Biconomy - Safety Module

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **March 15th, 2022 - March 28th, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	03/28/2022	Ataberk Yavuzer
0.2	Document Edits	03/28/2022	Ataberk Yavuzer
0.3	Document Edits	03/30/2022	Gabi Urrutia
1.0	Remediation Plan	04/08/2022	Omar Alshaeb
1.1	Remediation Plan Review	04/08/2022	Gabi Urrutia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
Ataberk Yavuzer	Halborn	Ataberk.Yavuzer@halborn.com
Omar Alshaeb	Halborn	Omar.Alshaeb@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Biconomy engaged Halborn to conduct a security audit on their smart contracts beginning on March 15th, 2022 and ending on March 28th, 2022 . The security assessment was scoped to the smart contracts provided to the Halborn team.

1.2 AUDIT SUMMARY

The team at Halborn was provided two weeks for the engagement and assigned a full-time security engineer to audit 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 audit 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 Biconomy team.

1.3 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy regarding the scope of the smart contract audit. 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 audit:

- Research into architecture and purpose.
- Smart Contract manual code review and walkthrough.
- Graphing out functionality and contract logic/connectivity/functions
- Manual Assessment of use and safety for the critical Solidity variables and functions in scope to identify any arithmetic related vulnerability classes.
- Dynamic Analysis (ganache-cli, brownie, hardhat)
- Static Analysis(slither)

RISK METHODOLOGY:

Vulnerabilities or issues observed by Halborn are ranked based on the risk assessment methodology by measuring the **LIKELIHOOD** of a security incident and the **IMPACT** should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. The quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that were used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

RISK SCALE - LIKELIHOOD

5 - Almost certain an incident will occur.
4 - High probability of an incident occurring.
3 - Potential of a security incident in the long term.
2 - Low probability of an incident occurring.
1 - Very unlikely issue will cause an incident.

RISK SCALE - IMPACT

5 - May cause devastating and unrecoverable impact or loss.
4 - May cause a significant level of impact or loss.
3 - May cause a partial impact or loss to many.
2 - May cause temporary impact or loss.
1 - May cause minimal or un-noticeable impact.

The risk level is then calculated using a sum of these two values, creating a value of 10 to 1 with 10 being the highest level of security risk.

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL

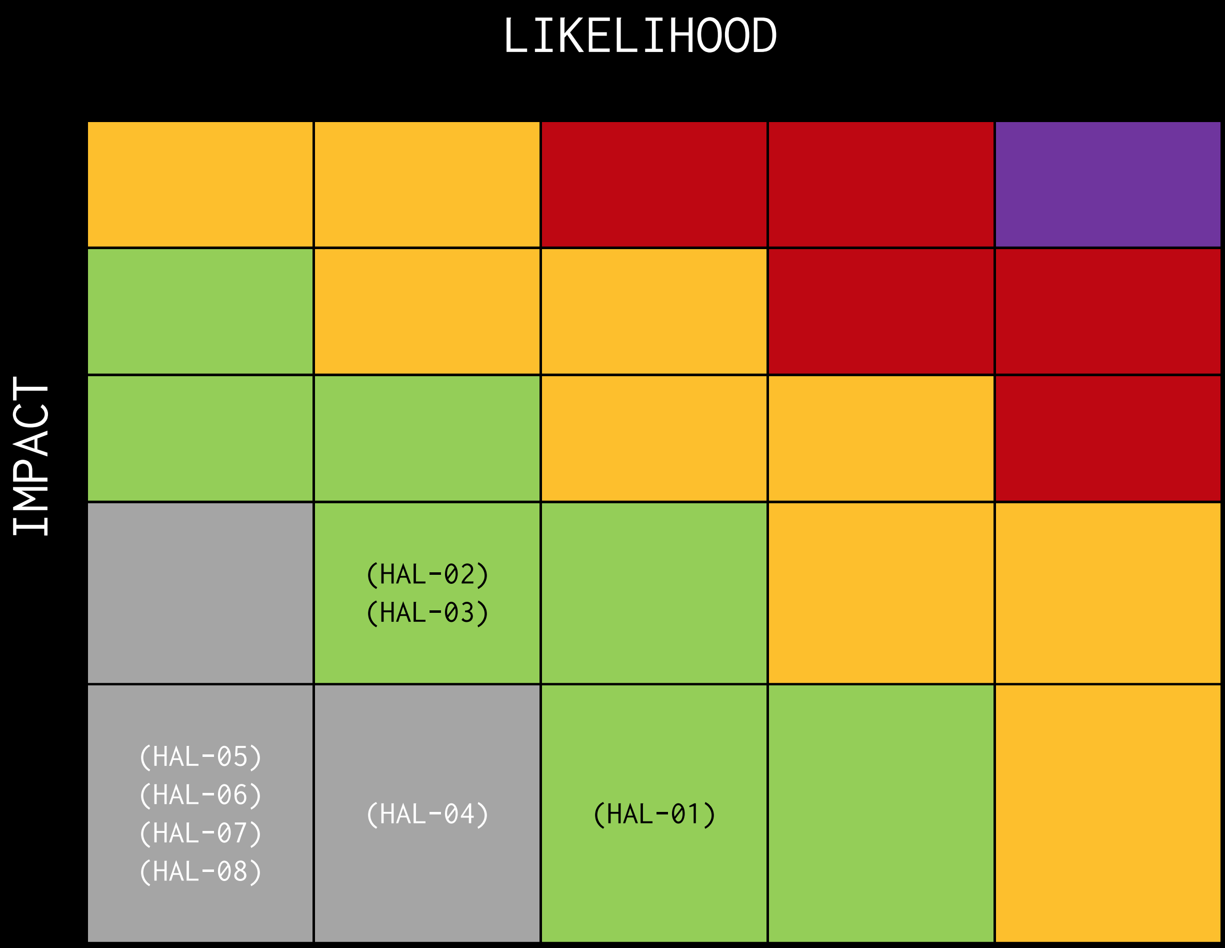
10 - CRITICAL
9 - 8 - HIGH
7 - 6 - MEDIUM
5 - 4 - LOW
3 - 1 - VERY LOW AND INFORMATIONAL

1.4 SCOPE

1. Safety Module
   (a) AaveDistributionManager.sol
   (b) BicoProtocolEcosystemReserve.sol
   (c) DistributionTypes.sol
   (d) InitializableAdminUpgradeabilityProxy.sol
   (e) StakedTokenBptRev2.sol
   (f) StakedTokenV3.sol
   (g) lib/GovernancePowerDelegationERC20.sol
   (h) lib/GovernancePowerWithSnapshot.sol
   (i) lib/VersionedInitializable.sol
2. Out-of-Scope Contracts
   (a) helper/*.sol
   (b) interface/*.sol
   (c) lib/*.sol
   (d) mock/*.sol

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	0	3	5



SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
(HAL-01) IGNORED RETURN VALUES	Low	SOLVED - 04/08/2022
(HAL-02) MISSING ZERO ADDRESS CHECKS	Low	SOLVED - 04/08/2022
(HAL-03) MISSING REENTRANCY GUARD	Low	SOLVED - 04/08/2022
(HAL-04) EXPERIMENTAL KEYWORD USAGE	Informational	ACKNOWLEDGED
(HAL-05) FLOATING PRAGMA	Informational	SOLVED - 04/08/2022
(HAL-06) USE 1E18 CONSTANT FOR GAS OPTIMIZATION	Informational	SOLVED - 04/08/2022
(HAL-07) MISUSE OF PUBLIC FUNCTIONS	Informational	SOLVED - 04/08/2022
(HAL-08) USE ++I INSTEAD OF I++ IN LOOPS FOR GAS OPTIMIZATION	Informational	ACKNOWLEDGED

FINDINGS & TECH DETAILS

3.1 (HAL-01) IGNORED RETURN VALUES - LOW

Description:

The return value of an external call is not stored in a local or state variable. In the BicoProtocolEcosystemReserve.sol contract, there is a function that ignores the return value.

Code Location:

Listing 1: BicoProtocolEcosystemReserve.sol (Line 170)

```
165 function transfer(
166     IERC20 token,
167     address recipient,
168     uint256 amount
169   ) external onlyFundsAdmin {
170     token.transfer(recipient, amount);
171   }
```

Risk Level:

Likelihood - 3
Impact - 1

Recommendation:

Add a return value check to prevent an unexpected contract crash. Return value checks provide better exception handling. As another solution, be sure to check the return value of the transfer.

Remediation Plan:

SOLVED: The issue was solved in commit ID:
20611133e6edd306f15840a2e5735577f1f08050

3.2 (HAL-02) MISSING ZERO ADDRESS CHECKS - LOW

Description:

Safety Module contracts have address fields in multiple functions. These functions are missing address validations. Each address should be validated and checked to be non-zero. This is also considered as a best practice.

During testing, it has been found that some of these inputs are not protected against using `address(0)` as the target address.

Code Location:

Listing 2: AaveDistributionManager.sol (Line 37)

```
36 constructor(address emissionManager, uint256 distributionDuration)
    ↳ public {
37     DISTRIBUTION_END = block.timestamp.add(distributionDuration);
38     EMISSION_MANAGER = emissionManager;
39   }
```

Listing 3: BicoProtocolEcosystemReserve.sol (Line 178)

```
177 function _setFundsAdmin(address admin) internal {
178     _fundsAdmin = admin;
179     emit NewFundsAdmin(admin);
180   }
```

Listing 4: StakedTokenBptRev2.sol (Lines 94,95,98)

```
81 constructor(
82     IERC20 stakedToken,
83     IERC20 rewardToken,
84     uint256 cooldownSeconds,
85     uint256 unstakeWindow,
86     address rewardsVault,
```

```
87      address emissionManager,
88      uint128 distributionDuration,
89      string memory name,
90      string memory symbol,
91      uint8 decimals,
92      address governance
93    ) public ERC20(name, symbol) AaveDistributionManager(
    ↳ emissionManager, distributionDuration) {
94      STAKED_TOKEN = stakedToken;
95      REWARD_TOKEN = rewardToken;
96      COOLDOWN_SECONDS = cooldownSeconds;
97      UNSTAKE_WINDOW = unstakeWindow;
98      REWARDS_VAULT = rewardsVault;
99      _aaveGovernance = ITransferHook(governance);
100     ERC20._setupDecimals(decimals);
101   }
```

Listing 5: StakedTokenV3.sol (Lines 94,95,98)

```
81  constructor(
82      IERC20 stakedToken,
83      IERC20 rewardToken,
84      uint256 cooldownSeconds,
85      uint256 unstakeWindow,
86      address rewardsVault,
87      address emissionManager,
88      uint128 distributionDuration,
89      string memory name,
90      string memory symbol,
91      uint8 decimals,
92      address governance
93    ) public ERC20(name, symbol) AaveDistributionManager(
    ↳ emissionManager, distributionDuration) {
94      STAKED_TOKEN = stakedToken;
95      REWARD_TOKEN = rewardToken;
96      COOLDOWN_SECONDS = cooldownSeconds;
97      UNSTAKE_WINDOW = unstakeWindow;
98      REWARDS_VAULT = rewardsVault;
99      _aaveGovernance = ITransferHook(governance);
100     ERC20._setupDecimals(decimals);
101   }
```

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

It is recommended to validate that each address input is non-zero.

Remediation Plan:

SOLVED: The issue was solved in commit ID:
20611133e6edd306f15840a2e5735577f1f08050

3.3 (HAL-03) MISSING REENTRANCY GUARD - LOW

Description:

To protect against cross-function re-entrancy attacks, it may be necessary to use a mutex. By using this lock, an attacker can no longer exploit the withdrawal function with a recursive call. OpenZeppelin has its own mutex implementation called ReentrancyGuard which provides a modifier to any function called nonReentrant that protects the function with a mutex against re-entrancy attacks.

Code Location:

Listing 6: Missing Reentrancy Guard - Functions

```
StakedTokenBptRev2::redeem()
StakedTokenBptRev2::claimRewards()
StakedTokenV3::redeem()
StakedTokenV3::claimRewards()
```

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

The functions in the code location section are missing nonReentrant modifiers. It is recommended to add the OpenZeppelin ReentrancyGuard library to the project and use the nonReentrant modifier to avoid introducing future re-entrancy vulnerabilities.

Remediation Plan:

SOLVED: The issue was solved in commit ID: 20611133e6edd306f15840a2e5735577f1f08050

3.4 (HAL-04) EXPERIMENTAL KEYWORD USAGE - INFORMATIONAL

Description:

ABIEncoderV2 is enabled and using experimental features could be dangerous in live deployments. The experimental ABI encoder does not handle non-integer values shorter than 32 bytes properly. This applies to bytesNN types, bool, enum and other types when they are part of an array or a struct and encoded directly from storage. This means these storage references have to be used directly inside abi.encode(...) as arguments in external function calls or in event data without prior assignment to a local variable. The types **bytesNN** and **bool** will result in corrupted data, while enum might lead to an invalid revert.

Code Location:

Listing 7: AaveDistributionManager.sol (Line 4)

```
pragma experimental ABIEncoderV2;
```

Listing 8: StakedTokenBptRev2.sol (Line 10)

```
pragma experimental ABIEncoderV2;
```

Listing 9: StakedTokenV3.sol (Line 10)

```
pragma experimental ABIEncoderV2;
```

Risk Level:

Likelihood - 2
Impact - 1

Recommendation:

When possible, do not use experimental features in the final live deployment.

Remediation Plan:

ACKNOWLEDGED: The Biconomy team acknowledged this issue.

3.5 (HAL-05) FLOATING PRAGMA - INFORMATIONAL

Description:

The project contains many instances of floating pragma. Contracts should be deployed with the same compiler version and flags that they have been tested with thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, either an outdated compiler version that might introduce bugs that affect the contract system negatively or a pragma version too recent which has not been extensively tested.

Code Location:

Listing 10: Floating Pragma

```
AaveDistributionManager.sol::pragma solidity ^0.7.5;
DistributionTypes.sol::pragma solidity ^0.7.5;
StakedTokenBptRev2.sol::pragma solidity ^0.7.5;
StakedTokenV3.sol::pragma solidity ^0.7.5;
GovernancePowerDelegationERC20.sol::pragma solidity ^0.7.5;
GovernancePowerWithSnapshot.sol::pragma solidity ^0.7.5;
VersionedInitializable.sol::pragma solidity ^0.7.5;
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Consider locking the pragma version with known bugs for the compiler version by removing the caret (^) symbol. When possible, do not use floating pragma in the final live deployment. Specifying a fixed compiler version ensures that the bytecode produced does not vary between builds.

This is especially important if you rely on bytecode-level verification of the code.

Remediation Plan:

SOLVED: The issue was solved in commit ID: 20611133e6edd306f15840a2e5735577f1f08050

3.6 (HAL-06) USE 1E18 CONSTANT FOR GAS OPTIMIZATION - INFORMATIONAL

Description:

In Solidiy, the exponentiation operation (**) costs up to 10 gas. It is possible to consume less gas to calculate the prices of the tokens if DECIMAL variable is fixed.

Code Location:

Listing 11: AaveDistributionManager (Line 201)

```
196 function _getRewards(
197     uint256 principalUserBalance,
198     uint256 reserveIndex,
199     uint256 userIndex
200   ) internal pure returns (uint256) {
201     return principalUserBalance.mul(reserveIndex.sub(userIndex)).div(10**uint256(PRECISION));
202   }
```

Listing 12: AaveDistributionManager (Line 231)

```
227 uint256 currentTimestamp =
228       block.timestamp > DISTRIBUTION_END ? DISTRIBUTION_END : block.timestamp;
229     uint256 timeDelta = currentTimestamp.sub(lastUpdateTimestamp);
230     return
231       emissionPerSecond.mul(timeDelta).mul(10**uint256(PRECISION)).div(totalBalance).add(
232         currentIndex
233       );
234   }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to use 1e18 instead of (10 ** 18) in price calculations to optimize gas usage.

Remediation Plan:

SOLVED: The issue was solved in commit ID:
20611133e6edd306f15840a2e5735577f1f08050

3.7 (HAL-07) MISUSE OF PUBLIC FUNCTIONS - INFORMATIONAL

Description:

In public functions, the array arguments are immediately copied into memory, while external functions can read directly from the calldata. Reading calldata is cheaper than allocating memory.

Public functions need to write arguments to memory because public functions can be called internally. Internal calls are passed internally via pointers to memory. Therefore, a function expects its arguments to be located in memory when the compiler generates the code for an internal function.

Code Location:

Listing 13: Misuse of Public Functions

```
AaveDistributionManager.getUserAssetData(address,address)
BicoProtocolEcosystemReserve.setFundsAdmin(address)
StakedTokenBptRev2.delegateByTypeBySig(address,
↳ IGovernancePowerDelegationToken.DelegationType,uint256,uint256,
↳ uint8,bytes32,bytes32)
StakedTokenBptRev2.delegateBySig(address,uint256,uint256,uint8,
↳ bytes32,bytes32)

```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Consider as much as possible declaring external functions instead of public functions. As for best practice, you should use external if you expect the function to be called only externally, and use public if you need to call the function internally. In short, public functions can be accessed by everyone, while external functions can only be accessed externally.

Remediation Plan:

SOLVED: The issue was solved in commit ID: 20611133e6edd306f15840a2e5735577f1f08050

3.8 (HAL-08) USE ++I INSTEAD OF I++ IN LOOPS FOR GAS OPTIMIZATION - INFORMATIONAL

Description:

In all the loops, the variable i is incremented using i++. It is known that, in loops, using ++i costs less gas per iteration than i++. This also affects variables incremented inside the loop code block.

Code Location:

Listing 14: AaveDistributionManager.sol (Line 51)

```
51 for (uint256 i = 0; i < assetsConfigInput.length; i++) {
52       AssetData storage assetConfig = assets[assetsConfigInput[i].
 ↳ underlyingAsset];
53
```

Listing 15: AaveDistributionManager.sol (Line 145)

```
145 for (uint256 i = 0; i < stakes.length; i++) {
146       accruedRewards = accruedRewards.add(
147         _updateUserAssetInternal(
```

Listing 16: AaveDistributionManager.sol (Line 172)

```
172  for (uint256 i = 0; i < stakes.length; i++) {
173       AssetData storage assetConfig = assets[stakes[i].
 ↳ underlyingAsset];
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to use `++i` instead of `i++` to increment the value of an `uint` variable inside a loop. This also applies to the variables declared inside the `for` loop, not just the iterator. On the other hand, this is not applicable outside of loops.

Remediation Plan:

ACKNOWLEDGED: The Biconomy team acknowledged this issue.

AUTOMATED TESTING

4.1 STATIC ANALYSIS REPORT

Description:

Halborn used automated testing techniques to enhance coverage of certain areas of the scoped contract. 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. 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.

Results:

```
BicoProtocolEcosystemReserve.transfer(IERC20,address,uint256) (contracts/BicoProtocolEcosystemReserve.sol#126-132) ignores return value by token.transfer(recipient,amount) (contracts/BicoProtocolEcosystemR
eserve.sol#131)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

BicoProtocolEcosystemReserve.approve(IERC20,address,uint256) (contracts/BicoProtocolEcosystemReserve.sol#118-124) ignores return value by token.approve(recipient,amount) (contracts/BicoProtocolEcosystemRes
erve.sol#123)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

Different versions of Solidity is used:
        - Version used: ['0.7.5', '^0.7.5']
        - 0.7.5 (contracts/BicoProtocolEcosystemReserve.sol#7)
        - ^0.7.5 (contracts/helper/VersionedInitializable.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

VersionedInitializable.getRevision() (contracts/helper/VersionedInitializable.sol#38) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Variable BicoProtocolEcosystemReserve._fundsAdmin (contracts/BicoProtocolEcosystemReserve.sol#97) is not in mixedCase
Variable VersionedInitializable.______gap (contracts/helper/VersionedInitializable.sol#41) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

VersionedInitializable.______gap (contracts/helper/VersionedInitializable.sol#41) is never used in BicoProtocolEcosystemReserve (contracts/BicoProtocolEcosystemReserve.sol#94-143)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

setFundsAdmin(address) should be declared external:
        - BicoProtocolEcosystemReserve.setFundsAdmin(address) (contracts/BicoProtocolEcosystemReserve.sol#134-136)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

```
GovernancePowerWithSnapshot._votingSnapshots (contracts/helper/GovernancePowerWithSnapshot.sol#23) is never initialized. It is used in:
        - StakedTokenBptRev2._getDelegationDataByType(IGovernancePowerDelegationToken.DelegationType) (contracts/StakedTokenBptRev2.sol#457-476)
GovernancePowerWithSnapshot._votingSnapshotsCounts (contracts/helper/GovernancePowerWithSnapshot.sol#24) is never initialized. It is used in:
        - StakedTokenBptRev2._getDelegationDataByType(IGovernancePowerDelegationToken.DelegationType) (contracts/StakedTokenBptRev2.sol#457-476)
StakedTokenBptRev2._votingDelegates (contracts/StakedTokenBptRev2.sol#57) is never initialized. It is used in:
        - StakedTokenBptRev2._beforeTokenTransfer(address,address,uint256) (contracts/StakedTokenBptRev2.sol#411-455)
        - StakedTokenBptRev2._getDelegationDataByType(IGovernancePowerDelegationToken.DelegationType) (contracts/StakedTokenBptRev2.sol#457-476)
StakedTokenBptRev2._propositionPowerSnapshots (contracts/StakedTokenBptRev2.sol#59) is never initialized. It is used in:
        - StakedTokenBptRev2._getDelegationDataByType(IGovernancePowerDelegationToken.DelegationType) (contracts/StakedTokenBptRev2.sol#457-476)
StakedTokenBptRev2._propositionPowerSnapshotsCounts (contracts/StakedTokenBptRev2.sol#60) is never initialized. It is used in:
        - StakedTokenBptRev2._getDelegationDataByType(IGovernancePowerDelegationToken.DelegationType) (contracts/StakedTokenBptRev2.sol#457-476)
StakedTokenBptRev2._propositionPowerDelegates (contracts/StakedTokenBptRev2.sol#61) is never initialized. It is used in:
        - StakedTokenBptRev2._beforeTokenTransfer(address,address,uint256) (contracts/StakedTokenBptRev2.sol#411-455)
        - StakedTokenBptRev2._getDelegationDataByType(IGovernancePowerDelegationToken.DelegationType) (contracts/StakedTokenBptRev2.sol#457-476)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables

AaveDistributionManager._getAssetIndex(uint256,uint256,uint128,uint256) (contracts/AaveDistributionManager.sol#212-234) uses a dangerous strict equality:
        - emissionPerSecond == 0 || totalBalance == 0 || lastUpdateTimestamp == block.timestamp || lastUpdateTimestamp >= DISTRIBUTION_END (contracts/AaveDistributionManager.sol#219-222)
GovernancePowerDelegationERC20._searchByBlockNumber(mapping(address => mapping(uint256 => GovernancePowerDelegationERC20.Snapshot)),mapping(address => uint256),address,uint256) (contracts/helper/Governance
PowerDelegationERC20.sol#203-241) uses a dangerous strict equality:
        - snapshot.blockNumber == blockNumber (contracts/helper/GovernancePowerDelegationERC20.sol#232)
AaveDistributionManager._updateAssetStateInternal(address,AaveDistributionManager.AssetData,uint256) (contracts/AaveDistributionManager.sol#76-99) uses a dangerous strict equality:
        - block.timestamp == lastUpdateTimestamp (contracts/AaveDistributionManager.sol#84)
GovernancePowerDelegationERC20._writeSnapshot(mapping(address => mapping(uint256 => GovernancePowerDelegationERC20.Snapshot)),mapping(address => uint256),address,uint128,uint128) (contracts/helper/Governan
cePowerDelegationERC20.sol#266-288) uses a dangerous strict equality:
        - ownerSnapshotsCount != 0 && snapshotsOwner[ownerSnapshotsCount - 1].blockNumber == currentBlock (contracts/helper/GovernancePowerDelegationERC20.sol#280-281)
StakedTokenBptRev2.getNextCooldownTimestamp(uint256,uint256,address,uint256) (contracts/StakedTokenBptRev2.sol#306-338) uses a dangerous strict equality:
        - toCooldownTimestamp == 0 (contracts/StakedTokenBptRev2.sol#313)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

StakedTokenBptRev2.constructor(IERC20,IERC20,uint256,uint256,address,address,uint128,string,string,uint8,address).name (contracts/StakedTokenBptRev2.sol#89) shadows:
        - ERC20.name() (contracts/helper/ERC20.sol#65-67) (function)
StakedTokenBptRev2.constructor(IERC20,IERC20,uint256,uint256,address,address,uint128,string,string,uint8,address).symbol (contracts/StakedTokenBptRev2.sol#90) shadows:
        - ERC20.symbol() (contracts/helper/ERC20.sol#73-75) (function)
StakedTokenBptRev2.constructor(IERC20,IERC20,uint256,uint256,address,address,uint128,string,string,uint8,address).decimals (contracts/StakedTokenBptRev2.sol#91) shadows:
        - ERC20.decimals() (contracts/helper/ERC20.sol#90-92) (function)
StakedTokenBptRev2.initialize(string,string,uint8,address).name (contracts/StakedTokenBptRev2.sol#107) shadows:
        - ERC20.name() (contracts/helper/ERC20.sol#65-67) (function)
StakedTokenBptRev2.initialize(string,string,uint8,address).symbol (contracts/StakedTokenBptRev2.sol#108) shadows:
        - ERC20.symbol() (contracts/helper/ERC20.sol#73-75) (function)
StakedTokenBptRev2.initialize(string,string,uint8,address).decimals (contracts/StakedTokenBptRev2.sol#109) shadows:
        - ERC20.decimals() (contracts/helper/ERC20.sol#90-92) (function)
StakedTokenBptRev2.initialize(string,string,uint8,address)._trustedForwarder (contracts/StakedTokenBptRev2.sol#110) shadows:
        - ERC2771Context._trustedForwarder (contracts/helper/ERC2771Context.sol#11) (state variable)
ERC20.constructor(string,string).name (contracts/helper/ERC20.sol#56) shadows:
        - ERC20.name() (contracts/helper/ERC20.sol#65-67) (function)
ERC20.constructor(string,string).symbol (contracts/helper/ERC20.sol#56) shadows:
        - ERC20.symbol() (contracts/helper/ERC20.sol#73-75) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
```

```
StakedTokenV3.constructor(IERC20,IERC20,uint256,uint256,address,address,uint128,string,string,uint8,address).name (contracts/StakedTokenV3.sol#89) shadows:
        - ERC20.name() (contracts/helper/ERC20.sol#65-67) (function)
StakedTokenV3.constructor(IERC20,IERC20,uint256,uint256,address,address,uint128,string,string,uint8,address).symbol (contracts/StakedTokenV3.sol#90) shadows:
        - ERC20.symbol() (contracts/helper/ERC20.sol#73-75) (function)
StakedTokenV3.constructor(IERC20,IERC20,uint256,uint256,address,address,uint128,string,string,uint8,address).decimals (contracts/StakedTokenV3.sol#91) shadows:
        - ERC20.decimals() (contracts/helper/ERC20.sol#90-92) (function)
StakedTokenV3.initialize(string,string,uint8,address).name (contracts/StakedTokenV3.sol#107) shadows:
        - ERC20.name() (contracts/helper/ERC20.sol#65-67) (function)
StakedTokenV3.initialize(string,string,uint8,address).symbol (contracts/StakedTokenV3.sol#108) shadows:
        - ERC20.symbol() (contracts/helper/ERC20.sol#73-75) (function)
StakedTokenV3.initialize(string,string,uint8,address).decimals (contracts/StakedTokenV3.sol#109) shadows:
        - ERC20.decimals() (contracts/helper/ERC20.sol#90-92) (function)
StakedTokenV3.initialize(string,string,uint8,address)._trustedForwarder (contracts/StakedTokenV3.sol#110) shadows:
        - ERC2771Context._trustedForwarder (contracts/helper/ERC2771Context.sol#11) (state variable)
ERC20.constructor(string,string).name (contracts/helper/ERC20.sol#56) shadows:
        - ERC20.name() (contracts/helper/ERC20.sol#65-67) (function)
ERC20.constructor(string,string).symbol (contracts/helper/ERC20.sol#56) shadows:
        - ERC20.symbol() (contracts/helper/ERC20.sol#73-75) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

ERC2771Context.initializeTrustedForwarder(address).trustedForwarder (contracts/helper/ERC2771Context.sol#13) lacks a zero-check on :
                - _trustedForwarder = trustedForwarder (contracts/helper/ERC2771Context.sol#14)
AaveDistributionManager.constructor(address,uint256).emissionManager (contracts/AaveDistributionManager.sol#36) lacks a zero-check on :
                - EMISSION_MANAGER = emissionManager (contracts/AaveDistributionManager.sol#38)
StakedTokenV3.constructor(IERC20,IERC20,uint256,uint256,address,address,uint128,string,string,uint8,address).rewardsVault (contracts/StakedTokenV3.sol#86) lacks a zero-check on :
                - REWARDS_VAULT = rewardsVault (contracts/StakedTokenV3.sol#98)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in StakedTokenV3.claimRewards(address,uint256) (contracts/StakedTokenV3.sol#218-228):
        External calls:
        - REWARD_TOKEN.safeTransferFrom(REWARDS_VAULT,to,amountToClaim) (contracts/StakedTokenV3.sol#225)
        Event emitted after the call(s):
        - RewardsClaimed(_msgSender(),to,amountToClaim) (contracts/StakedTokenV3.sol#227)
Reentrancy in StakedTokenV3.redeem(address,uint256) (contracts/StakedTokenV3.sol#172-199):
        External calls:
        - IERC20(STAKED_TOKEN).safeTransfer(to,amountToRedeem) (contracts/StakedTokenV3.sol#196)
        Event emitted after the call(s):
        - Redeem(_msgSender(),to,amountToRedeem) (contracts/StakedTokenV3.sol#198)
Reentrancy in StakedTokenV3.stake(address,uint256) (contracts/StakedTokenV3.sol#148-165):
        External calls:
        - IERC20(STAKED_TOKEN).safeTransferFrom(_msgSender(),address(this),amount) (contracts/StakedTokenV3.sol#162)
        Event emitted after the call(s):
        - Staked(_msgSender(),onBehalfOf,amount) (contracts/StakedTokenV3.sol#164)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

AaveDistributionManager._updateAssetStateInternal(address,AaveDistributionManager.AssetData,uint256) (contracts/AaveDistributionManager.sol#76-99) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp == lastUpdateTimestamp (contracts/AaveDistributionManager.sol#84)
        - newIndex != oldIndex (contracts/AaveDistributionManager.sol#91)
AaveDistributionManager._updateUserAssetInternal(address,address,uint256,uint256) (contracts/AaveDistributionManager.sol#109-131) uses timestamp for comparisons
        Dangerous comparisons:
```

As a result of the tests carried out with the Slither tool, some results were obtained and these results were reviewed by Halborn. Based on the results reviewed, some vulnerabilities were determined to be false positives and these results were not included in the report. The actual vulnerabilities found by Slither are already included in the report findings.

THANK YOU FOR CHOOSING

HALBORN