sol
- contracts/misc/WHBAR/SafeCast.sol
- contracts/misc/WHBAR/SafeHederaTokenService.sol
- contracts/misc/WHBAR/WHBARContract.sol
- contracts/stake/AaveDistributionManager.sol
- contracts/stake/AaveIncentivesController.sol
- contracts/stake/StakedAave.sol
- contracts/stake/StakedToken.sol

Out-of-Scope: External libraries and financial-related attacks., New features/implementations after/with the remediation commit IDs.

REMEDIATION COMMIT ID:

- `1ecf5991ecf599`
- `b1b2698b1b2698`
- `155419e155419e`
- `f3e259bf3e259b`
- `5d970165d97016`

Out-of-Scope: New features/implementations after the remediation commit IDs.

6. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
2	0	1	2	5

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
MISCALCULATED PERIOD DISRUPT EMISSION SCHEDULE EXECUTION	CRITICAL	SOLVED - 07/11/2024
WRONG REWARD DISTRIBUTION LEADS TO DOUBLE SPEND	CRITICAL	SOLVED - 07/11/2024
INVALID EMISSIONSCHEDULE LENGTH CHECK	MEDIUM	SOLVED - 07/14/2024
MISSING DUPLICATE CALLER ADDRESSES CHECK	LOW	SOLVED - 07/11/2024
HARDCODED TESTNET EVM ADDRESS INSTEAD OF MAINNET	LOW	SOLVED - 07/11/2024
INCREASE PERIOD_DURATION TO 30 DAYS	INFORMATIONAL	SOLVED - 07/11/2024
INCONSISTENT STRUCT PARAMETER UPDATE	INFORMATIONAL	SOLVED - 07/11/2024
USAGE OF OUTDATED SAFECAST	INFORMATIONAL	ACKNOWLEDGED
USE OF MEMORY INSTEAD OF CALLDATA IN FUNCTION VARIABLE	INFORMATIONAL	ACKNOWLEDGED

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
REDUNDANT PARAMETERS IN FUNCTION _GETASSETINDEX	INFORMATIONAL	SOLVED - 07/11/2024

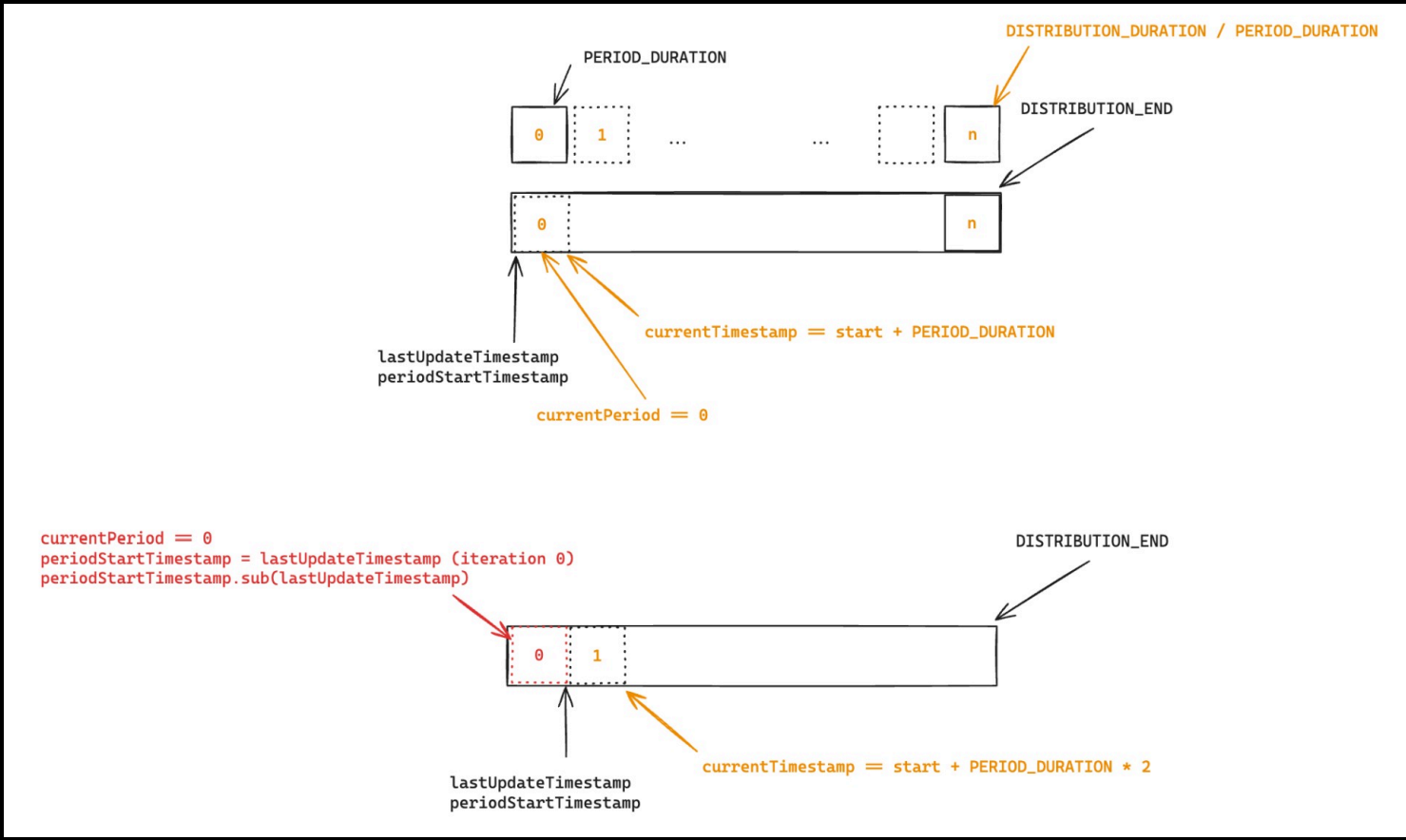
7. FINDINGS & TECH DETAILS

7.1 MISCALCULATED PERIOD DISRUPT EMISSION SCHEDULE EXECUTION

// CRITICAL

Description

The current implementation of the `_getAssetIndex` function in the `AaveDistributionManager` contract incorrectly calculates the `currentPeriod` using `periodStartTimestamp.sub(lastUpdateTimestamp)`. Initially, during the first iteration, both `periodStartTimestamp` and `lastUpdateTimestamp` are the same, making the difference zero, which works correctly. However, this approach fails when `lastUpdateTimestamp` is higher or equal to the timestamp of period 1 (index 0). This leads to incorrect period calculations and potential out-of-bounds errors when accessing the emission schedule.
The issue arises because the code does not correctly handle cases where the `lastUpdateTimestamp` is already in a later period, as shown in the provided diagrams. This flaw was not evident during initial setup testing when `lastUpdateTimestamp` was set to the current timestamp, corresponding to index 0.



BVSS

AO:A/AC:L/AX:L/C:N/I:C/A:H/D:H/Y:M/R:N/S:C (10.0)

Recommendation

To resolve this critical issue, the `currentPeriod` should be calculated using the formula `DISTRIBUTION_DURATION - (DISTRIBUTION_END - periodStartTimestamp) / PERIOD_DURATION` or

by maintaining a mechanism to keep track of the last distributed period and calculating the elapsed time difference with the current timestamp.
The new formula needs to be thoroughly tested to ensure it handles all edge cases correctly. This testing should include scenarios where updates are made in periods beyond the initial setup and various edge cases such as period transitions and partial periods.

Remediation Plan

SOLVED: The **Bonzo team** solved this issue as recommended.

Remediation Hash

`1ecf5995b8ed3b7fa84441a7be114b4dbdf551bc`

7.2 WRONG REWARD DISTRIBUTION LEADS TO DOUBLE SPEND

// CRITICAL

Description

The `AaveIncentivesController` contract has a vulnerability that results in double spending of rewards. The issue occurs in the `claimRewards` function, where the contract transfers rewards both as `REWARD_TOKEN` and as `IWHBAR`. Specifically, the contract transfers the reward amount to the recipient twice, once via `REWARD_TOKEN.transferFrom(REWARDS_VAULT, to, amountToClaim)` and again via `IWHBAR(_whbarContract).withdraw(REWARDS_VAULT, to, amountToClaim)`.
This leads to the recipient receiving the reward amount twice, causing financial loss to the rewards vault and effectively doubling the rewards for the claimant.

BVSS

AO:A/AC:L/AX:L/C:N/I:N/A:C/D:C/Y:C/R:P/S:C (9.4)

Recommendation

To fix this issue, the rewards should first be transferred to the contract's address and then transferred to the recipient. Additionally, proper approvals should be set up for the `IWHBAR` contract to transfer the reward amounts.
By implementing this change, the rewards will be correctly transferred to the recipient only once, and the risk of double spending will be mitigated. Additionally, ensure that the `REWARD_TOKEN` has been approved for spending by the `IWHBAR` contract to facilitate the withdrawal. This approach prevents financial loss and maintains the integrity of the rewards' distribution.

Remediation Plan

SOLVED: The **Bonzo team** solved this issue as recommended. The `REWARDS_VAULT` will require to approve `AaveIncentivesController` to spend the `_whbarContract` tokens.

Remediation Hash

b1b269852a55fd7a2456b5d802054ba61ddffd8f

References

AaveIncentivesController.sol#169-170

7.3 INVALID EMISSIONSCHEDULE LENGTH CHECK

// MEDIUM

Description

The `AaveDistributionManager` contract allows the direct call to the `configureAssets` function without validating the length of the `emissionSchedule`. This can lead to an out-of-bounds error when fetching the asset index if an invalid `emissionSchedule` length is provided. The initialization functions in `StakedToken` and `AaveIncentivesController` correctly perform this check, but it should also be enforced in the `configureAssets` function of `AaveDistributionManager` to prevent potential vulnerabilities from improper usage.

BVSS

AO:A/AC:L/AX:L/C:N/I:H/A:N/D:N/Y:N/R:P/S:C (4.7)

Recommendation

Add the following check in the `configureAssets` function of the `AaveDistributionManager` contract to ensure the validity of the `emissionSchedule` length:

```
require(
  emissionSchedule.length * PERIOD_DURATION == DISTRIBUTION_DURATION,
  'Invalid emission schedule length'
);
```

This will ensure that the emission schedule length is always valid, thereby preventing out-of-bounds errors when accessing the asset index.

Remediation Plan

SOLVED: The **Bonzo team** solved this issue as recommended.

Remediation Hash

155419ee299f2162c61e064bb422327516d568c6

7.4 MISSING DUPLICATE CALLER ADDRESSES CHECK

// LOW

Description

The `AaveIncentivesController` contract lacks a check in the `addCallerAssets` function to prevent the addition of duplicate caller addresses. The current implementation uses a list to store whitelisted callers for the `onlyCallerAsset` modifier, which is not optimized for checking or preventing duplicates. This can lead to inefficiencies and potential issues with duplicate entries. The use of `EnumerableSet` from OpenZeppelin or a two-struct approach (a mapping for O(1) access and a list for enumeration) would optimize the code and prevent duplicates effectively.

BVSS

AO:S/AC:L/AX:L/C:N/I:C/A:L/D:N/Y:N/R:N/S:C (2.7)

Recommendation

Optimize the `addCallerAssets` function by using `EnumerableSet` from OpenZeppelin or a two-struct approach to handle the whitelist of caller addresses. This will prevent duplicate entries and improve the efficiency of the whitelist check.

Remediation Plan

SOLVED: The **Bonzo team** solved this issue as recommended. Moreover, a new struct was added to track the assets, this struct makes the addition, removal and most important, the check easier and more efficient for large amount of registered assets.

Remediation Hash

f3e259b8109e1f13de69ba8079f43b119b6ad643

7.5 HARDCODED TESTNET EVM ADDRESS INSTEAD OF MAINNET

// LOW

Description

The contract contains a hardcoded Ethereum address pointing to a testnet environment instead of the mainnet. This can lead to malfunction or security risks when deployed on the mainnet, as the address will not point to the intended contract or service. As shown in the SaucerSwap contract deployment page, the EVM addresses for WHBAR are:

- **MAINNET**: `0x0000000000000000000000000000000000163b59`
- **TESTNET**: `0x0000000000000000000000000000000000003ad1`

BVSS

AO:S/AC:L/AX:L/C:N/I:C/A:N/D:N/Y:N/R:N/S:C (2.5)

Recommendation

It is crucial to update the address to the mainnet equivalent before deployment, or use configurable parameters to set the correct environment-specific address dynamically, ensuring proper functionality and security across different environments.

Remediation Plan

SOLVED: The **Bonzo team** solved this issue as recommended.

Remediation Hash

5d9701676aa921a2a895a36097f5b710162ccdc5

References

AaveIncentivesController.sol#L82

7.6 INCREASE PERIOD_DURATION TO 30 DAYS

// INFORMATIONAL

Description

The contract contains a `TODO` comment indicating the intention to increase the immutable `PERIOD_DURATION` variable to 30 days at a later time. Leaving such `TODO`s unresolved can lead to inconsistencies and potential issues in the contract's functionality.

BVSS

AO:S/AC:L/AX:L/C:N/I:H/A:N/D:N/Y:N/R:N/S:C (1.9)

Recommendation

It is crucial to address this change promptly by updating `PERIOD_DURATION` to 30 days or removing the TODO if the change is no longer necessary. Ensuring that all `TODO`s are resolved enhances the contract's reliability and clarity.

Remediation Plan

SOLVED: The **Bonzo team** solved this issue as recommended.

Remediation Hash

5d9701676aa921a2a895a36097f5b710162ccdc5

References

AaveDistributionManager.sol#27-28

7.7 INCONSISTENT STRUCT PARAMETER UPDATE

// INFORMATIONAL

Description

The parameter `emissionPerSecond` of the `AssetData` struct in `AaveDistributionManager` has been modified to include an array of different emissions depending on the period, but this update was not reflected in the out-of-scope `IStakedToken.sol` interface, which is utilized in `StakeUIHelper.sol`. This inconsistency can lead to integration issues and potential bugs, as the outdated interface does not match the current contract implementation.

BVSS

AO:S/AC:L/AX:L/C:N/I:N/A:H/D:N/Y:N/R:F/S:C (0.5)

Recommendation

It is recommended to update the `IStakedToken.sol` interface to reflect the changes in the `emissionPerSecond` parameter, and, if willing to be used, refactor `StakeUIHelper.sol` accordingly in order to ensure seamless interaction between the contracts and preventing any discrepancies.

Remediation Plan

SOLVED: The **Bonzo team** solved this issue as recommended. The `IStakedToken` was removed.

Remediation Hash

`1ecf5995b8ed3b7fa84441a7be114b4dbdf551bc`

7.8 USAGE OF OUTDATED SAFECAST

// INFORMATIONAL

Description

The contract uses an outdated version of `SafeCast` that allows implicit casting from `uint` to `int`. This implicit casting can introduce vulnerabilities and unintended behavior due to the differences in value ranges between unsigned and signed integers. Updating to a newer version of `SafeCast` that requires explicit casting will ensure safer and more predictable type conversions, mitigating potential risks, and will allow the use of newer and safer compiler versions.

BVSS

AO:S/AC:L/AX:L/C:N/I:N/A:H/D:N/Y:N/R:F/S:C (0.5)

Recommendation

By using explicit type conversions, the contract ensures clarity and prevents unintended behavior, aligning with the safety improvements introduced in recent Solidity versions.

Remediation Plan

ACKNOWLEDGED: Ignoring as it causes compatibility issues with the older versions of the `SafeHederaTokenService` contract

References

SafeHederaTokenService.sol#6

7.9 USE OF MEMORY INSTEAD OF CALLDATA IN FUNCTION VARIABLE

// INFORMATIONAL

Description

The `configureAssets` function in the `AaveDistributionManager` contract declares the parameter `assetsConfigInput` as `memory`. However, since this parameter is read-only during the function execution, using `calldata` instead of `memory` is more efficient. `Calldata` directly references the input data without copying it to memory, resulting in reduced gas consumption and improved performance.

Score

AO:S/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:N/S:C (0.0)

Recommendation

Keeping the variable as `calldata` can optimize the function's performance and decrease transaction costs.

Remediation Plan

ACKNOWLEDGED: Changing assetsConfigInput to memory starts giving downstream compilation issues in `StakedToken.sol` and `AaveIncentivesController.sol`. So we are ignoring it.

References

AaveDistributionManager.sol#51

7.10 REDUNDANT PARAMETERS IN FUNCTION _GETASSETINDEX

// INFORMATIONAL

Description

The `_getAssetIndex` function in the `AaveDistributionManager` contract currently takes four parameters: `currentIndex`, `assetConfig`, `lastUpdateTimestamp`, and `totalBalance`. However, `assetConfig` already contains the `lastUpdateTimestamp` and `totalBalance` data, making the other two parameters redundant. This redundancy leads to unnecessary gas consumption and inefficiency. Simplifying the function to accept only the composite parameter `assetConfig` can optimize gas usage and improve overall performance.

Score

AO:S/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:N/S:C (0.0)

Recommendation

By eliminating the redundant parameters, the function becomes more efficient, reducing gas costs and enhancing performance. This approach ensures a more streamlined and cost-effective implementation.

Remediation Plan

SOLVED: The issue was solved by removing `lastUpdateTimestamp`.

Remediation Hash

`1ecf5995b8ed3b7fa84441a7be114b4dbdf551bc`

References

AaveDistributionManager.sol#225-230

8. AUTOMATED TESTING

Static Analysis Report

Description

Halborn used automated testing techniques to enhance the coverage of certain areas of the scoped contracts. Among the tools used was Slither, a Solidity static analysis framework. After Halborn verified all the contracts in the repository and was able to compile them correctly into their abi and binary formats, Slither was run on the all-scoped contracts. This tool can statically verify mathematical relationships between Solidity variables to detect invalid or inconsistent usage of the contracts' APIs across the entire code-base.

Slither Results

WHBARContract

```
INFO:Detectors:
WHBARContract.withdraw(address,address,uint256) (src/misc/WHBAR/WHBARContract.sol#66-79) sends eth to a
rbitrary user
        Dangerous calls:
        - (sent,None) = address(dst).call{value: wad}() (src/misc/WHBAR/WHBARContract.sol#76)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-a
rbitrary-destinations
INFO:Detectors:
WHBARContract.constructor().myToken (src/misc/WHBAR/WHBARContract.sol#30) is a local variable never ini
tialized
WHBARContract.constructor().expiry (src/misc/WHBAR/WHBARContract.sol#26) is a local variable never init
ialized
WHBARContract.constructor().supplyKeyValue (src/misc/WHBAR/WHBARContract.sol#18) is a local variable ne
ver initialized
WHBARContract.constructor().supplyKeyType (src/misc/WHBAR/WHBARContract.sol#17) is a local variable nev
er initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
IHederaTokenService.allowance(address,address,address).allowance (src/interfaces/IHederaTokenService.so
l#216) shadows:
        - IHederaTokenService.allowance(address,address,address) (src/interfaces/IHederaTokenService.so
l#212-216) (function)
IHederaTokenService.isToken(address).isToken (src/interfaces/IHederaTokenService.sol#330) shadows:
        - IHederaTokenService.isToken(address) (src/interfaces/IHederaTokenService.sol#330) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
WHBARContract.withdraw(address,address,uint256).dst (src/misc/WHBAR/WHBARContract.sol#68) lacks a zero-
check on :
                - (sent,None) = address(dst).call{value: wad}() (src/misc/WHBAR/WHBARContract.sol#76)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validatio
n
INFO:Detectors:
Reentrancy in WHBARContract.deposit() (src/misc/WHBAR/WHBARContract.sol#50-56):
        External calls:
        - safeMintToken(token,msg.sender,msg.value,new bytes[](0)) (src/misc/WHBAR/WHBARContract.sol#53
)
                - (success,result) = precompileAddress.call(abi.encodeWithSelector(IHederaTokenService.
mintToken.selector,token,amount,metadata)) (src/misc/WHBAR/HederaTokenService.sol#61-63)
        - safeTransferToken(token,address(this),msg.sender,msg.value) (src/misc/WHBAR/WHBARContract.sol
#54)
                - (success,result) = precompileAddress.call(abi.encodeWithSelector(IHederaTokenService.
transferToken.selector,token,sender,receiver,amount)) (src/misc/WHBAR/HederaTokenService.sol#138-140)
        Event emitted after the call(s):
        - Deposit(msg.sender,msg.sender,msg.value) (src/misc/WHBAR/WHBARContract.sol#55)
        - Transfer(sender,receiver,uint64(amount)) (src/misc/WHBAR/SafeHederaTokenService.sol#68)
                - safeTransferToken(token,address(this),msg.sender,msg.value) (src/misc/WHBAR/WHBARCont
ract.sol#54)
```

```
Reentrancy in WHBARContract.deposit(address,address) (src/misc/WHBAR/WHBARContract.sol#58-64):
        External calls:
        - safeMintToken(token,src,msg.value,new bytes[](0)) (src/misc/WHBAR/WHBARContract.sol#61)
                - (success,result) = precompileAddress.call(abi.encodeWithSelector(IHederaTokenService.
mintToken.selector,token,amount,metadata)) (src/misc/WHBAR/HederaTokenService.sol#61-63)
        - safeTransferToken(token,address(this),dst,msg.value) (src/misc/WHBAR/WHBARContract.sol#62)
                - (success,result) = precompileAddress.call(abi.encodeWithSelector(IHederaTokenService.
transferToken.selector,token,sender,receiver,amount)) (src/misc/WHBAR/HederaTokenService.sol#138-140)
        Event emitted after the call(s):
        - Deposit(src,dst,msg.value) (src/misc/WHBAR/WHBARContract.sol#63)
        - Transfer(sender,receiver,uint64(amount)) (src/misc/WHBAR/SafeHederaTokenService.sol#68)
                - safeTransferToken(token,address(this),dst,msg.value) (src/misc/WHBAR/WHBARContract.so
l#62)
Reentrancy in SafeHederaTokenService.safeBurnToken(address,address,uint256,int64[]) (src/misc/WHBAR/Saf
eHederaTokenService.sol#31-45):
        External calls:
        - (responseCode,newTotalSupply) = HederaTokenService.burnToken(token,amount.toUint64(),serialNu
mbers) (src/misc/WHBAR/SafeHederaTokenService.sol#38-42)
                - (success,result) = precompileAddress.call(abi.encodeWithSelector(IHederaTokenService.
burnToken.selector,token,amount,serialNumbers)) (src/misc/WHBAR/HederaTokenService.sol#82-84)
        Event emitted after the call(s):
        - Transfer(to,address(0),amount.toUint64()) (src/misc/WHBAR/SafeHederaTokenService.sol#44)
Reentrancy in SafeHederaTokenService.safeMintToken(address,address,uint256,bytes[]) (src/misc/WHBAR/Saf
eHederaTokenService.sol#15-29):
        External calls:
        - (responseCode,newTotalSupply,serialNumbers) = HederaTokenService.mintToken(token,amount.toUin
t64(),metadata) (src/misc/WHBAR/SafeHederaTokenService.sol#22-26)
                - (success,result) = precompileAddress.call(abi.encodeWithSelector(IHederaTokenService.
mintToken.selector,token,amount,metadata)) (src/misc/WHBAR/HederaTokenService.sol#61-63)
        Event emitted after the call(s):
        - Transfer(address(0),to,amount.toUint64()) (src/misc/WHBAR/SafeHederaTokenService.sol#28)
Reentrancy in SafeHederaTokenService.safeTransferToken(address,address,address,uint256) (src/misc/WHBAR
/SafeHederaTokenService.sol#59-69):
        External calls:
        - (responseCode) = HederaTokenService.transferToken(token,sender,receiver,amount.toInt64()) (sr
c/misc/WHBAR/SafeHederaTokenService.sol#66)
                - (success,result) = precompileAddress.call(abi.encodeWithSelector(IHederaTokenService.
transferToken.selector,token,sender,receiver,amount)) (src/misc/WHBAR/HederaTokenService.sol#138-140)
        Event emitted after the call(s):
        - Transfer(sender,receiver,uint64(amount)) (src/misc/WHBAR/SafeHederaTokenService.sol#68)
Reentrancy in SafeHederaTokenService.safeTransferTokenRouter(address,address,address,uint256) (src/misc
/WHBAR/SafeHederaTokenService.sol#71-86):
        External calls:
        - (responseCode) = HederaTokenService.transferTokenRouter(token,sender,receiver,amount.toInt64(
)) (src/misc/WHBAR/SafeHederaTokenService.sol#78-83)
                - (success,result) = precompileAddress.delegatecall(abi.encodeWithSelector(IHederaToken
Service.transferToken.selector,token,sender,receiver,amount)) (src/misc/WHBAR/HederaTokenService.sol#14
8-150)
        Event emitted after the call(s):
        - Transfer(sender,receiver,uint64(amount)) (src/misc/WHBAR/SafeHederaTokenService.sol#85)
```

```
Reentrancy in WHBARContract.withdraw(address,address,uint256) (src/misc/WHBAR/WHBARContract.sol#66-79):
        External calls:
        - safeTransferToken(token,src,address(this),wad) (src/misc/WHBAR/WHBARContract.sol#73)
                - (success,result) = precompileAddress.call(abi.encodeWithSelector(IHederaTokenService.
transferToken.selector,token,sender,receiver,amount)) (src/misc/WHBAR/HederaTokenService.sol#138-140)
        - safeBurnToken(token,src,wad,new int64[](0)) (src/misc/WHBAR/WHBARContract.sol#74)
                - (success,result) = precompileAddress.call(abi.encodeWithSelector(IHederaTokenService.
burnToken.selector,token,amount,serialNumbers)) (src/misc/WHBAR/HederaTokenService.sol#82-84)
        Event emitted after the call(s):
        - Transfer(to,address(0),amount.toUint64()) (src/misc/WHBAR/SafeHederaTokenService.sol#44)
                - safeBurnToken(token,src,wad,new int64[](0)) (src/misc/WHBAR/WHBARContract.sol#74)
Reentrancy in WHBARContract.withdraw(address,address,uint256) (src/misc/WHBAR/WHBARContract.sol#66-79):
        External calls:
        - safeTransferToken(token,src,address(this),wad) (src/misc/WHBAR/WHBARContract.sol#73)
                - (success,result) = precompileAddress.call(abi.encodeWithSelector(IHederaTokenService.
transferToken.selector,token,sender,receiver,amount)) (src/misc/WHBAR/HederaTokenService.sol#138-140)
        - safeBurnToken(token,src,wad,new int64[](0)) (src/misc/WHBAR/WHBARContract.sol#74)
                - (success,result) = precompileAddress.call(abi.encodeWithSelector(IHederaTokenService.
burnToken.selector,token,amount,serialNumbers)) (src/misc/WHBAR/HederaTokenService.sol#82-84)
        - (sent,None) = address(dst).call{value: wad}() (src/misc/WHBAR/WHBARContract.sol#76)
        External calls sending eth:
        - (sent,None) = address(dst).call{value: wad}() (src/misc/WHBAR/WHBARContract.sol#76)
        Event emitted after the call(s):
        - Withdrawal(src,dst,wad) (src/misc/WHBAR/WHBARContract.sol#78)
Reentrancy in WHBARContract.withdraw(uint256) (src/misc/WHBAR/WHBARContract.sol#81-90):
        External calls:
        - safeTransferToken(token,msg.sender,address(this),wad) (src/misc/WHBAR/WHBARContract.sol#84)
                - (success,result) = precompileAddress.call(abi.encodeWithSelector(IHederaTokenService.
transferToken.selector,token,sender,receiver,amount)) (src/misc/WHBAR/HederaTokenService.sol#138-140)
        - safeBurnToken(token,msg.sender,wad,new int64[](0)) (src/misc/WHBAR/WHBARContract.sol#85)
                - (success,result) = precompileAddress.call(abi.encodeWithSelector(IHederaTokenService.
burnToken.selector,token,amount,serialNumbers)) (src/misc/WHBAR/HederaTokenService.sol#82-84)
        Event emitted after the call(s):
        - Transfer(to,address(0),amount.toUint64()) (src/misc/WHBAR/SafeHederaTokenService.sol#44)
                - safeBurnToken(token,msg.sender,wad,new int64[](0)) (src/misc/WHBAR/WHBARContract.sol#
85)
Reentrancy in WHBARContract.withdraw(uint256) (src/misc/WHBAR/WHBARContract.sol#81-90):
        External calls:
        - safeTransferToken(token,msg.sender,address(this),wad) (src/misc/WHBAR/WHBARContract.sol#84)
                - (success,result) = precompileAddress.call(abi.encodeWithSelector(IHederaTokenService.
transferToken.selector,token,sender,receiver,amount)) (src/misc/WHBAR/HederaTokenService.sol#138-140)
        - safeBurnToken(token,msg.sender,wad,new int64[](0)) (src/misc/WHBAR/WHBARContract.sol#85)
                - (success,result) = precompileAddress.call(abi.encodeWithSelector(IHederaTokenService.
burnToken.selector,token,amount,serialNumbers)) (src/misc/WHBAR/HederaTokenService.sol#82-84)
        - (sent,None) = address(msg.sender).call{value: wad}() (src/misc/WHBAR/WHBARContract.sol#87)
        External calls sending eth:
        - (sent,None) = address(msg.sender).call{value: wad}() (src/misc/WHBAR/WHBARContract.sol#87)
        Event emitted after the call(s):
        - Withdrawal(msg.sender,msg.sender,wad) (src/misc/WHBAR/WHBARContract.sol#89)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```

StakedAave

```
INFO:Detectors:
StakedToken.claimRewards(address,uint256) (src/stake/StakedToken.sol#196-206) uses arbitrary from in tr
ansferFrom: REWARD_TOKEN.safeTransferFrom(REWARDS_VAULT,to,amountToClaim) (src/stake/StakedToken.sol#20
3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#arbitrary-from-in-transferfrom
INFO:Detectors:
AaveDistributionManager._getAssetIndex(uint256,AaveDistributionManager.AssetData,uint128,uint256) (src/
stake/AaveDistributionManager.sol#218-269) uses a dangerous strict equality:
        - totalBalance == 0 || lastUpdateTimestamp == block.timestamp || lastUpdateTimestamp >= DISTRIB
UTION_END (src/stake/AaveDistributionManager.sol#225-227)
AaveDistributionManager._updateAssetStateInternal(address,AaveDistributionManager.AssetData,uint256) (s
rc/stake/AaveDistributionManager.sol#83-105) uses a dangerous strict equality:
        - block.timestamp == lastUpdateTimestamp (src/stake/AaveDistributionManager.sol#91)
ERC20WithSnapshot._writeSnapshot(address,uint128,uint128) (src/lib/ERC20WithSnapshot.sol#47-69) uses a
dangerous strict equality:
        - ownerCountOfSnapshots != 0 && snapshotsOwner[ownerCountOfSnapshots.sub(1)].blockNumber == cur
rentBlock (src/lib/ERC20WithSnapshot.sol#59-60)
StakedToken.getNextCooldownTimestamp(uint256,uint256,address,uint256) (src/stake/StakedToken.sol#284-31
6) uses a dangerous strict equality:
        - toCooldownTimestamp == 0 (src/stake/StakedToken.sol#291)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Reentrancy in StakedToken.redeem(address,uint256) (src/stake/StakedToken.sol#150-177):
        External calls:
        - _burn(msg.sender,amountToRedeem) (src/stake/StakedToken.sol#168)
                - aaveGovernance.onTransfer(from,to,amount) (src/lib/ERC20WithSnapshot.sol#101)
        State variables written after the call(s):
        - stakersCooldowns[msg.sender] = 0 (src/stake/StakedToken.sol#171)
        StakedToken.stakersCooldowns (src/stake/StakedToken.sol#43) can be used in cross function reent
rancies:
        - StakedToken._transfer(address,address,uint256) (src/stake/StakedToken.sol#214-242)
        - StakedToken.cooldown() (src/stake/StakedToken.sol#183-189)
        - StakedToken.getNextCooldownTimestamp(uint256,uint256,address,uint256) (src/stake/StakedToken.
sol#284-316)
        - StakedToken.redeem(address,uint256) (src/stake/StakedToken.sol#150-177)
        - StakedToken.stake(address,uint256) (src/stake/StakedToken.sol#126-143)
        - StakedToken.stakersCooldowns (src/stake/StakedToken.sol#43)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
IHederaTokenService.allowance(address,address,address).allowance (src/interfaces/IHederaTokenService.so
l#216) shadows:
        - IHederaTokenService.allowance(address,address,address) (src/interfaces/IHederaTokenService.so
l#212-216) (function)
IHederaTokenService.isToken(address).isToken (src/interfaces/IHederaTokenService.sol#330) shadows:
        - IHederaTokenService.isToken(address) (src/interfaces/IHederaTokenService.sol#330) (function)
ERC20.constructor(string,string,uint8).name (src/lib/ERC20.sol#25) shadows:
        - ERC20.name() (src/lib/ERC20.sol#37-39) (function)
        - IERC20Detailed.name() (src/interfaces/IERC20Detailed.sol#10) (function)
ERC20.constructor(string,string,uint8).symbol (src/lib/ERC20.sol#26) shadows:
        - ERC20.symbol() (src/lib/ERC20.sol#44-46) (function)
        - IERC20Detailed.symbol() (src/interfaces/IERC20Detailed.sol#12) (function)
ERC20.constructor(string,string,uint8).decimals (src/lib/ERC20.sol#27) shadows:
        - ERC20.decimals() (src/lib/ERC20.sol#51-53) (function)
        - IERC20Detailed.decimals() (src/interfaces/IERC20Detailed.sol#14) (function)
ERC20WithSnapshot.constructor(string,string,uint8).name (src/lib/ERC20WithSnapshot.sol#32) shadows:
        - ERC20.name() (src/lib/ERC20.sol#37-39) (function)
        - IERC20Detailed.name() (src/interfaces/IERC20Detailed.sol#10) (function)
```

```
StakedToken.constructor(IERC20,IERC20,uint256,uint256,address,address,uint128,string,string,uint8).name
 (src/stake/StakedToken.sol#65) shadows:
        - ERC20.name() (src/lib/ERC20.sol#37-39) (function)
        - IERC20Detailed.name() (src/interfaces/IERC20Detailed.sol#10) (function)
StakedToken.constructor(IERC20,IERC20,uint256,uint256,address,address,uint128,string,string,uint8).symb
ol (src/stake/StakedToken.sol#66) shadows:
        - ERC20.symbol() (src/lib/ERC20.sol#44-46) (function)
        - IERC20Detailed.symbol() (src/interfaces/IERC20Detailed.sol#12) (function)
StakedToken.constructor(IERC20,IERC20,uint256,uint256,address,address,uint128,string,string,uint8).deci
mals (src/stake/StakedToken.sol#67) shadows:
        - ERC20.decimals() (src/lib/ERC20.sol#51-53) (function)
        - IERC20Detailed.decimals() (src/interfaces/IERC20Detailed.sol#14) (function)
StakedToken.initialize(ITransferHook,string,string,uint8,uint128[]).name (src/stake/StakedToken.sol#85)
 shadows:
        - ERC20.name() (src/lib/ERC20.sol#37-39) (function)
        - IERC20Detailed.name() (src/interfaces/IERC20Detailed.sol#10) (function)
StakedToken.initialize(ITransferHook,string,string,uint8,uint128[]).symbol (src/stake/StakedToken.sol#8
6) shadows:
        - ERC20.symbol() (src/lib/ERC20.sol#44-46) (function)
        - IERC20Detailed.symbol() (src/interfaces/IERC20Detailed.sol#12) (function)
StakedToken.initialize(ITransferHook,string,string,uint8,uint128[]).decimals (src/stake/StakedToken.sol
#87) shadows:
        - ERC20.decimals() (src/lib/ERC20.sol#51-53) (function)
        - IERC20Detailed.decimals() (src/interfaces/IERC20Detailed.sol#14) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
StakedToken.constructor(IERC20,IERC20,uint256,uint256,address,address,uint128,string,string,uint8).rewa
rdsVault (src/stake/StakedToken.sol#62) lacks a zero-check on :
                - REWARDS_VAULT = rewardsVault (src/stake/StakedToken.sol#77)
AaveDistributionManager.constructor(address,uint256).emissionManager (src/stake/AaveDistributionManager
.sol#40) lacks a zero-check on :
                - EMISSION_MANAGER = emissionManager (src/stake/AaveDistributionManager.sol#43)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validatio
n
INFO:Detectors:
Reentrancy in StakedToken.initialize(ITransferHook,string,string,uint8,uint128[]) (src/stake/StakedToke
n.sol#83-109):
        External calls:
        - _associate(address(STAKED_TOKEN)) (src/stake/StakedToken.sol#94)
                - (success,result) = hts.call(abi.encodeWithSelector(IHederaTokenService.associateToken
.selector,address(this),_asset)) (src/stake/StakedToken.sol#116-119)
        State variables written after the call(s):
        - configureAssets(configInput) (src/stake/StakedToken.sol#108)
                - assetConfig.emissionPerSecond[period] = assetsConfigInput[i].emissionPerSecond[period
] (src/stake/AaveDistributionManager.sol#66)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in StakedToken.claimRewards(address,uint256) (src/stake/StakedToken.sol#196-206):
        External calls:
        - REWARD_TOKEN.safeTransferFrom(REWARDS_VAULT,to,amountToClaim) (src/stake/StakedToken.sol#203)
        Event emitted after the call(s):
        - RewardsClaimed(msg.sender,to,amountToClaim) (src/stake/StakedToken.sol#205)
Reentrancy in StakedToken.initialize(ITransferHook,string,string,uint8,uint128[]) (src/stake/StakedToke
n.sol#83-109):
        External calls:
        - _associate(address(STAKED_TOKEN)) (src/stake/StakedToken.sol#94)
                - (success,result) = hts.call(abi.encodeWithSelector(IHederaTokenService.associateToken
.selector,address(this),_asset)) (src/stake/StakedToken.sol#116-119)
```

```
        Event emitted after the call(s):
        - AssetConfigUpdated(assetsConfigInput[i].underlyingAsset,assetsConfigInput[i].emissionPerSecon
d[0]) (src/stake/AaveDistributionManager.sol#69-72)
                - configureAssets(configInput) (src/stake/StakedToken.sol#108)
        - AssetIndexUpdated(underlyingAsset,newIndex) (src/stake/AaveDistributionManager.sol#99)
                - configureAssets(configInput) (src/stake/StakedToken.sol#108)
Reentrancy in StakedToken.redeem(address,uint256) (src/stake/StakedToken.sol#150-177):
        External calls:
        - _burn(msg.sender,amountToRedeem) (src/stake/StakedToken.sol#168)
                - aaveGovernance.onTransfer(from,to,amount) (src/lib/ERC20WithSnapshot.sol#101)
        - IERC20(STAKED_TOKEN).safeTransfer(to,amountToRedeem) (src/stake/StakedToken.sol#174)
        Event emitted after the call(s):
        - Redeem(msg.sender,to,amountToRedeem) (src/stake/StakedToken.sol#176)
Reentrancy in StakedToken.stake(address,uint256) (src/stake/StakedToken.sol#126-143):
        External calls:
        - _mint(onBehalfOf,amount) (src/stake/StakedToken.sol#139)
                - aaveGovernance.onTransfer(from,to,amount) (src/lib/ERC20WithSnapshot.sol#101)
        - IERC20(STAKED_TOKEN).safeTransferFrom(msg.sender,address(this),amount) (src/stake/StakedToken
.sol#140)
        Event emitted after the call(s):
        - Staked(msg.sender,onBehalfOf,amount) (src/stake/StakedToken.sol#142)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
AaveDistributionManager._updateAssetStateInternal(address,AaveDistributionManager.AssetData,uint256) (s
rc/stake/AaveDistributionManager.sol#83-105) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp == lastUpdateTimestamp (src/stake/AaveDistributionManager.sol#91)
AaveDistributionManager._getAssetIndex(uint256,AaveDistributionManager.AssetData,uint128,uint256) (src/
stake/AaveDistributionManager.sol#218-269) uses timestamp for comparisons
        Dangerous comparisons:
        - totalBalance == 0 || lastUpdateTimestamp == block.timestamp || lastUpdateTimestamp >= DISTRIB
UTION_END (src/stake/AaveDistributionManager.sol#225-227)
        - periodStartTimestamp < currentTimestamp (src/stake/AaveDistributionManager.sol#239)
        - nextPeriodTimestamp > currentTimestamp (src/stake/AaveDistributionManager.sol#241)
        - block.timestamp > DISTRIBUTION_END (src/stake/AaveDistributionManager.sol#232-233)
StakedToken.redeem(address,uint256) (src/stake/StakedToken.sol#150-177) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp > cooldownStartTimestamp.add(COOLDOWN_SECONDS),INSUFFICI
ENT_COOLDOWN) (src/stake/StakedToken.sol#154-157)
        - require(bool,string)(block.timestamp.sub(cooldownStartTimestamp.add(COOLDOWN_SECONDS)) <= UNS
TAKE_WINDOW,UNSTAKE_WINDOW_FINISHED) (src/stake/StakedToken.sol#158-161)
StakedToken.getNextCooldownTimestamp(uint256,uint256,address,uint256) (src/stake/StakedToken.sol#284-31
6) uses timestamp for comparisons
        Dangerous comparisons:
        - toCooldownTimestamp == 0 (src/stake/StakedToken.sol#291)
        - minimalValidCooldownTimestamp > toCooldownTimestamp (src/stake/StakedToken.sol#298)
        - fromCooldownTimestamp_scope_0 < toCooldownTimestamp (src/stake/StakedToken.sol#306)
        - (minimalValidCooldownTimestamp > fromCooldownTimestamp) (src/stake/StakedToken.sol#301-304)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
```

The reentrancy instances flagged by Slither were checked individually, and have been categorised as false positives.

Halborn strongly recommends conducting a follow-up assessment of the project either within six months or immediately following any material changes to the codebase, whichever comes first. This approach is crucial for maintaining the project's integrity and addressing potential vulnerabilities introduced by code modifications.