HALBORN

Bware Labs - Staking Protocol

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **April 24th, 2023 - May 8th, 2023**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	05/01/2023	Miguel Jalon
0.2	Document Edits	05/01/2023	Miguel Jalon
0.3	Document Edits	05/03/2023	Miguel Jalon
0.4	Draft Version	05/08/2023	Miguel Jalon
0.5	Draft Review	05/08/2023	Grzegorz Trawinski
0.6	Draft Review	05/08/2023	Piotr Cielas
0.7	Draft Review	05/10/2023	Gabi Urrutia
1.0	Remediation Plan	05/17/2023	Grzegorz Trawinski
1.1	Remediation Plan Review	05/17/2023	Ataberk Yavuzer
1.2	Remediation Plan Review	05/17/2023	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
Grzegorz Trawinski	Halborn	Grzegorz.Trawinski@halborn.com
Ataberk Yavuzer	Halborn	Ataberk.Yavuzer@halborn.com
Miguel Jalon	Halborn	Miguel.Jalon@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Bware Labs aims to tackle Web3 challenges and boost global adoption by offering the industry's highest-performance and most reliable infrastructure services and development tools.

Staking protocol allows users to stake tokens in the protocol through delegations to contribute along with the node operators to gain rewards without the need to run a validator or a node themselves.

Bware Labs engaged Halborn to conduct a security audit on their smart contracts beginning on April 24th, 2023 and ending on May 8th, 2023. The security assessment was scoped to the smart contracts provided to the Halborn team.

On May 12th, 2023, the Bware Labs team confirmed that no more code updates will be provided, and the report is ready for finalization.

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 acknowledged by the Bware Labs team.

1.3 SCOPE

1. Staking Protocol - Bware Labs

- Repository: Bware
- First Commit ID (initial commit of the audit): 64aab3d98dc6f82dd886925272d7fa9306bc34c8
- Second Commit ID (including the Saturation Cap Remediation): f067af42fd0b02a057e8714211af6a1d05cb7de8
- Smart Contracts in scope:

    1. Staking.sol
    2. StakingStorage.sol
    3. Finalizer.sol
    4. DelegatorRewardsAdapter.sol
    5. StakingEpochManager.sol
    6. DelegationsAdapter.sol
    7. DepositsAdapter.sol
    8. TransfersAdapter.sol
    9. StakingConstants.sol
    10. Configurable.sol

1.4 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of this audit. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of the contracts' solidity code 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.
- Manual assessment of use and safety for the critical Solidity variables and functions in scope to identify any arithmetic related vulnerability classes.
- Manual testing with custom scripts. (Foundry).
- Static Analysis of security for scoped contract, and imported functions manually.
- Scanning of solidity files for vulnerabilities, security hotspots or bugs. (MythX)
- Static Analysis of security for scoped contract, and imported functions. (Slither)
- Testnet deployment (Anvil).

2. 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.

2.1 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:

Exploitability 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
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:

$$E = \prod m_e$$

2.2 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)	0
	Low (I:L)	0.25
	Medium (I:M)	0.5
	High (I:H)	0.75
	Critical (I:C)	1
Integrity (I)	None (I:N)	0
	Low (I:L)	0.25
	Medium (I:M)	0.5
	High (I:H)	0.75
	Critical (I:C)	1
Availability (A)	None (A:N)	0
	Low (A:L)	0.25
	Medium (A:M)	0.5
	High (A:H)	0.75
	Critical	1
Deposit (D)	None (D:N)	0
	Low (D:L)	0.25
	Medium (D:M)	0.5
	High (D:H)	0.75
	Critical (D:C)	1
Yield (Y)	None (Y:N)	0
	Low (Y:L)	0.25
	Medium: (Y:M)	0.5
	High: (Y:H)	0.75
	Critical (Y:H)	1

Impact I is calculated using the following formula:

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

2.3 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.

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

3. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	0	2	2

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
UNUSABLE STAKING POOL AFTER POOL OWNER UNSTAKE	Low (2.5)	NOT APPLICABLE
NOT ALL ROLES ARE SET UP ON INITIALIZATION	Low (2.5)	RISK ACCEPTED
USE CUSTOM ERRORS TO SAVE GAS	Informational (0.0)	ACKNOWLEDGED
INEFFICIENT CONDITION IN THE FINALIZEPOOL INTERNAL FUNCTION	Informational (0.0)	ACKNOWLEDGED

FINDINGS & TECH DETAILS

4.1 (HAL-01) UNUSABLE STAKING POOL AFTER POOL OWNER UNSTAKE - LOW (2.5)

Description:

A staking pool cannot be reused once the pool owner unstakes their tokens from the pool. That means, if a poolOwner removes all their stake from the node operator and the pool is set to TERMINATED, if the pool owner wants to stake again in the same pool with the same address, nodeIndex and chainId, they cannot do it.

Code Location:

Code Section - DelegationsAdapter.sol#L128

Listing 1: DelegationsAdapter.sol (Line 128)

```
126     function _stake(bytes32 poolId, address delegator, Currency[]
 ↳ memory currencies) internal {
127         validateCurrencies(currencies);
128         require(poolOperative(poolId), "E104");
129         // regular delegators can only stake main token
130         require(
131             (currencies.length == 1 && currencies[0].erc20 ==
 ↳ rewardsToken()) || isPoolOwner(delegator, poolId),
132             "E105"
133         );
134 
135         _updateDelegation(poolId, delegator, currencies, true);
136 
137         assert(poolOperative(poolId));
138     }
```

Code Section - StakingStorage.sol#L116

Listing 2: StakingStorage.sol (Line 116)

```
114     function poolOperative(bytes32 poolId) public view returns (
 ↳ bool) {
115         Pool storage poolPtr = _poolsArchive[poolId];
116         return poolExists(poolId) && poolPtr.delegations[poolPtr.
 ↳ meta.owner].deposited.nextEpochBalance != 0;
117     }
```

Proof Of Concept:

Listing 3: HalbornTest.t.sol

```
 1     // HAL01
 2     function test_IT_01_poolOwner_01() public {
 3         bytes32 poolId = _setUpSimpleEnv(MIN_POOL_OWNER_STAKE,
 ↳ alice);
 4
 5         uint256 times = 5;
 6         _basicNextEpoch(times, poolId);
 7
 8         vm.startPrank(alice);
 9         stakingContract.undelegate(poolId, currencies);
10         vm.stopPrank();
11
12         _basicNextEpoch(times, poolId);
13         _delegate(alice, poolId, 10 ether, address(aToken));
14     }
15
16     // HELPER FUNCTIONS
17     function _setUpSimpleEnv(uint256 poolOwnerStake, address
 ↳ poolOwner) internal returns (bytes32 poolId){
18         vm.startPrank(owner);
19         aToken.transfer(poolOwner, poolOwnerStake);
20         vm.stopPrank();
21
22         vm.startPrank(poolOwner);
23         aToken.approve(address(stakingContract), poolOwnerStake);
24
25         aCurrency.erc20 = IERC20Upgradeable(address(aToken));
26         aCurrency.amount = MIN_POOL_OWNER_STAKE;
```

```
27          currencies.push(aCurrency);
28
29          stakingContract.createNewPool(0, 0, uint16(
↳ MIN_POOL_COMMISSION_FEE), currencies); //currencies
30
31          poolId = bytes32(abi.encodePacked(poolOwner, uint64(0),
↳ uint32(0)));
32          vm.stopPrank();
33      }
34
35      function _basicNextEpoch(uint256 times, bytes32 poolId)
↳ internal {
36          vm.startPrank(owner);
37          bwr.approve(address(stakingContract), 100000 ether);
38          stakingContract.depositRewards(100000 ether);
39          vm.warp(block.timestamp + 1 days);
40          poolIds.push(poolId);
41          performances.push(1000);
42          ownersBaseApy.push(0);
43          vm.stopPrank();
44
45          vm.startPrank(finalizer);
46          for (uint256 i; i < times; ++i){
47              vm.warp(block.timestamp + 1 days);
48              stakingContract.finalizePools(poolIds, performances,
↳ ownersBaseApy);
49              vm.warp(block.timestamp + 1 days);
50              stakingContract.endEpoch();
51          }
52          vm.stopPrank();
53          poolIds.pop();
54          performances.pop();
55          ownersBaseApy.pop();
56      }
57
58      function _delegate(address user, bytes32 poolId, uint256
↳ amount, address token) internal {
59          currencies.pop();
60          vm.startPrank(owner);
61          IERC20Upgradeable(token).transfer(user, amount);
62          vm.stopPrank();
63
64          vm.startPrank(user);
```

```
65          IERC20Upgradeable(token).approve(address(stakingContract),
↳  amount);
66
67          bCurrency.erc20 = IERC20Upgradeable(token);
68          bCurrency.amount = amount;
69          currencies.push(bCurrency);
70
71          stakingContract.delegate(poolId, currencies);
72
73          vm.stopPrank();
74      }
```

```
├─ [0] VM::startPrank(alice: [0x61A1D7fD8C9bbd82932D99DFD47bD2581C23b08c])
│   └─ ← ()
├─ [22529] MockERC20::approve(Staking: [0x01cb6c0007A0d23cC00c5A448Ce4452502b73aC9], 10000000000000000000)
│   ├─ emit Approval(owner: alice: [0x61A1D7fD8C9bbd82932D99DFD47bD2581C23b08c], spender: Staking: [0x01cb6c0007A0d23cC00
│   └─ ← true
├─ [10136] Staking::delegate(0x61a1d7fd8c9bbd82932d99dfd47bd2581c23b08c0000000000000000000000000000, [(0x2efD0Bc8Fc1050Ed24f5
│   ├─ [585] MockERC20::balanceOf(Staking: [0x01cb6c0007A0d23cC00c5A448Ce4452502b73aC9]) [staticcall]
│   │   └─ ← 30000000000000000000
│   ├─ [4659] MockERC20::transferFrom(alice: [0x61A1D7fD8C9bbd82932D99DFD47bD2581C23b08c], Staking: [0x01cb6c0007A0d23cC0
│   │   ├─ emit Approval(owner: alice: [0x61A1D7fD8C9bbd82932D99DFD47bD2581C23b08c], spender: Staking: [0x01cb6c0007A0d23
│   │   ├─ emit Transfer(from: alice: [0x61A1D7fD8C9bbd82932D99DFD47bD2581C23b08c], to: Staking: [0x01cb6c0007A0d23cC00c5
│   │   └─ ← true
│   ├─ [585] MockERC20::balanceOf(Staking: [0x01cb6c0007A0d23cC00c5A448Ce4452502b73aC9]) [staticcall]
│   │   └─ ← 30100000000000000000000
│   └─ ← "E104"
└─ ← "E104"

Test result: FAILED. 0 passed; 1 failed; finished in 3.57ms

Failing tests:
Encountered 1 failing test in test/MyTest2.t.sol:MyTest2
[FAIL. Reason: E104] test_IT_01_poolOwner_01() (gas: 1527671)
```

BVSS:

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

Recommendation:

It is recommended to consider allowing the poolOwner to change the state of the pool from TERMINATED to OPERATIVE on the restake operation.

Remediation Plan:

NOT APPLICABLE: The Bware Labs stated this is a design choice. Whether the poolOwner removes all their stake from the node operator, the pool is set to TERMINATED state, which is a desired behavior.

4.2 (HAL-02) NOT ALL ROLES ARE SET UP ON INITIALIZATION - LOW (2.5)

Description:

In the Staking.sol contract, the initialize() function does not call the setupRole() functions for all available roles when initializing the contract. The PAUSER_ROLE, JAILER_ROLE, FINALIZER_ROLE roles need to be set with the _setupRole() function manually by the account assigned the OWNER_ROLE.

Code Location:

Code Section - Staking.sol#L81

Listing 4: Staking.sol (Lines 89-92)

```
81 function initialize(
82         address protocolOwner,
83         IERC20Upgradeable rewardsToken_,
84         uint256 minEpochDuration
85     ) public initializer {
86         __StakingEpochManager_init_unchained(minEpochDuration);
87 
88         _setupRole(OWNER_ROLE, protocolOwner);
89         _setRoleAdmin(OWNER_ROLE, OWNER_ROLE);
90         _setRoleAdmin(PAUSER_ROLE, OWNER_ROLE);
91         _setRoleAdmin(JAILER_ROLE, OWNER_ROLE);
92         _setRoleAdmin(FINALIZER_ROLE, OWNER_ROLE);
93 
94         _initializeRewardsToken(rewardsToken_);
95 
96         // pause protocol until entirely configured
97         _pause();
98     }
```

BVSS:

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

Recommendation:

Set up all available roles in the contract on initialization.

Remediation Plan:

RISK ACCEPTED: The Bware Labs team plans to set roles manually by the multi-signature account, that has OWNER_ROLE role, after the deployment.

4.3 (HAL-03) USE CUSTOM ERRORS TO SAVE GAS - INFORMATIONAL (0.0)

Description:

Custom errors are available from Solidity version 0.8.4. Custom errors save ~50 gas each time they are hit by avoiding having to allocate and store the revert string. Not defining strings also saves deployment gas.

Code Location:

In the following contracts, the errors can be customized:

- Staking.sol
- DelegationsAdapter.sol
- TransfersAdapter.sol
- StakingEpochManager.sol
- Finalizer.sol

BVSS:

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

Recommendation:

Consider replacing all revert strings with custom errors.

Remediation Plan:

ACKNOWLEDGED: The Bware Labs acknowledged this finding.

4.4 (HAL-04) INEFFICIENT CONDITION IN THE FINALIZEPOOL INTERNAL FUNCTION - INFORMATIONAL (0.0)

Description:

When a pool is jailed, as the baseReward and the poolReward is equal to zero after the code in line 179 executes, the arithmetic operations performed in lines 181 and 182 can be skipped and the poolReward and baseReward variables can be immediately set to 0.

Code Location:

Code Section - Finalizer.sol#175

Listing 5: Finalizer.sol (Lines 175,176)

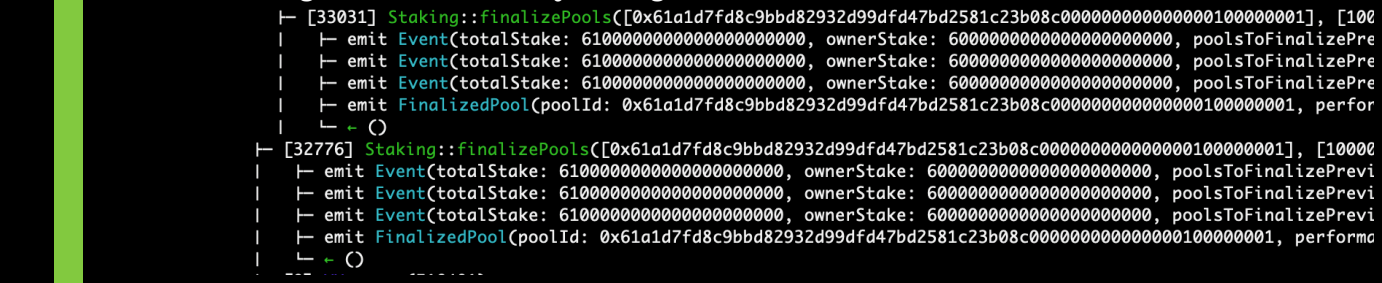
```
170         if (poolPtr.meta.jailed) {
171             poolReward = 0;
172             baseReward = 0;
173         }
174         // apply slashing on all reward types
175         poolReward = (poolReward * performance) /
 ↳ PERCENTAGE_SCALING_FACTOR;
176         baseReward = (baseReward * performance) /
 ↳ PERCENTAGE_SCALING_FACTOR;
```

BVSS:

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

Proof of Concept:

As you can see in the following images, the function _finalizePool costs 300 gas units less by using the else condition as mentioned below.



Recommendation:

Consider adding an `else` condition after the if statement that encapsulates these lines.

Listing 6: Finalizer.sol (Line 175)

```
170         if (poolPtr.meta.jailed) {
171             poolReward = 0;
172             baseReward = 0;
173         }
174         // apply slashing on all reward types
175         else {
176             poolReward = (poolReward * performance) /
 ↳ PERCENTAGE_SCALING_FACTOR;
177             baseReward = (baseReward * performance) /
 ↳ PERCENTAGE_SCALING_FACTOR;
178         }
```

Remediation Plan:

ACKNOWLEDGED: The Bware Labs acknowledged this finding.

RECOMMENDATIONS OVERVIEW

RECOMMENDATIONS OVERVIEW

1. It is recommended that the poolOwner can still change the state of the pool from TERMINATED to OPERATIVE if he stakes again.
2. Set all the needed roles within the contract during initialization.
3. Consider replacing all revert strings with custom errors.
4. Consider adding an else condition after the if statement that encapsulates these lines.

RE-TEST

The issue described in this section was brought to Halborn's attention by the Bware Labs team during the engagement.

6.1 BWARE01 - THERE IS NO CAP FOR THE POOL OWNERS STAKE

Description:

Please note, this finding was brought to Halborn's attention by the Bware Labs team.

The protocol has a saturation cap, which limits the amounts of rewards distributed per time unit over the entire protocol. However, a cap for the pool owner's stake, when computing its additional rewards from the base API, is missing.

Code Location:

Code Section - Finalizer.sol#165

Listing 7: Finalizer.sol (Line 165)

```
143
144     function _finalizePool(bytes32 poolId, uint16 performance,
 ↳ uint16 ownerBaseApy) private {
145         require(performance <= PERCENTAGE_SCALING_FACTOR &&
 ↳ ownerBaseApy <= MAX_OWNER_BASE_APY, "E401");
146
147         uint64 currentEpoch = currentEpoch();
148         CumulativeReward memory cumulativeReward =
 ↳ poolCumulativeReward(poolId, currentEpoch);
149         if (cumulativeReward.finalized) {
150             // pool already finalized for the previous epoch
151             return;
152         }
153
154         Pool storage poolPtr = _poolsArchive[poolId];
155         uint256 totalStake = _previousEpochBalance(poolPtr.meta.
```

```
 ↳ deposited);
156         uint256 ownerStake = _previousEpochBalance(poolPtr.
 ↳ delegations[poolPtr.meta.owner].deposited);
157         if (ownerStake == 0) {
158             // pool not operative at the previous epoch
159             return;
160         }
161 
162         // compute pool`s total rewards earned at previous epoch
163         uint256 poolReward = computePoolEpochRewards(ownerStake,
 ↳ totalStake);
164 
165         // compute owner's additional rewards from its supplied
 ↳ base APY
166         uint256 baseReward = (((ownerStake * ownerBaseApy) /
 ↳ PERCENTAGE_SCALING_FACTOR) * previousEpochDuration()) /
167             ONE_YEAR_SECS;
168 
169         // apply jailing on all reward types
170         if (poolPtr.meta.jailed) {
171             poolReward = 0;
172             baseReward = 0;
173         }
174         // apply slashing on all reward types
175         poolReward = (poolReward * performance) /
 ↳ PERCENTAGE_SCALING_FACTOR;
176         baseReward = (baseReward * performance) /
 ↳ PERCENTAGE_SCALING_FACTOR;
177 
178         // reserve the distributed rewards from the total rewards
 ↳ fund
179         require(_depositedRewardsFund >= (poolReward + baseReward)
 ↳ , "E205");
180         _depositedRewardsFund -= (poolReward + baseReward);
181 
182         // compute owner's commission rewards
183         uint256 commissionReward = (poolReward * poolPtr.meta.
 ↳ commissionFee) / PERCENTAGE_SCALING_FACTOR;
184 
185         // compute pool's cumulative reward up to this epoch
186         cumulativeReward.finalized = true;
187         cumulativeReward.value = poolCumulativeReward(poolId,
 ↳ currentEpoch - 1).value;
188         cumulativeReward.value += SafeCastUpgradeable.toUint248(
```

```
189             ((poolReward - commissionReward) *
 ↳ TOKEN_FRACTION_SCALING_FACTOR) / totalStake
190         );
191
192         // set cumulative reward at this epoch and implicitly mark
 ↳  the pool as finalized
193         _poolCumulativeReward[poolId][currentEpoch] =
 ↳ cumulativeReward;
194
195         // reward the pool owner its commission + base rewards
196         _unclaimedRewards[poolPtr.meta.owner] += (baseReward +
 ↳ commissionReward);
197
198         // remove this pool from the finalization accounting of
 ↳ previous epoch
199         _poolsToFinalizePreviousEpoch -= 1;
200
201         emit FinalizedPool(poolId, performance, ownerBaseApy,
 ↳ cumulativeReward, poolReward, baseReward);
202     }
```

Recommendation:

It is recommended to set a cap to the owner's stake when computing its additional rewards from the base API.

Remediation Plan:

SOLVED: The Bware Labs team found this issue and solved it by capping the owner's stake:

f067af42fd0b02a057e8714211af6a1d05cb7de8::Finalizer.sol#L165

MANUAL TESTING

7.1 TESTING METHODOLOGY

The tests for the Bware Labs audit were focused on three different parts:

1. UNIT TESTING
    - Halborn wrote unit tests for all the components in scope has to identify corner cases independent of the rest of the protocol.
2. INTEGRATION TESTING
    - Halborn performed integration testing of the components in scope, combining different use cases and interactions of different kinds of users to test if the output is as expected and the functions work correctly.
3. EDGE CASE TESTING
    - Halborn performed edge cases tests, trying to force the protocol to corner casuistic that could make the protocol behave incorrectly.

7.2 TESTS

Unit testing of all the components involved in the protocol:

Halborn wrote unit tests for all the components in scope has to identify corner cases independent of the rest of the protocol.

As an example, it was tracked how re-delegate is correctly executed or how a pool can be jailed by the jailer and not by any other user.

UNIT TEST AROUND ENDING EPOCH

- Alice creates a pool
- Bobby delegates to Alice's pool
- Owner ends the epoch

```
Running 1 test for test/MyTest.t.sol:MyTest
[PASS] test_EndEpoch() (gas: 504283)
Logs:
  Current Epoch 1
  EXECUTING TX ------> 'stakingContract.endEpoch()'
  Current Epoch 2
```

Integration Testing of use cases integrating all the stakeholders:

Halborn performed integration testing of the components in scope, combining different use cases and interactions of different kinds of users to test if the output is as expected and the functions work correctly.

INTEGRATION TEST 1:

Within this use case, a pool's owner makes a deposit and then withdraws all assets.

1. Two different pools are created with different `nodeIds` from two different users.
2. We wait a couple of epochs while they have 100% of performance.
3. The pool owner unstake his tokens from the pool.

4. It has to be correctly terminated, and the epochs should end correctly without finalization.

```
Running 1 test for test/MyTest.t.sol:MyTest
[PASS] test_IT_01_FinalizingPoolsAfterUnstake() (gas: 1529179)
Logs:
  bwr token ---->  0
  x token ------>  0
  y token ------>  0
  bwr token ---->  0
  x token ------>  0
  y token ------>  0
  bwr token ---->  0
  x token ------>  0
  y token ------>  0
  bwr token ---->  107468071654373021
  x token ------>  300000000000000000000
  y token ------>  300000000000000000000
```

INTEGRATION TEST 2:

Within this use case, multiple users participate in staking and reward collection. Eventually, withdraw all assets.

1. Alice creates a pool with 2 tokens (x and y).
2. Edgar delegates to Alice with the bwr token.
3. The owner deposits the rewards to the protocol.
4. We ran 2 epochs.
5. Edgar wants to claim, but he cannot yet.
6. We run more epochs.
7. Edgar can claim.
8. Edgar wants to claim again, but there are no more rewards for him.
9. We run more epochs.
10. Edgar claimed 2 times, again only the first worked.
11. We run more epochs.
12. Alice undelegates.
13. We run more epochs.
14. Edgar can claim.
15. We run more epochs.
16. Edgar undelegates all his stake.
17. Edgar wants to undelegate one more time -----> reverts as expected.
18. Edgar cannot claim anymore (no more rewards as the pool is terminated).
19. Alice and Edgar can get all their rewards + recover stake.

```
Running 1 test for test/MyTest.t.sol:MyTest
[PASS] test_IT_02() (gas: 1997224)
Logs:
   ----------------
   --- STAGE 0 ----
   ----------------
  Alice X token:    150000000000000000000
  Alice T token:    150000000000000000000
  Alice BWR token:  0
  Edgar X token:    150000000000000000000
  Edgar Y token:    150000000000000000000
  Edgar BWR token:  50000000000000000000
   ----------------
   --- STAGE 1 ----
   ----------------
  Alice X token:    120000000000000000000
  Alice T token:    120000000000000000000
  Alice BWR token:  0
  Edgar X token:    150000000000000000000
  Edgar Y token:    150000000000000000000
  Edgar BWR token:  0
  pool terminated?  -   true
   ----------------
   --- STAGE 2 ----
   ----------------
  Alice X token:    150000000000000000000
  Alice T token:    150000000000000000000
  Alice BWR token:  573487881981032659
  Edgar X token:    150000000000000000000
  Edgar Y token:    150000000000000000000
  Edgar BWR token:  50000000000000000000
```

INTEGRATION TEST 3:

Within this use, the pool creations and delegations were executed.

1. Alice creates 2 different pools: (nodeIndex: 0, chainId: 1) and (nodeIndex: 1, chainId: 1).
2. Edgar creates 2 different pools: (nodeIndex: 1, chainId: 2) and (nodeIndex: 1, chainId: 1).
3. Bobby delegates to some pools.
4. The owner deposits the rewards.
5. We ran a couple of epochs.
6. Alice creates again the same pool (1,1) --> Revert, expected.
7. Alice undelegates.
8. We ran a couple of epochs.
9. We track everyone has the correct rewards.

```
Running 1 test for test/MyTest.t.sol:MyTest
[PASS] test_tt2() (gas: 3168345)
Logs:
  Alice X token:    300000000000000000000
  Alice T token:    300000000000000000000
  Alice BWR token:  852515068493150674
  Edgar X token:    300000000000000000000
  Edgar Y token:    300000000000000000000
  Edgar BWR token:  845285563751317167
  Bobby X token:    0
  Bobby Y token:    0
  Bobby BWR token:  190000000000000000000
```

Edge Case Scenario:

A malicious pool owner tries to create a pool which the protocol will not be able to finalize. Thus, leading to a Denial of Service (DoS) of the overall protocol because of an integer overflow. An attempt to set `0` for the `ownerStake` variable. Also, an attempt to turn `_depositedRewardsFund`, `poolReward`, `baseReward` and `totalStake` variables into incorrect state.

```
    ├─ [26879] Staking::depositRewards(1000000000000000000)
    |   ├─ [585] MockERC20::balanceOf(Staking: [0x96625080A005894504f342a187b1421F26C6aB0D]) [staticcall]
    |   |   └─ ← 1000000000000000000
    |   ├─ [4659] MockERC20::transferFrom(owner: [0xb6587E55021da8b935C7C66E0c03C04E7b831D4F], Staking: [0x9662
    |   |   ├─ emit Approval(owner: owner: [0xb6587E55021da8b935C7C66E0c03C04E7b831D4F], spender: Staking: [0x9
    |   |   ├─ emit Transfer(from: owner: [0xb6587E55021da8b935C7C66E0c03C04E7b831D4F], to: Staking: [0x9662508
    |   |   └─ ← true
    |   ├─ [585] MockERC20::balanceOf(Staking: [0x96625080A005894504f342a187b1421F26C6aB0D]) [staticcall]
    |   |   └─ ← 1100000000000000000
    |   ├─ emit DepositRewards(amount: 100000000000000000)
    |   └─ ← ()
    ├─ [23342] Staking::endEpoch()
    |   ├─ emit EndEpoch()
    |   └─ ← ()
    ├─ [0] VM::warp(2592001)
    |   └─ ← ()
    ├─ [52140] Staking::finalizePools([0x61a1d7fd8c9bbd82932d99dfd47bd2581c23b08c000000000000000000000001], [10
    |   ├─ emit Event(totalStake: 61000000000000000000, ownerStake: 60000000000000000000, poolsToFinalizePr
    |   ├─ emit Event(totalStake: 61000000000000000000, ownerStake: 60000000000000000000, poolsToFinalizePr
    |   ├─ emit Event(totalStake: 61000000000000000000, ownerStake: 60000000000000000000, poolsToFinalizePr
    |   ├─ emit FinalizedPool(poolId: 0x61a1d7fd8c9bbd82932d99dfd47bd2581c23b08c000000000000000000000001, perfo
    |   └─ ← ()
    └─ ← ()

Test result: ok. 1 passed; 0 failed; finished in 3.48ms
```

OVERVIEW OF THE RESULTS OF THE TESTS PERFORMED:

```
Running 19 tests for test/MyTest.t.sol:MyTest
[PASS] test_CannotFinalizePools() (gas: 812198)
[PASS] test_CreatePool() (gas: 299118)
[PASS] test_DepositRewards() (gas: 104006)
[PASS] test_EndEpoch() (gas: 504326)
[PASS] test_IT_01_FinalizingPoolsAfterUnstake() (gas: 1467773)
[PASS] test_IT_02() (gas: 1997224)
[PASS] test_IT_03_UnstakeTooMuch() (gas: 1386866)
[PASS] test_IT_22_Jailed() (gas: 1548782)
[PASS] test_InfiniteDelegateAttack() (gas: 2469673)
[PASS] test_cannotFinalizePools1() (gas: 638324)
[PASS] test_cannotFinalizePools2() (gas: 638271)
[PASS] test_delegateStakePool() (gas: 438096)
[PASS] test_finalizePools() (gas: 1048540)
[PASS] test_finalizePools2() (gas: 787717)
[PASS] test_finalizePoolsUserRecoveringStake() (gas: 820825)
[PASS] test_poolJailedPoC() (gas: 905965)
[PASS] test_tt() (gas: 1983116)
[PASS] test_tt2() (gas: 3168412)
[PASS] test_undelegateStakePool() (gas: 529633)
Test result: ok. 19 passed; 0 failed; finished in 7.29ms

Running 8 tests for test/MyTest2.t.sol:MyTest2
[PASS] test1() (gas: 42759)
[PASS] test_AT_01() (gas: 16792054384)
[PASS] test_UT_01_poolOwner_01() (gas: 320774)
[PASS] test_UT_01_poolOwner_02() (gas: 42188)
[PASS] test_UT_3_user_01() (gas: 320782)
[PASS] test_UT_Jail_02_owner_01() (gas: 730029)
[PASS] test_UT_Jail_1_poolOwner_03() (gas: 750799)
[PASS] test_UT_finalizer1() (gas: 320819)
Test result: ok. 8 passed; 0 failed; finished in 12.29s
```

AUTOMATED TESTING

8.1 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' ABIs across the entire code-base.

Slither Results:

- Staking.sol

```
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts-
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-upgr
        - inverse = (3 * denominator) ^ 2 (lib/openzeppelin-contracts/contracts-upgra
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts-
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-upgr
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contracts-u
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts-
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-upgr
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contracts-u
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts-
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-upgr
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contracts-u
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts-
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-upgr
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contracts-u
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts-
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-upgr
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contracts-u
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts-
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-upgr
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contracts-u
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts-
        - prod0 = prod0 / twos (lib/openzeppelin-contracts/contracts-upgradeable/utils
        - result = prod0 * inverse (lib/openzeppelin-contracts/contracts-upgradeable/u
Finalizer._finalizePool(bytes32,uint16,uint16) (src/staking/rewards/Finalizer.sol#144-
        - baseReward = (((ownerStake * ownerBaseApy) / PERCENTAGE_SCALING_FACTOR) * pr
Finalizer._finalizePool(bytes32,uint16,uint16) (src/staking/rewards/Finalizer.sol#144-
        - baseReward = (((ownerStake * ownerBaseApy) / PERCENTAGE_SCALING_FACTOR) * pr
        - baseReward = (baseReward * performance) / PERCENTAGE_SCALING_FACTOR (src/sta
Finalizer._finalizePool(bytes32,uint16,uint16) (src/staking/rewards/Finalizer.sol#144-
        - poolReward = (poolReward * performance) / PERCENTAGE_SCALING_FACTOR (src/sta
        - commissionReward = (poolReward * poolPtr.meta.commissionFee) / PERCENTAGE_SC
Finalizer.computePoolEpochRewards(uint256,uint256) (src/staking/rewards/Finalizer.sol#
        - earningStake = (ownerStake * (totalStake - earningStake)) / saturationCap (s
        - earningStake = (earningStake * pledgeFactorNum) / pledgeFactorDen (src/staki
Finalizer.computePoolEpochRewards(uint256,uint256) (src/staking/rewards/Finalizer.sol#
        - earningStake = (earningStake * pledgeFactorNum) / pledgeFactorDen (src/staki
        - earningStake = (earningStake * pledgeFactorDen) / (pledgeFactorDen + pledgeF
Finalizer.computePoolEpochRewards(uint256,uint256) (src/staking/rewards/Finalizer.sol#
        - (((earningStake * getConfig(POOL_EARNING_STAKE_APY)) / PERCENTAGE_SCALING_FA
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before

DelegatorRewardsAdapter._computeDelegatorReward(address,bytes32) (src/staking/rewards/
        - delegation.currentEpoch == 0 (src/staking/rewards/DelegatorRewardsAdapter.so
DelegatorRewardsAdapter._computeDelegatorReward(address,bytes32) (src/staking/rewards/
        - delegation.currentEpoch == currentEpoch (src/staking/rewards/DelegatorRewar
```

```
TransfersAdapter._strictTransferFrom(address,IERC20Upgradeable,uint256) (src/st
        - require(bool,string)(erc20.balanceOf(address(this)) == previousBalanc
StakingStorage.isPoolOwner(address,bytes32) (src/staking/StakingStorage.sol#100
        - _poolsArchive[poolId].meta.owner == account (src/staking/StakingStorc
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#danger

DelegationsAdapter._updateDelegationCurrencies(bytes32,address,StakingStorage.C
DelegationsAdapter.validateCurrencies(StakingStorage.Currency[]).it (src/stakir
Staking.executePendingWithdrawal(bytes32).it (src/staking/Staking.sol#298) is c
DelegatorRewardsAdapter._collectDelegatorRewardsMany(address,bytes32[]).it (src
TransfersAdapter._strictTransferFrom(address,StakingStorage.Currency[]).it (src
Finalizer.finalizePools(bytes32[],uint16[],uint16[]).it (src/staking/rewards/Fi
DelegatorRewardsAdapter.computeDelegatorReward(address,bytes32[]).it (src/staki
Staking._createPendingWithdrawal(bytes32,address,StakingStorage.Currency[]).it
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninit

DepositsAdapter._loadDeposit(DepositsAdapter.DeferredDeposit).currentEpoch (src
        - StakingEpochManager.currentEpoch() (src/staking/epochs/StakingEpochMc
DepositsAdapter._previousEpochBalance(DepositsAdapter.DeferredDeposit).currentE
        - StakingEpochManager.currentEpoch() (src/staking/epochs/StakingEpochMc
DelegatorRewardsAdapter._computeDelegatorRewardOverInterval(bytes32,uint256,uir
        - Finalizer.endEpoch() (src/staking/rewards/Finalizer.sol#106-115) (fur
DelegatorRewardsAdapter._computeDelegatorReward(address,bytes32).currentEpoch (
        - StakingEpochManager.currentEpoch() (src/staking/epochs/StakingEpochMc
Finalizer._finalizePool(bytes32,uint16,uint16).currentEpoch (src/staking/rewarc
        - StakingEpochManager.currentEpoch() (src/staking/epochs/StakingEpochMc
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-

AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (lib/ope
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls

Reentrancy in Staking.createNewPool(uint64,uint32,uint16,StakingStorage.Currenc
        External calls:
        - _strictTransferFrom(_msgSender(),currencies) (src/staking/Staking.sol
                - returndata = address(token).functionCall(data,SafeERC20: low-
                - erc20.safeTransferFrom(account,address(this),amount) (src/sta
                - (success,returndata) = target.call{value: value}(data) (lib/c
        External calls sending eth:
        - _strictTransferFrom(_msgSender(),currencies) (src/staking/Staking.sol
                - (success,returndata) = target.call{value: value}(data) (lib/c
        State variables written after the call(s):
        - _initializePoolOwner(poolId,_msgSender(),currencies) (src/staking/Sta
                - _poolsArchive[poolId].meta.owner = owner (src/staking/deposit
                - delegation.currencies[currency.erc20] += currency.amount (src
                - delegation.currencies[currency.erc20] -= currency.amount (src
        - _setPoolCommission(poolId,commissionFee) (src/staking/Staking.sol#173
                - _poolsArchive[poolId].meta.commissionFee = commissionFee (src
                - _poolsArchive[poolId].meta.commissionLastSet = currentEpoch()
        - _initializePoolOwner(poolId,_msgSender(),currencies) (src/staking/Sta
                - _poolsToFinalizeNextEpoch ++ (src/staking/rewards/Finalizer.s
        - _initializePoolOwner(poolId,_msgSender(),currencies) (src/staking/Sta
                - _unclaimedRewards[delegator] += rewards (src/staking/rewards/
Reentrancy in Staking.delegate(bytes32,StakingStorage.Currency[]) (src/staking/
```

```
        External calls:
        - _strictTransferFrom(_msgSender(),currencies) (src/staking/Staking.sol#210
                - returndata = address(token).functionCall(data,SafeERC20: low-leve
                - erc20.safeTransferFrom(account,address(this),amount) (src/staking
                - (success,returndata) = target.call{value: value}(data) (lib/openz
        External calls sending eth:
        - _strictTransferFrom(_msgSender(),currencies) (src/staking/Staking.sol#210
                - (success,returndata) = target.call{value: value}(data) (lib/openz
        State variables written after the call(s):
        - _stake(poolId,_msgSender(),currencies) (src/staking/Staking.sol#211)
                - delegation.currencies[currency.erc20] += currency.amount (src/sta
                - delegation.currencies[currency.erc20] -= currency.amount (src/sta
        - _stake(poolId,_msgSender(),currencies) (src/staking/Staking.sol#211)
                - _unclaimedRewards[delegator] += rewards (src/staking/rewards/Dele
Reentrancy in Staking.depositRewards(uint256) (src/staking/Staking.sol#144-149):
        External calls:
        - _strictTransferFrom(_msgSender(),rewardsToken(),amount) (src/staking/Stak
                - returndata = address(token).functionCall(data,SafeERC20: low-leve
                - erc20.safeTransferFrom(account,address(this),amount) (src/staking
                - (success,returndata) = target.call{value: value}(data) (lib/openz
        External calls sending eth:
        - _strictTransferFrom(_msgSender(),rewardsToken(),amount) (src/staking/Stak
                - (success,returndata) = target.call{value: value}(data) (lib/openz
        State variables written after the call(s):
        - _depositedRewardsFund += amount (src/staking/Staking.sol#146)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy

Reentrancy in Staking.claim(bytes32[]) (src/staking/Staking.sol#310-319):
        External calls:
        - _strictTransferTo(_msgSender(),rewardsToken(),amount) (src/staking/Stakin
                - returndata = address(token).functionCall(data,SafeERC20: low-leve
                - erc20.safeTransfer(account,amount) (src/staking/deposits/Transfer
                - (success,returndata) = target.call{value: value}(data) (lib/openz
        External calls sending eth:
        - _strictTransferTo(_msgSender(),rewardsToken(),amount) (src/staking/Stakin
                - (success,returndata) = target.call{value: value}(data) (lib/openz
        Event emitted after the call(s):
        - Claim(_msgSender(),amount) (src/staking/Staking.sol#318)
Reentrancy in Staking.createNewPool(uint64,uint32,uint16,StakingStorage.Currency[])
        External calls:
        - _strictTransferFrom(_msgSender(),currencies) (src/staking/Staking.sol#167
                - returndata = address(token).functionCall(data,SafeERC20: low-leve
                - erc20.safeTransferFrom(account,address(this),amount) (src/staking
                - (success,returndata) = target.call{value: value}(data) (lib/openz
        External calls sending eth:
        - _strictTransferFrom(_msgSender(),currencies) (src/staking/Staking.sol#167
                - (success,returndata) = target.call{value: value}(data) (lib/openz
        Event emitted after the call(s):
        - CollectDelegatorRewards(delegator,poolId,rewards) (src/staking/rewards/De
                - _initializePoolOwner(poolId,_msgSender(),currencies) (src/staking
        - CreateNewPool(_msgSender(),poolId,currencies) (src/staking/Staking.sol#17
        - SetPoolCommissionFee(poolId,commissionFee) (src/staking/Staking.sol#201)
                - _setPoolCommission(poolId,commissionFee) (src/staking/Staking.sol
Reentrancy in Staking.delegate(bytes32,StakingStorage.Currency[]) (src/staking/Stak
```

```
        External calls:
        - _strictTransferFrom(_msgSender(),currencies) (src/staking/Staking.sol#21
                - returndata = address(token).functionCall(data,SafeERC20: low-lev
                - erc20.safeTransferFrom(account,address(this),amount) (src/stakin
                - (success,returndata) = target.call{value: value}(data) (lib/open
        External calls sending eth:
        - _strictTransferFrom(_msgSender(),currencies) (src/staking/Staking.sol#21
                - (success,returndata) = target.call{value: value}(data) (lib/open
        Event emitted after the call(s):
        - CollectDelegatorRewards(delegator,poolId,rewards) (src/staking/rewards/D
                - _stake(poolId,_msgSender(),currencies) (src/staking/Staking.sol#
        - Delegate(_msgSender(),poolId,currencies) (src/staking/Staking.sol#213)
Reentrancy in Staking.depositRewards(uint256) (src/staking/Staking.sol#144-149):
        External calls:
        - _strictTransferFrom(_msgSender(),rewardsToken(),amount) (src/staking/Sta
                - returndata = address(token).functionCall(data,SafeERC20: low-lev
                - erc20.safeTransferFrom(account,address(this),amount) (src/stakin
                - (success,returndata) = target.call{value: value}(data) (lib/open
        External calls sending eth:
        - _strictTransferFrom(_msgSender(),rewardsToken(),amount) (src/staking/Sta
                - (success,returndata) = target.call{value: value}(data) (lib/open
        Event emitted after the call(s):
        - DepositRewards(amount) (src/staking/Staking.sol#148)
Reentrancy in Staking.executePendingWithdrawal(bytes32) (src/staking/Staking.sol#2
        External calls:
        - _strictTransferTo(_msgSender(),withdrawal.currencies[it].erc20,withdrawa
                - returndata = address(token).functionCall(data,SafeERC20: low-lev
                - erc20.safeTransfer(account,amount) (src/staking/deposits/Transfe
                - (success,returndata) = target.call{value: value}(data) (lib/open
        External calls sending eth:
        - _strictTransferTo(_msgSender(),withdrawal.currencies[it].erc20,withdrawa
                - (success,returndata) = target.call{value: value}(data) (lib/open
        Event emitted after the call(s):
        - ExecutePendingWithdrawal(_msgSender(),poolId) (src/staking/Staking.sol#3
Reentrancy in Staking.unjail(bytes32) (src/staking/Staking.sol#358-365):
        External calls:
        - depositRewards(getConfig(POOL_UNJAILING_FEE)) (src/staking/Staking.sol#3
                - returndata = address(token).functionCall(data,SafeERC20: low-lev
                - erc20.safeTransferFrom(account,address(this),amount) (src/stakin
                - (success,returndata) = target.call{value: value}(data) (lib/open
        External calls sending eth:
        - depositRewards(getConfig(POOL_UNJAILING_FEE)) (src/staking/Staking.sol#3
                - (success,returndata) = target.call{value: value}(data) (lib/open
        Event emitted after the call(s):
        - UnjailPool(poolId) (src/staking/Staking.sol#364)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentranc
```

```
Staking.setPoolCommission(bytes32,uint16) (src/staking/Staking.sol#182-189) uses ti
        Dangerous comparisons:
        - require(bool,string)(currentEpoch() >= _poolsArchive[poolId].meta.commiss
Staking._createPendingWithdrawal(bytes32,address,StakingStorage.Currency[]) (src/st
        Dangerous comparisons:
        - require(bool,string)(withdrawal.currencies.length == 0,E110) (src/staking
Staking.canRedelegate(address,bytes32) (src/staking/Staking.sol#252-254) uses times
        Dangerous comparisons:
        - (poolTerminated(poolId) || poolJailed(poolId)) && ! isPoolOwner(account,p
Staking.executePendingWithdrawal(bytes32) (src/staking/Staking.sol#290-304) uses ti
        Dangerous comparisons:
        - require(bool,string)(withdrawal.executeTime <= block.timestamp,E113) (src
        - require(bool,string)(! (isPoolOwner(_msgSender(),poolId) && poolJailed(po
StakingEpochManager._endEpoch() (src/staking/epochs/StakingEpochManager.sol#69-76)
        Dangerous comparisons:
        - require(bool,string)(block.timestamp >= getEpochEarliestEndTime(),E301) (
Finalizer.unclaimedRewards(address) (src/staking/rewards/Finalizer.sol#75-81) uses
        Dangerous comparisons:
        - rewards != 0 (src/staking/rewards/Finalizer.sol#78)
Finalizer._finalizePool(bytes32,uint16,uint16) (src/staking/rewards/Finalizer.sol#1
        Dangerous comparisons:
        - require(bool,string)(_depositedRewardsFund >= (poolReward + baseReward),E
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-time

AddressUpgradeable._revert(bytes,string) (lib/openzeppelin-contracts/contracts-upgr
        - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/Addres
StringsUpgradeable.toString(uint256) (lib/openzeppelin-contracts/contracts-upgradea
        - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/String
        - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/String
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/M
        - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/M
        - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/M
StakingStorage.filterPoolIdsIdle(bytes32[]) (src/staking/StakingStorage.sol#185-197
        - INLINE ASM (src/staking/StakingStorage.sol#194-196)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-u

Different versions of Solidity are used:
        - Version used: ['^0.8.0', '^0.8.1', '^0.8.11', '^0.8.2']
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/access/AccessCon
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/access/IAccessCo
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/security/Pausabl
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/token/ERC20/IERC
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/token/ERC20/exte
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/token/ERC20/exte
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/token/ERC20/util
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/ContextUpg
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/StringsUpg
```

```
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/StringsUpg
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/introspect
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/introspect
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/MathU
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/SafeC
        - ^0.8.1 (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpg
        - ^0.8.11 (src/staking/Staking.sol#2)
        - ^0.8.11 (src/staking/StakingStorage.sol#2)
        - ^0.8.11 (src/staking/config/Configurable.sol#2)
        - ^0.8.11 (src/staking/config/StakingConstants.sol#2)
        - ^0.8.11 (src/staking/deposits/DelegationsAdapter.sol#2)
        - ^0.8.11 (src/staking/deposits/DepositsAdapter.sol#2)
        - ^0.8.11 (src/staking/deposits/TransfersAdapter.sol#2)
        - ^0.8.11 (src/staking/epochs/StakingEpochManager.sol#2)
        - ^0.8.11 (src/staking/rewards/DelegatorRewardsAdapter.sol#2)
        - ^0.8.11 (src/staking/rewards/Finalizer.sol#2)
        - ^0.8.2 (lib/openzeppelin-contracts/contracts-upgradeable/proxy/utils/Init
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-

Finalizer._finalizePool(bytes32,uint16,uint16) (src/staking/rewards/Finalizer.sol#1
        - _depositedRewardsFund -= (poolReward + baseReward) (src/staking/rewards/F
Finalizer._finalizePool(bytes32,uint16,uint16) (src/staking/rewards/Finalizer.sol#1
        - _poolsToFinalizePreviousEpoch -= 1 (src/staking/rewards/Finalizer.sol#205
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-ope

AccessControlUpgradeable.__AccessControl_init() (lib/openzeppelin-contracts/contrac
AccessControlUpgradeable.__AccessControl_init_unchained() (lib/openzeppelin-contrac
AddressUpgradeable.functionCall(address,bytes) (lib/openzeppelin-contracts/contract
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (lib/openzeppelin-c
AddressUpgradeable.functionStaticCall(address,bytes) (lib/openzeppelin-contracts/co
AddressUpgradeable.functionStaticCall(address,bytes,string) (lib/openzeppelin-contr
AddressUpgradeable.sendValue(address,uint256) (lib/openzeppelin-contracts/contracts
AddressUpgradeable.verifyCallResult(bool,bytes,string) (lib/openzeppelin-contracts/
ContextUpgradeable.__Context_init() (lib/openzeppelin-contracts/contracts-upgradeab
ContextUpgradeable.__Context_init_unchained() (lib/openzeppelin-contracts/contracts
ContextUpgradeable._msgData() (lib/openzeppelin-contracts/contracts-upgradeable/uti
ERC165Upgradeable.__ERC165_init() (lib/openzeppelin-contracts/contracts-upgradeable/
ERC165Upgradeable.__ERC165_init_unchained() (lib/openzeppelin-contracts/contracts-up
Initializable._disableInitializers() (lib/openzeppelin-contracts/contracts-upgradeab
Initializable._getInitializedVersion() (lib/openzeppelin-contracts/contracts-upgrad
Initializable._isInitializing() (lib/openzeppelin-contracts/contracts-upgradeable/pr
MathUpgradeable.average(uint256,uint256) (lib/openzeppelin-contracts/contracts-upgr
MathUpgradeable.ceilDiv(uint256,uint256) (lib/openzeppelin-contracts/contracts-upgr
MathUpgradeable.log10(uint256) (lib/openzeppelin-contracts/contracts-upgradeable/ut
MathUpgradeable.log10(uint256,MathUpgradeable.Rounding) (lib/openzeppelin-contracts/
MathUpgradeable.log2(uint256) (lib/openzeppelin-contracts/contracts-upgradeable/uti
MathUpgradeable.log2(uint256,MathUpgradeable.Rounding) (lib/openzeppelin-contracts/
```

- StakingStorage.sol

```
StakingStorage._depositedRewardsFund (src/staking/StakingSTORAGE.sol#71) is never i
        - StakingStorage.depositedRewardsFund() (src/staking/StakingSTORAGE.sol#86-
StakingStorage._poolsArchive (src/staking/StakingSTORAGE.sol#74) is never initializ
        - StakingStorage.poolMetadata(bytes32) (src/staking/StakingSTORAGE.sol#93-9
        - StakingStorage.isPoolOwner(address,bytes32) (src/staking/StakingSTORAGE.s
        - StakingStorage.poolExists(bytes32) (src/staking/StakingSTORAGE.sol#107-10
        - StakingStorage.poolOperative(bytes32) (src/staking/StakingSTORAGE.sol#114
        - StakingStorage.poolJailed(bytes32) (src/staking/StakingSTORAGE.sol#129-13
        - StakingStorage.getPoolStake(bytes32) (src/staking/StakingSTORAGE.sol#137-
        - StakingStorage.getDelegatorStake(address,bytes32) (src/staking/StakingSTO
        - StakingStorage.getDelegatorCurrency(address,bytes32,IERC20Upgradeable) (s
        - StakingStorage.getDelegatorPendingWithdrawal(address,bytes32) (src/stakin
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitiali

MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-up
        - inverse = (3 * denominator) ^ 2 (lib/openzeppelin-contracts/contracts-upg
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-up
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contract
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-up
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contract
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-up
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contract
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-up
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contract
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-up
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contract
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-up
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contract
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - prod0 = prod0 / twos (lib/openzeppelin-contracts/contracts-upgradeable/ut
        - result = prod0 * inverse (lib/openzeppelin-contracts/contracts-upgradeabl
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-bef

DepositsAdapter._loadDeposit(DepositsAdapter.DeferredDeposit).currentEpoch (src/sta
        - StakingEpochManager.currentEpoch() (src/staking/epochs/StakingEpochManage
DepositsAdapter._previousEpochBalance(DepositsAdapter.DeferredDeposit).currentEpoch
        - StakingEpochManager.currentEpoch() (src/staking/epochs/StakingEpochManage
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-vari

StakingEpochManager._endEpoch() (src/staking/epochs/StakingEpochManager.sol#69-76)
        Dangerous comparisons:
        - require(bool,string)(block.timestamp >= getEpochEarliestEndTime(),E301) (
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-time
```

```
AddressUpgradeable._revert(bytes,string) (lib/openzeppelin-contracts/contracts-upg
        - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/Addre
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contra
        - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/
        - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/
        - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/
StakingStorage.filterPoolIdsIdle(bytes32[]) (src/staking/StakingSTORAGE.sol#185-19
        - INLINE ASM (src/staking/StakingSTORAGE.sol#194-196)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-

Different versions of Solidity are used:
        - Version used: ['^0.8.0', '^0.8.1', '^0.8.11', '^0.8.2']
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/token/ERC20/IER
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/Math
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/Safe
        - ^0.8.1 (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUp
        - ^0.8.11 (src/staking/StakingSTORAGE.sol#2)
        - ^0.8.11 (src/staking/config/Configurable.sol#2)
        - ^0.8.11 (src/staking/config/StakingConstants.sol#2)
        - ^0.8.11 (src/staking/deposits/DepositsAdapter.sol#2)
        - ^0.8.11 (src/staking/epochs/StakingEpochManager.sol#2)
        - ^0.8.2 (lib/openzeppelin-contracts/contracts-upgradeable/proxy/utils/Ini
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different

AddressUpgradeable._revert(bytes,string) (lib/openzeppelin-contracts/contracts-upg
AddressUpgradeable.functionCall(address,bytes) (lib/openzeppelin-contracts/contrac
AddressUpgradeable.functionCall(address,bytes,string) (lib/openzeppelin-contracts/
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (lib/openzeppelin-
AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (lib/openze
AddressUpgradeable.functionStaticCall(address,bytes) (lib/openzeppelin-contracts/c
AddressUpgradeable.functionStaticCall(address,bytes,string) (lib/openzeppelin-cont
AddressUpgradeable.isContract(address) (lib/openzeppelin-contracts/contracts-upgra
AddressUpgradeable.sendValue(address,uint256) (lib/openzeppelin-contracts/contract
AddressUpgradeable.verifyCallResult(bool,bytes,string) (lib/openzeppelin-contracts.
AddressUpgradeable.verifyCallResultFromTarget(address,bool,bytes,string) (lib/open
Configurable._setConfig(bytes32,uint256) (src/staking/config/Configurable.sol#26-3
DepositsAdapter._decreaseNextBalance(DepositsAdapter.DeferredDeposit,uint256) (src.
DepositsAdapter._increaseNextBalance(DepositsAdapter.DeferredDeposit,uint256) (src.
DepositsAdapter._previousEpochBalance(DepositsAdapter.DeferredDeposit) (src/stakin
DepositsAdapter._storeDeposit(DepositsAdapter.DeferredDeposit,DepositsAdapter.Defe
Initializable._disableInitializers() (lib/openzeppelin-contracts/contracts-upgrade
Initializable._getInitializedVersion() (lib/openzeppelin-contracts/contracts-upgra
Initializable._isInitializing() (lib/openzeppelin-contracts/contracts-upgradeable/
MathUpgradeable.average(uint256,uint256) (lib/openzeppelin-contracts/contracts-upg
MathUpgradeable.ceilDiv(uint256,uint256) (lib/openzeppelin-contracts/contracts-upg
MathUpgradeable.log10(uint256) (lib/openzeppelin-contracts/contracts-upgradeable/u
MathUpgradeable.log10(uint256,MathUpgradeable.Rounding) (lib/openzeppelin-contract
```

```
MathUpgradeable.log2(uint256,MathUpgradeable.Rounding) (lib/openzeppelin-contracts
MathUpgradeable.log256(uint256) (lib/openzeppelin-contracts/contracts-upgradeable/
MathUpgradeable.log256(uint256,MathUpgradeable.Rounding) (lib/openzeppelin-contrac
MathUpgradeable.min(uint256,uint256) (lib/openzeppelin-contracts/contracts-upgrade
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contra
MathUpgradeable.mulDiv(uint256,uint256,uint256,MathUpgradeable.Rounding) (lib/open
MathUpgradeable.sqrt(uint256) (lib/openzeppelin-contracts/contracts-upgradeable/ut
MathUpgradeable.sqrt(uint256,MathUpgradeable.Rounding) (lib/openzeppelin-contracts
SafeCastUpgradeable.toInt104(int256) (lib/openzeppelin-contracts/contracts-upgrade
SafeCastUpgradeable.toInt112(int256) (lib/openzeppelin-contracts/contracts-upgrade
SafeCastUpgradeable.toInt120(int256) (lib/openzeppelin-contracts/contracts-upgrade
SafeCastUpgradeable.toInt128(int256) (lib/openzeppelin-contracts/contracts-upgrade
SafeCastUpgradeable.toInt136(int256) (lib/openzeppelin-contracts/contracts-upgrade
SafeCastUpgradeable.toInt144(int256) (lib/openzeppelin-contracts/contracts-upgrade
SafeCastUpgradeable.toInt152(int256) (lib/openzeppelin-contracts/contracts-upgrade
SafeCastUpgradeable.toInt16(int256) (lib/openzeppelin-contracts/contracts-upgradea
SafeCastUpgradeable.toInt160(int256) (lib/openzeppelin-contracts/contracts-upgrade
SafeCastUpgradeable.toInt168(int256) (lib/openzeppelin-contracts/contracts-upgrade
SafeCastUpgradeable.toInt176(int256) (lib/openzeppelin-contracts/contracts-upgrade
SafeCastUpgradeable.toInt184(int256) (lib/openzeppelin-contracts/contracts-upgrade
SafeCastUpgradeable.toInt192(int256) (lib/openzeppelin-contracts/contracts-upgrade
SafeCastUpgradeable.toInt200(int256) (lib/openzeppelin-contracts/contracts-upgrade
SafeCastUpgradeable.toInt208(int256) (lib/openzeppelin-contracts/contracts-upgrade
SafeCastUpgradeable.toInt216(int256) (lib/openzeppelin-contracts/contracts-upgrade
SafeCastUpgradeable.toInt224(int256) (lib/openzeppelin-contracts/contracts-upgrade
SafeCastUpgradeable.toInt232(int256) (lib/openzeppelin-contracts/contracts-upgrade
SafeCastUpgradeable.toInt24(int256) (lib/openzeppelin-contracts/contracts-upgradea
SafeCastUpgradeable.toInt240(int256) (lib/openzeppelin-contracts/contracts-upgrade
SafeCastUpgradeable.toInt248(int256) (lib/openzeppelin-contracts/contracts-upgrade
SafeCastUpgradeable.toInt256(uint256) (lib/openzeppelin-contracts/contracts-upgrad
SafeCastUpgradeable.toInt32(int256) (lib/openzeppelin-contracts/contracts-upgradea
SafeCastUpgradeable.toInt40(int256) (lib/openzeppelin-contracts/contracts-upgradea
SafeCastUpgradeable.toInt48(int256) (lib/openzeppelin-contracts/contracts-upgradea
SafeCastUpgradeable.toInt56(int256) (lib/openzeppelin-contracts/contracts-upgradea
SafeCastUpgradeable.toInt64(int256) (lib/openzeppelin-contracts/contracts-upgradea
SafeCastUpgradeable.toInt72(int256) (lib/openzeppelin-contracts/contracts-upgradea
SafeCastUpgradeable.toInt8(int256) (lib/openzeppelin-contracts/contracts-upgradeab
SafeCastUpgradeable.toInt80(int256) (lib/openzeppelin-contracts/contracts-upgradea
SafeCastUpgradeable.toInt88(int256) (lib/openzeppelin-contracts/contracts-upgradea
SafeCastUpgradeable.toInt96(int256) (lib/openzeppelin-contracts/contracts-upgradea
SafeCastUpgradeable.toUint104(uint256) (lib/openzeppelin-contracts/contracts-upgra
SafeCastUpgradeable.toUint112(uint256) (lib/openzeppelin-contracts/contracts-upgra
SafeCastUpgradeable.toUint120(uint256) (lib/openzeppelin-contracts/contracts-upgra
SafeCastUpgradeable.toUint128(uint256) (lib/openzeppelin-contracts/contracts-upgra
SafeCastUpgradeable.toUint136(uint256) (lib/openzeppelin-contracts/contracts-upgra
SafeCastUpgradeable.toUint144(uint256) (lib/openzeppelin-contracts/contracts-upgra
SafeCastUpgradeable.toUint152(uint256) (lib/openzeppelin-contracts/contracts-upgra
SafeCastUpgradeable.toUint16(uint256) (lib/openzeppelin-contracts/contracts-upgrad
```

- Finalizer.sol

```
StakingStorage._poolsArchive (src/staking/StakingStorage.sol#74) is never initialize
        - StakingStorage.poolMetadata(bytes32) (src/staking/StakingStorage.sol#93-95
        - StakingStorage.isPoolOwner(address,bytes32) (src/staking/StakingStorage.so
        - StakingStorage.poolExists(bytes32) (src/staking/StakingStorage.sol#107-109
        - StakingStorage.poolOperative(bytes32) (src/staking/StakingStorage.sol#114-
        - StakingStorage.poolJailed(bytes32) (src/staking/StakingStorage.sol#129-131
        - StakingStorage.getPoolStake(bytes32) (src/staking/StakingStorage.sol#137-1
        - StakingStorage.getDelegatorStake(address,bytes32) (src/staking/StakingStor
        - StakingStorage.getDelegatorCurrency(address,bytes32,IERC20Upgradeable) (sr
        - StakingStorage.getDelegatorPendingWithdrawal(address,bytes32) (src/staking
        - Finalizer._finalizePool(bytes32,uint16,uint16) (src/staking/rewards/finali
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitializ

MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contract
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-upg
        - inverse = (3 * denominator) ^ 2 (lib/openzeppelin-contracts/contracts-upgr
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contract
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-upg
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contracts
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contract
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-upg
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contracts
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contract
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-upg
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contracts
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contract
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-upg
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contracts
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contract
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-upg
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contracts
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contract
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-upg
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contracts
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contract
        - prod0 = prod0 / twos (lib/openzeppelin-contracts/contracts-upgradeable/uti
        - result = prod0 * inverse (lib/openzeppelin-contracts/contracts-upgradeable
Finalizer._finalizePool(bytes32,uint16,uint16) (src/staking/rewards/finalizer.sol#14
        - baseReward = (((ownerStake * ownerBaseApy) / PERCENTAGE_SCALING_FACTOR) *
Finalizer._finalizePool(bytes32,uint16,uint16) (src/staking/rewards/finalizer.sol#14
        - baseReward = (((ownerStake * ownerBaseApy) / PERCENTAGE_SCALING_FACTOR) *
        - baseReward = (baseReward * performance) / PERCENTAGE_SCALING_FACTOR (src/s
Finalizer._finalizePool(bytes32,uint16,uint16) (src/staking/rewards/finalizer.sol#14
        - poolReward = (poolReward * performance) / PERCENTAGE_SCALING_FACTOR (src/s
        - commissionReward = (poolReward * poolPtr.meta.commissionFee) / PERCENTAGE_
Finalizer.computePoolEpochRewards(uint256,uint256) (src/staking/rewards/finalizer.so
        - earningStake = (ownerStake * (totalStake - earningStake)) / saturationCap
        - earningStake = (earningStake * pledgeFactorNum) / pledgeFactorDen (src/sta
Finalizer.computePoolEpochRewards(uint256,uint256) (src/staking/rewards/finalizer.so
        - earningStake = (earningStake * pledgeFactorNum) / pledgeFactorDen (src/sta
        - earningStake = (earningStake * pledgeFactorDen) / (pledgeFactorDen + pledg
Finalizer.computePoolEpochRewards(uint256,uint256) (src/staking/rewards/finalizer.so
        - (((earningStake * getConfig(POOL_EARNING_STAKE_APY)) / PERCENTAGE_SCALING_
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-befo
```

```
DepositsAdapter._loadDeposit(DepositsAdapter.DeferredDeposit).currentEpoch (src/sta
        - StakingEpochManager.currentEpoch() (src/staking/epochs/StakingEpochManage
DepositsAdapter._previousEpochBalance(DepositsAdapter.DeferredDeposit).currentEpoch
        - StakingEpochManager.currentEpoch() (src/staking/epochs/StakingEpochManage
Finalizer._finalizePool(bytes32,uint16,uint16).currentEpoch (src/staking/rewards/fi
        - StakingEpochManager.currentEpoch() (src/staking/epochs/StakingEpochManage
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-vari

StakingEpochManager._endEpoch() (src/staking/epochs/StakingEpochManager.sol#69-76)
        Dangerous comparisons:
        - require(bool,string)(block.timestamp >= getEpochEarliestEndTime(),E301) (
Finalizer.unclaimedRewards(address) (src/staking/rewards/finalizer.sol#75-81) uses
        Dangerous comparisons:
        - rewards != 0 (src/staking/rewards/finalizer.sol#78)
Finalizer._finalizePool(bytes32,uint16,uint16) (src/staking/rewards/finalizer.sol#1
        Dangerous comparisons:
        - require(bool,string)(_depositedRewardsFund >= (poolReward + baseReward),E
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-time

AddressUpgradeable._revert(bytes,string) (lib/openzeppelin-contracts/contracts-upgr
        - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/Addres
StringsUpgradeable.toString(uint256) (lib/openzeppelin-contracts/contracts-upgradea
        - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/String
        - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/String
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/M
        - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/M
        - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/M
StakingStorage.filterPoolIdsIdle(bytes32[]) (src/staking/StakingStorage.sol#185-197
        - INLINE ASM (src/staking/StakingStorage.sol#194-196)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-u

Different versions of Solidity are used:
        - Version used: ['^0.8.0', '^0.8.1', '^0.8.11', '^0.8.2']
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/access/AccessCon
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/access/IAccessCo
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/security/Pausabl
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/token/ERC20/IERC
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/ContextUpg
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/StringsUpg
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/introspect
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/introspect
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/MathU
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/SafeC
        - ^0.8.1 (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpg
        - ^0.8.11 (src/staking/StakingStorage.sol#2)
        - ^0.8.11 (src/staking/config/Configurable.sol#2)
        - ^0.8.11 (src/staking/config/StakingConstants.sol#2)
        - ^0.8.11 (src/staking/deposits/DepositsAdapter.sol#2)
        - ^0.8.11 (src/staking/epochs/StakingEpochManager.sol#2)
        - ^0.8.11 (src/staking/rewards/finalizer.sol#2)
        - ^0.8.2 (lib/openzeppelin-contracts/contracts-upgradeable/proxy/utils/Init
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-
```

```
Finalizer._finalizePool(bytes32,uint16,uint16) (src/staking/rewards/finalizer.sol#1
        - _depositedRewardsFund -= (poolReward + baseReward) (src/staking/rewards/f
Finalizer._finalizePool(bytes32,uint16,uint16) (src/staking/rewards/finalizer.sol#1
        - _poolsToFinalizePreviousEpoch -= 1 (src/staking/rewards/finalizer.sol#205
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-ope

AccessControlUpgradeable.__AccessControl_init() (lib/openzeppelin-contracts/contrac
AccessControlUpgradeable.__AccessControl_init_unchained() (lib/openzeppelin-contrac
AccessControlUpgradeable._setRoleAdmin(bytes32,bytes32) (lib/openzeppelin-contracts
AccessControlUpgradeable._setupRole(bytes32,address) (lib/openzeppelin-contracts/co
AddressUpgradeable._revert(bytes,string) (lib/openzeppelin-contracts/contracts-upgr
AddressUpgradeable.functionCall(address,bytes) (lib/openzeppelin-contracts/contract
AddressUpgradeable.functionCall(address,bytes,string) (lib/openzeppelin-contracts/c
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (lib/openzeppelin-c
AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (lib/openzep
AddressUpgradeable.functionStaticCall(address,bytes) (lib/openzeppelin-contracts/co
AddressUpgradeable.functionStaticCall(address,bytes,string) (lib/openzeppelin-contr
AddressUpgradeable.isContract(address) (lib/openzeppelin-contracts/contracts-upgrad
AddressUpgradeable.sendValue(address,uint256) (lib/openzeppelin-contracts/contracts
AddressUpgradeable.verifyCallResult(bool,bytes,string) (lib/openzeppelin-contracts/
AddressUpgradeable.verifyCallResultFromTarget(address,bool,bytes,string) (lib/openz
Configurable._setConfig(bytes32,uint256) (src/staking/config/Configurable.sol#26-30
ContextUpgradeable.__Context_init() (lib/openzeppelin-contracts/contracts-upgradeab
ContextUpgradeable.__Context_init_unchained() (lib/openzeppelin-contracts/contracts
ContextUpgradeable._msgData() (lib/openzeppelin-contracts/contracts-upgradeable/uti
DepositsAdapter._decreaseNextBalance(DepositsAdapter.DeferredDeposit,uint256) (src/
DepositsAdapter._increaseNextBalance(DepositsAdapter.DeferredDeposit,uint256) (src/
DepositsAdapter._storeDeposit(DepositsAdapter.DeferredDeposit,DepositsAdapter.Defer
ERC165Upgradeable.__ERC165_init() (lib/openzeppelin-contracts/contracts-upgradeable
ERC165Upgradeable.__ERC165_init_unchained() (lib/openzeppelin-contracts/contracts-u
Finalizer._markPoolAsCreated(bytes32) (src/staking/rewards/finalizer.sol#86-89) is
Finalizer._markPoolAsTerminated(bytes32) (src/staking/rewards/finalizer.sol#96-101)
Initializable._disableInitializers() (lib/openzeppelin-contracts/contracts-upgradea
Initializable._getInitializedVersion() (lib/openzeppelin-contracts/contracts-upgrad
Initializable._isInitializing() (lib/openzeppelin-contracts/contracts-upgradeable/p
MathUpgradeable.average(uint256,uint256) (lib/openzeppelin-contracts/contracts-upgr
MathUpgradeable.ceilDiv(uint256,uint256) (lib/openzeppelin-contracts/contracts-upgr
MathUpgradeable.log10(uint256) (lib/openzeppelin-contracts/contracts-upgradeable/ut
MathUpgradeable.log10(uint256,MathUpgradeable.Rounding) (lib/openzeppelin-contracts
MathUpgradeable.log2(uint256) (lib/openzeppelin-contracts/contracts-upgradeable/uti
MathUpgradeable.log2(uint256,MathUpgradeable.Rounding) (lib/openzeppelin-contracts/
MathUpgradeable.log256(uint256) (lib/openzeppelin-contracts/contracts-upgradeable/u
MathUpgradeable.log256(uint256,MathUpgradeable.Rounding) (lib/openzeppelin-contract
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
MathUpgradeable.mulDiv(uint256,uint256,uint256,MathUpgradeable.Rounding) (lib/openz
MathUpgradeable.sqrt(uint256) (lib/openzeppelin-contracts/contracts-upgradeable/uti
MathUpgradeable.sqrt(uint256,MathUpgradeable.Rounding) (lib/openzeppelin-contracts/
PausableUpgradeable.__Pausable_init() (lib/openzeppelin-contracts/contracts-upgrade
PausableUpgradeable.__Pausable_init_unchained() (lib/openzeppelin-contracts/contrac
PausableUpgradeable._pause() (lib/openzeppelin-contracts/contracts-upgradeable/secu
```

AUTOMATED TESTING

- DelegatorRewardsAdapter.sol

```
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-up
        - inverse = (3 * denominator) ^ 2 (lib/openzeppelin-contracts/contracts-upg
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-up
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contract
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-up
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contract
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-up
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contract
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-up
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contract
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-up
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contract
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-up
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contract
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - prod0 = prod0 / twos (lib/openzeppelin-contracts/contracts-upgradeable/ut
        - result = prod0 * inverse (lib/openzeppelin-contracts/contracts-upgradeabl
Finalizer._finalizePool(bytes32,uint16,uint16) (src/staking/rewards/Finalizer.sol#1
        - baseReward = (((ownerStake * ownerBaseApy) / PERCENTAGE_SCALING_FACTOR) *
Finalizer._finalizePool(bytes32,uint16,uint16) (src/staking/rewards/Finalizer.sol#1
        - baseReward = (((ownerStake * ownerBaseApy) / PERCENTAGE_SCALING_FACTOR) *
        - baseReward = (baseReward * performance) / PERCENTAGE_SCALING_FACTOR (src/
Finalizer._finalizePool(bytes32,uint16,uint16) (src/staking/rewards/Finalizer.sol#1
        - poolReward = (poolReward * performance) / PERCENTAGE_SCALING_FACTOR (src/
        - commissionReward = (poolReward * poolPtr.meta.commissionFee) / PERCENTAGE
Finalizer.computePoolEpochRewards(uint256,uint256) (src/staking/rewards/Finalizer.s
        - earningStake = (ownerStake * (totalStake - earningStake)) / saturationCap
        - earningStake = (earningStake * pledgeFactorNum) / pledgeFactorDen (src/st
Finalizer.computePoolEpochRewards(uint256,uint256) (src/staking/rewards/Finalizer.s
        - earningStake = (earningStake * pledgeFactorNum) / pledgeFactorDen (src/st
        - earningStake = (earningStake * pledgeFactorDen) / (pledgeFactorDen + pled
Finalizer.computePoolEpochRewards(uint256,uint256) (src/staking/rewards/Finalizer.s
        - (((earningStake * getConfig(POOL_EARNING_STAKE_APY)) / PERCENTAGE_SCALING
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-bef

DelegatorRewardsAdapter._collectDelegatorRewardsMany(address,bytes32[]).it (src/sta
DelegatorRewardsAdapter.computeDelegatorReward(address,bytes32[]).it (src/staking/r
Finalizer.finalizePools(bytes32[],uint16[],uint16[]).it (src/staking/rewards/Finali
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitiali

DepositsAdapter._loadDeposit(DepositsAdapter.DeferredDeposit).currentEpoch (src/sta
        - StakingEpochManager.currentEpoch() (src/staking/epochs/StakingEpochManage
DepositsAdapter._previousEpochBalance(DepositsAdapter.DeferredDeposit).currentEpoch
```

```
        - StakingEpochManager.currentEpoch() (src/staking/epochs/StakingEpochManage
DelegatorRewardsAdapter._computeDelegatorRewardOverInterval(bytes32,uint256,uint64,
        - Finalizer.endEpoch() (src/staking/rewards/Finalizer.sol#106-115) (functio
DelegatorRewardsAdapter._computeDelegatorReward(address,bytes32).currentEpoch (src/
        - StakingEpochManager.currentEpoch() (src/staking/epochs/StakingEpochManage
Finalizer._finalizePool(bytes32,uint16,uint16).currentEpoch (src/staking/rewards/Fi
        - StakingEpochManager.currentEpoch() (src/staking/epochs/StakingEpochManage
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-vari

StakingEpochManager._endEpoch() (src/staking/epochs/StakingEpochManager.sol#69-76)
        Dangerous comparisons:
        - require(bool,string)(block.timestamp >= getEpochEarliestEndTime(),E301) (
Finalizer.unclaimedRewards(address) (src/staking/rewards/Finalizer.sol#75-81) uses
        Dangerous comparisons:
        - rewards != 0 (src/staking/rewards/Finalizer.sol#78)
Finalizer._finalizePool(bytes32,uint16,uint16) (src/staking/rewards/Finalizer.sol#1
        Dangerous comparisons:
        - require(bool,string)(_depositedRewardsFund >= (poolReward + baseReward),E
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-time

AddressUpgradeable._revert(bytes,string) (lib/openzeppelin-contracts/contracts-upgr
        - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/Addres
StringsUpgradeable.toString(uint256) (lib/openzeppelin-contracts/contracts-upgradea
        - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/String
        - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/String
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/M
        - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/M
        - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/M
StakingStorage.filterPoolIdsIdle(bytes32[]) (src/staking/StakingStorage.sol#185-197
        - INLINE ASM (src/staking/StakingStorage.sol#194-196)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-u

Different versions of Solidity are used:
        - Version used: ['^0.8.0', '^0.8.1', '^0.8.11', '^0.8.2']
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/access/AccessCon
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/access/IAccessCo
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/security/Pausabl
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/token/ERC20/IERC
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/ContextUpg
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/StringsUpg
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/introspect
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/introspect
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/MathU
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/SafeC
        - ^0.8.1 (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpg
        - ^0.8.11 (src/staking/StakingStorage.sol#2)
        - ^0.8.11 (src/staking/config/Configurable.sol#2)
        - ^0.8.11 (src/staking/config/StakingConstants.sol#2)
        - ^0.8.11 (src/staking/deposits/DepositsAdapter.sol#2)
        - ^0.8.11 (src/staking/epochs/StakingEpochManager.sol#2)
        - ^0.8.11 (src/staking/rewards/DelegatorRewardsAdapter.sol#2)
        - ^0.8.11 (src/staking/rewards/Finalizer.sol#2)
        - ^0.8.2 (lib/openzeppelin-contracts/contracts-upgradeable/proxy/utils/Init
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-
```

```
AccessControlUpgradeable.__AccessControl_init() (lib/openzeppelin-contracts/contrac
AccessControlUpgradeable.__AccessControl_init_unchained() (lib/openzeppelin-contrac
AccessControlUpgradeable._setRoleAdmin(bytes32,bytes32) (lib/openzeppelin-contracts
AccessControlUpgradeable._setupRole(bytes32,address) (lib/openzeppelin-contracts/co
AddressUpgradeable._revert(bytes,string) (lib/openzeppelin-contracts/contracts-upgr
AddressUpgradeable.functionCall(address,bytes) (lib/openzeppelin-contracts/contract
AddressUpgradeable.functionCall(address,bytes,string) (lib/openzeppelin-contracts/c
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (lib/openzeppelin-c
AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (lib/openzep
AddressUpgradeable.functionStaticCall(address,bytes) (lib/openzeppelin-contracts/co
AddressUpgradeable.functionStaticCall(address,bytes,string) (lib/openzeppelin-contr
AddressUpgradeable.isContract(address) (lib/openzeppelin-contracts/contracts-upgrad
AddressUpgradeable.sendValue(address,uint256) (lib/openzeppelin-contracts/contracts
AddressUpgradeable.verifyCallResult(bool,bytes,string) (lib/openzeppelin-contracts/
AddressUpgradeable.verifyCallResultFromTarget(address,bool,bytes,string) (lib/openz
Configurable._setConfig(bytes32,uint256) (src/staking/config/Configurable.sol#26-30
ContextUpgradeable.__Context_init() (lib/openzeppelin-contracts/contracts-upgradeab
ContextUpgradeable.__Context_init_unchained() (lib/openzeppelin-contracts/contracts
ContextUpgradeable._msgData() (lib/openzeppelin-contracts/contracts-upgradeable/uti
DelegatorRewardsAdapter._collectDelegatorRewards(address,bytes32) (src/staking/rewa
DelegatorRewardsAdapter._collectDelegatorRewardsMany(address,bytes32[]) (src/stakin
DepositsAdapter._decreaseNextBalance(DepositsAdapter.DeferredDeposit,uint256) (src/
DepositsAdapter._increaseNextBalance(DepositsAdapter.DeferredDeposit,uint256) (src/
DepositsAdapter._storeDeposit(DepositsAdapter.DeferredDeposit,DepositsAdapter.Defer
ERC165Upgradeable.__ERC165_init() (lib/openzeppelin-contracts/contracts-upgradeable
ERC165Upgradeable.__ERC165_init_unchained() (lib/openzeppelin-contracts/contracts-u
Finalizer._markPoolAsCreated(bytes32) (src/staking/rewards/Finalizer.sol#86-89) is
Finalizer._markPoolAsTerminated(bytes32) (src/staking/rewards/Finalizer.sol#96-101)
Initializable._disableInitializers() (lib/openzeppelin-contracts/contracts-upgradea
Initializable._getInitializedVersion() (lib/openzeppelin-contracts/contracts-upgrad
Initializable._isInitializing() (lib/openzeppelin-contracts/contracts-upgradeable/p
MathUpgradeable.average(uint256,uint256) (lib/openzeppelin-contracts/contracts-upgr
MathUpgradeable.ceilDiv(uint256,uint256) (lib/openzeppelin-contracts/contracts-upgr
MathUpgradeable.log10(uint256) (lib/openzeppelin-contracts/contracts-upgradeable/ut
MathUpgradeable.log10(uint256,MathUpgradeable.Rounding) (lib/openzeppelin-contracts
MathUpgradeable.log2(uint256) (lib/openzeppelin-contracts/contracts-upgradeable/uti
MathUpgradeable.log2(uint256,MathUpgradeable.Rounding) (lib/openzeppelin-contracts/
MathUpgradeable.log256(uint256) (lib/openzeppelin-contracts/contracts-upgradeable/u
MathUpgradeable.log256(uint256,MathUpgradeable.Rounding) (lib/openzeppelin-contract
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
MathUpgradeable.mulDiv(uint256,uint256,uint256,MathUpgradeable.Rounding) (lib/openz
MathUpgradeable.sqrt(uint256) (lib/openzeppelin-contracts/contracts-upgradeable/uti
MathUpgradeable.sqrt(uint256,MathUpgradeable.Rounding) (lib/openzeppelin-contracts/
PausableUpgradeable.__Pausable_init() (lib/openzeppelin-contracts/contracts-upgrade
PausableUpgradeable.__Pausable_init_unchained() (lib/openzeppelin-contracts/contrac
PausableUpgradeable._pause() (lib/openzeppelin-contracts/contracts-upgradeable/secu
PausableUpgradeable._requirePaused() (lib/openzeppelin-contracts/contracts-upgradea
```

- StakingEpochManager.sol

```
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-up
        - inverse = (3 * denominator) ^ 2 (lib/openzeppelin-contracts/contracts-upg
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-up
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contract
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-up
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contract
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-up
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contract
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-up
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contract
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-up
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contract
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-up
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contract
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - prod0 = prod0 / twos (lib/openzeppelin-contracts/contracts-upgradeable/ut
        - result = prod0 * inverse (lib/openzeppelin-contracts/contracts-upgradeabl
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-bef

StakingEpochManager._endEpoch() (src/staking/epochs/StakingEpochManager.sol#69-76)
        Dangerous comparisons:
        - require(bool,string)(block.timestamp >= getEpochEarliestEndTime(),E301) (
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-time

AddressUpgradeable._revert(bytes,string) (lib/openzeppelin-contracts/contracts-upgr
        - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/Addres
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/M
        - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/M
        - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/M
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-u

Different versions of Solidity are used:
        - Version used: ['^0.8.0', '^0.8.1', '^0.8.11', '^0.8.2']
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/MathU
        - ^0.8.1 (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpg
        - ^0.8.11 (src/staking/config/Configurable.sol#2)
        - ^0.8.11 (src/staking/config/StakingConstants.sol#2)
        - ^0.8.11 (src/staking/epochs/StakingEpochManager.sol#2)
        - ^0.8.2 (lib/openzeppelin-contracts/contracts-upgradeable/proxy/utils/Init
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-
```

```
AddressUpgradeable._revert(bytes,string) (lib/openzeppelin-contracts/contracts-upgr
AddressUpgradeable.functionCall(address,bytes) (lib/openzeppelin-contracts/contract
AddressUpgradeable.functionCall(address,bytes,string) (lib/openzeppelin-contracts/c
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (lib/openzeppelin-c
AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (lib/openzep
AddressUpgradeable.functionStaticCall(address,bytes) (lib/openzeppelin-contracts/co
AddressUpgradeable.functionStaticCall(address,bytes,string) (lib/openzeppelin-contr
AddressUpgradeable.isContract(address) (lib/openzeppelin-contracts/contracts-upgrad
AddressUpgradeable.sendValue(address,uint256) (lib/openzeppelin-contracts/contracts
AddressUpgradeable.verifyCallResult(bool,bytes,string) (lib/openzeppelin-contracts/
AddressUpgradeable.verifyCallResultFromTarget(address,bool,bytes,string) (lib/openz
Configurable._setConfig(bytes32,uint256) (src/staking/config/Configurable.sol#26-30
Initializable._disableInitializers() (lib/openzeppelin-contracts/contracts-upgradea
Initializable._getInitializedVersion() (lib/openzeppelin-contracts/contracts-upgrad
Initializable._isInitializing() (lib/openzeppelin-contracts/contracts-upgradeable/p
MathUpgradeable.average(uint256,uint256) (lib/openzeppelin-contracts/contracts-upgr
MathUpgradeable.ceilDiv(uint256,uint256) (lib/openzeppelin-contracts/contracts-upgr
MathUpgradeable.log10(uint256) (lib/openzeppelin-contracts/contracts-upgradeable/ut
MathUpgradeable.log10(uint256,MathUpgradeable.Rounding) (lib/openzeppelin-contracts
MathUpgradeable.log2(uint256) (lib/openzeppelin-contracts/contracts-upgradeable/uti
MathUpgradeable.log2(uint256,MathUpgradeable.Rounding) (lib/openzeppelin-contracts/
MathUpgradeable.log256(uint256) (lib/openzeppelin-contracts/contracts-upgradeable/u
MathUpgradeable.log256(uint256,MathUpgradeable.Rounding) (lib/openzeppelin-contract
MathUpgradeable.min(uint256,uint256) (lib/openzeppelin-contracts/contracts-upgradea
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
MathUpgradeable.mulDiv(uint256,uint256,uint256,MathUpgradeable.Rounding) (lib/openz
MathUpgradeable.sqrt(uint256) (lib/openzeppelin-contracts/contracts-upgradeable/uti
MathUpgradeable.sqrt(uint256,MathUpgradeable.Rounding) (lib/openzeppelin-contracts/
StakingEpochManager.__StakingEpochManager_init_unchained(uint256) (src/staking/epoc
StakingEpochManager._endEpoch() (src/staking/epochs/StakingEpochManager.sol#69-76)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.2 (lib/openzeppelin-contracts/contracts-upgradeable/proxy/utils/
Pragma version^0.8.1 (lib/openzeppelin-contracts/contracts-upgradeable/utils/Addres
Pragma version^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/M
Pragma version^0.8.11 (src/staking/config/Configurable.sol#2) allows old versions
Pragma version^0.8.11 (src/staking/config/StakingConstants.sol#2) allows old versio
Pragma version^0.8.11 (src/staking/epochs/StakingEpochManager.sol#2) allows old ver
solc-0.8.11 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-

Low level call in AddressUpgradeable.sendValue(address,uint256) (lib/openzeppelin-c
        - (success) = recipient.call{value: amount}() (lib/openzeppelin-contracts/c
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,st
        - (success,returndata) = target.call{value: value}(data) (lib/openzeppelin-
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (lib/
        - (success,returndata) = target.staticcall(data) (lib/openzeppelin-contract
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-
```

```
Variable Configurable.__gap (src/staking/config/Configurable.sol#36) is not in mixe
Function StakingEpochManager.__StakingEpochManager_init_unchained(uint256) (src/sta
Variable StakingEpochManager.__gap (src/staking/epochs/StakingEpochManager.sol#78)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformanc

Variable StakingConstants.OWNER_STAKE_INFLUENCE_DEN (src/staking/config/StakingCons
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-r

StakingConstants.OWNER_ROLE (src/staking/config/StakingConstants.sol#8) is never us
StakingConstants.PAUSER_ROLE (src/staking/config/StakingConstants.sol#11) is never
StakingConstants.FINALIZER_ROLE (src/staking/config/StakingConstants.sol#14) is nev
StakingConstants.JAILER_ROLE (src/staking/config/StakingConstants.sol#17) is never
StakingConstants.POOL_EARNING_STAKE_APY (src/staking/config/StakingConstants.sol#26
StakingConstants.MIN_POOL_OWNER_STAKE (src/staking/config/StakingConstants.sol#29)
StakingConstants.MIN_DELEGATOR_STAKE (src/staking/config/StakingConstants.sol#32) i
StakingConstants.POOL_SATURATION_CAP (src/staking/config/StakingConstants.sol#35) i
StakingConstants.OWNER_STAKE_INFLUENCE_NUM (src/staking/config/StakingConstants.sol
StakingConstants.OWNER_STAKE_INFLUENCE_DEN (src/staking/config/StakingConstants.sol
StakingConstants.WITHDRAWAL_LOCK_TIME (src/staking/config/StakingConstants.sol#42)
StakingConstants.POOL_COMMISSION_FEE_COOLDOWN (src/staking/config/StakingConstants.
StakingConstants.POOL_UNJAILING_FEE (src/staking/config/StakingConstants.sol#48) is
StakingConstants.PERCENTAGE_SCALING_FACTOR (src/staking/config/StakingConstants.sol
StakingConstants.MAX_OWNER_BASE_APY (src/staking/config/StakingConstants.sol#56) is
StakingConstants.MIN_POOL_COMMISSION_FEE (src/staking/config/StakingConstants.sol#5
StakingConstants.TOKEN_FRACTION_SCALING_FACTOR (src/staking/config/StakingConstants
StakingConstants.ONE_YEAR_SECS (src/staking/config/StakingConstants.sol#68) is neve
StakingEpochManager.__gap (src/staking/epochs/StakingEpochManager.sol#78) is never
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-sta
```

AUTOMATED TESTING

- DelegationsAdapter.sol

```
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-up
        - inverse = (3 * denominator) ^ 2 (lib/openzeppelin-contracts/contracts-upg
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-up
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contract
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-up
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contract
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-up
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contract
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-up
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contract
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-up
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contract
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-up
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contract
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - prod0 = prod0 / twos (lib/openzeppelin-contracts/contracts-upgradeable/ut
        - result = prod0 * inverse (lib/openzeppelin-contracts/contracts-upgradeabl
Finalizer._finalizePool(bytes32,uint16,uint16) (src/staking/rewards/Finalizer.sol#1
        - baseReward = (((ownerStake * ownerBaseApy) / PERCENTAGE_SCALING_FACTOR) *
Finalizer._finalizePool(bytes32,uint16,uint16) (src/staking/rewards/Finalizer.sol#1
        - baseReward = (((ownerStake * ownerBaseApy) / PERCENTAGE_SCALING_FACTOR) *
        - baseReward = (baseReward * performance) / PERCENTAGE_SCALING_FACTOR (src/
Finalizer._finalizePool(bytes32,uint16,uint16) (src/staking/rewards/Finalizer.sol#1
        - poolReward = (poolReward * performance) / PERCENTAGE_SCALING_FACTOR (src/
        - commissionReward = (poolReward * poolPtr.meta.commissionFee) / PERCENTAGE
Finalizer.computePoolEpochRewards(uint256,uint256) (src/staking/rewards/Finalizer.s
        - earningStake = (ownerStake * (totalStake - earningStake)) / saturationCap
        - earningStake = (earningStake * pledgeFactorNum) / pledgeFactorDen (src/st
Finalizer.computePoolEpochRewards(uint256,uint256) (src/staking/rewards/Finalizer.s
        - earningStake = (earningStake * pledgeFactorNum) / pledgeFactorDen (src/st
        - earningStake = (earningStake * pledgeFactorDen) / (pledgeFactorDen + pled
Finalizer.computePoolEpochRewards(uint256,uint256) (src/staking/rewards/Finalizer.s
        - (((earningStake * getConfig(POOL_EARNING_STAKE_APY)) / PERCENTAGE_SCALING
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-bef

DelegatorRewardsAdapter._collectDelegatorRewardsMany(address,bytes32[]).it (src/sta
DelegationsAdapter._updateDelegationCurrencies(bytes32,address,StakingStorage.Curre
Finalizer.finalizePools(bytes32[],uint16[],uint16[]).it (src/staking/rewards/Finali
DelegatorRewardsAdapter.computeDelegatorReward(address,bytes32[]).it (src/staking/r
DelegationsAdapter.validateCurrencies(StakingStorage.Currency[]).it (src/staking/de
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitiali
```

```
DepositsAdapter._loadDeposit(DepositsAdapter.DeferredDeposit).currentEpoch (src/sta
        - StakingEpochManager.currentEpoch() (src/staking/epochs/StakingEpochManage
DepositsAdapter._previousEpochBalance(DepositsAdapter.DeferredDeposit).currentEpoch
        - StakingEpochManager.currentEpoch() (src/staking/epochs/StakingEpochManage
DelegatorRewardsAdapter._computeDelegatorRewardOverInterval(bytes32,uint256,uint64,
        - Finalizer.endEpoch() (src/staking/rewards/Finalizer.sol#106-115) (functio
DelegatorRewardsAdapter._computeDelegatorReward(address,bytes32).currentEpoch (src/
        - StakingEpochManager.currentEpoch() (src/staking/epochs/StakingEpochManage
Finalizer._finalizePool(bytes32,uint16,uint16).currentEpoch (src/staking/rewards/Fi
        - StakingEpochManager.currentEpoch() (src/staking/epochs/StakingEpochManage
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-vari

StakingEpochManager._endEpoch() (src/staking/epochs/StakingEpochManager.sol#69-76)
        Dangerous comparisons:
        - require(bool,string)(block.timestamp >= getEpochEarliestEndTime(),E301) (
Finalizer.unclaimedRewards(address) (src/staking/rewards/Finalizer.sol#75-81) uses
        Dangerous comparisons:
        - rewards != 0 (src/staking/rewards/Finalizer.sol#78)
Finalizer._finalizePool(bytes32,uint16,uint16) (src/staking/rewards/Finalizer.sol#1
        Dangerous comparisons:
        - require(bool,string)(_depositedRewardsFund >= (poolReward + baseReward),E
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-time

AddressUpgradeable._revert(bytes,string) (lib/openzeppelin-contracts/contracts-upgr
        - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/Addres
StringsUpgradeable.toString(uint256) (lib/openzeppelin-contracts/contracts-upgradea
        - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/String
        - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/String
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/M
        - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/M
        - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/M
StakingStorage.filterPoolIdsIdle(bytes32[]) (src/staking/StakingStorage.sol#185-197
        - INLINE ASM (src/staking/StakingStorage.sol#194-196)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-u

Different versions of Solidity are used:
        - Version used: ['^0.8.0', '^0.8.1', '^0.8.11', '^0.8.2']
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/access/AccessCon
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/access/IAccessCo
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/security/Pausabl
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/token/ERC20/IERC
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/ContextUpg
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/StringsUpg
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/introspect
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/introspect
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/MathU
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/SafeC
        - ^0.8.1 (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpg
        - ^0.8.11 (src/staking/StakingStorage.sol#2)
```

```
        - ^0.8.11 (src/staking/config/StakingConstants.sol#2)
        - ^0.8.11 (src/staking/deposits/DelegationsAdapter.sol#2)
        - ^0.8.11 (src/staking/deposits/DepositsAdapter.sol#2)
        - ^0.8.11 (src/staking/epochs/StakingEpochManager.sol#2)
        - ^0.8.11 (src/staking/rewards/DelegatorRewardsAdapter.sol#2)
        - ^0.8.11 (src/staking/rewards/Finalizer.sol#2)
        - ^0.8.2 (lib/openzeppelin-contracts/contracts-upgradeable/proxy/utils/Init
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-

Finalizer._finalizePool(bytes32,uint16,uint16) (src/staking/rewards/Finalizer.sol#1
        - _depositedRewardsFund -= (poolReward + baseReward) (src/staking/rewards/F
Finalizer._finalizePool(bytes32,uint16,uint16) (src/staking/rewards/Finalizer.sol#1
        - _poolsToFinalizePreviousEpoch -= 1 (src/staking/rewards/Finalizer.sol#205
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-ope

AccessControlUpgradeable.__AccessControl_init() (lib/openzeppelin-contracts/contrac
AccessControlUpgradeable.__AccessControl_init_unchained() (lib/openzeppelin-contrac
AccessControlUpgradeable._setRoleAdmin(bytes32,bytes32) (lib/openzeppelin-contracts
AccessControlUpgradeable._setupRole(bytes32,address) (lib/openzeppelin-contracts/co
AddressUpgradeable._revert(bytes,string) (lib/openzeppelin-contracts/contracts-upgr
AddressUpgradeable.functionCall(address,bytes) (lib/openzeppelin-contracts/contract
AddressUpgradeable.functionCall(address,bytes,string) (lib/openzeppelin-contracts/c
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (lib/openzeppelin-c
AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (lib/openzep
AddressUpgradeable.functionStaticCall(address,bytes) (lib/openzeppelin-contracts/co
AddressUpgradeable.functionStaticCall(address,bytes,string) (lib/openzeppelin-contr
AddressUpgradeable.isContract(address) (lib/openzeppelin-contracts/contracts-upgrad
AddressUpgradeable.sendValue(address,uint256) (lib/openzeppelin-contracts/contracts
AddressUpgradeable.verifyCallResult(bool,bytes,string) (lib/openzeppelin-contracts/
AddressUpgradeable.verifyCallResultFromTarget(address,bool,bytes,string) (lib/openz
Configurable._setConfig(bytes32,uint256) (src/staking/config/Configurable.sol#26-30
ContextUpgradeable.__Context_init() (lib/openzeppelin-contracts/contracts-upgradeab
ContextUpgradeable.__Context_init_unchained() (lib/openzeppelin-contracts/contracts
ContextUpgradeable._msgData() (lib/openzeppelin-contracts/contracts-upgradeable/uti
DelegationsAdapter._initializePoolOwner(bytes32,address,StakingStorage.Currency[])
DelegationsAdapter._stake(bytes32,address,StakingStorage.Currency[]) (src/staking/d
DelegationsAdapter._unstake(bytes32,address,StakingStorage.Currency[]) (src/staking
DelegationsAdapter._updateDeferredDeposit(DepositsAdapter.DeferredDeposit,uint256,b
DelegationsAdapter._updateDelegation(bytes32,address,StakingStorage.Currency[],bool
DelegationsAdapter._updateDelegationCurrencies(bytes32,address,StakingStorage.Curre
DelegationsAdapter.validateCurrencies(StakingStorage.Currency[]) (src/staking/depos
DelegatorRewardsAdapter._collectDelegatorRewards(address,bytes32) (src/staking/rewa
DelegatorRewardsAdapter._collectDelegatorRewardsMany(address,bytes32[]) (src/stakin
DepositsAdapter._decreaseNextBalance(DepositsAdapter.DeferredDeposit,uint256) (src/
DepositsAdapter._increaseNextBalance(DepositsAdapter.DeferredDeposit,uint256) (src/
DepositsAdapter._storeDeposit(DepositsAdapter.DeferredDeposit,DepositsAdapter.Defer
ERC165Upgradeable.__ERC165_init() (lib/openzeppelin-contracts/contracts-upgradeable
ERC165Upgradeable.__ERC165_init_unchained() (lib/openzeppelin-contracts/contracts-u
Finalizer._markPoolAsCreated(bytes32) (src/staking/rewards/Finalizer.sol#86-89) is
Finalizer._markPoolAsTerminated(bytes32) (src/staking/rewards/Finalizer.sol#96-101)
Initializable._disableInitializers() (lib/openzeppelin-contracts/contracts-upgradea
```

- DepositsAdapter.sol

```
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-up
        - inverse = (3 * denominator) ^ 2 (lib/openzeppelin-contracts/contracts-upg
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-up
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contract
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-up
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contract
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-up
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contract
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-up
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contract
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-up
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contract
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-up
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contract
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - prod0 = prod0 / twos (lib/openzeppelin-contracts/contracts-upgradeable/ut
        - result = prod0 * inverse (lib/openzeppelin-contracts/contracts-upgradeabl
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-bef

DepositsAdapter._loadDeposit(DepositsAdapter.DeferredDeposit).currentEpoch (src/sta
        - StakingEpochManager.currentEpoch() (src/staking/epochs/StakingEpochManage
DepositsAdapter._previousEpochBalance(DepositsAdapter.DeferredDeposit).currentEpoch
        - StakingEpochManager.currentEpoch() (src/staking/epochs/StakingEpochManage
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-vari

StakingEpochManager._endEpoch() (src/staking/epochs/StakingEpochManager.sol#69-76)
        Dangerous comparisons:
        - require(bool,string)(block.timestamp >= getEpochEarliestEndTime(),E301) (
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-time

AddressUpgradeable._revert(bytes,string) (lib/openzeppelin-contracts/contracts-upgr
        - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/Addres
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/M
        - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/M
        - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/M
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-u
```

```
Different versions of Solidity are used:
        - Version used: ['^0.8.0', '^0.8.1', '^0.8.11', '^0.8.2']
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/MathU
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/SafeC
        - ^0.8.1 (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpg
        - ^0.8.11 (src/staking/config/Configurable.sol#2)
        - ^0.8.11 (src/staking/config/StakingConstants.sol#2)
        - ^0.8.11 (src/staking/deposits/DepositsAdapter.sol#2)
        - ^0.8.11 (src/staking/epochs/StakingEpochManager.sol#2)
        - ^0.8.2 (lib/openzeppelin-contracts/contracts-upgradeable/proxy/utils/Init
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-

AddressUpgradeable._revert(bytes,string) (lib/openzeppelin-contracts/contracts-upgr
AddressUpgradeable.functionCall(address,bytes) (lib/openzeppelin-contracts/contract
AddressUpgradeable.functionCall(address,bytes,string) (lib/openzeppelin-contracts/c
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (lib/openzeppelin-c
AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (lib/openzep
AddressUpgradeable.functionStaticCall(address,bytes) (lib/openzeppelin-contracts/co
AddressUpgradeable.functionStaticCall(address,bytes,string) (lib/openzeppelin-contr
AddressUpgradeable.isContract(address) (lib/openzeppelin-contracts/contracts-upgrad
AddressUpgradeable.sendValue(address,uint256) (lib/openzeppelin-contracts/contracts
AddressUpgradeable.verifyCallResult(bool,bytes,string) (lib/openzeppelin-contracts/
AddressUpgradeable.verifyCallResultFromTarget(address,bool,bytes,string) (lib/openz
Configurable._setConfig(bytes32,uint256) (src/staking/config/Configurable.sol#26-30
DepositsAdapter._decreaseNextBalance(DepositsAdapter.DeferredDeposit,uint256) (src/
DepositsAdapter._increaseNextBalance(DepositsAdapter.DeferredDeposit,uint256) (src/
DepositsAdapter._loadDeposit(DepositsAdapter.DeferredDeposit) (src/staking/deposits
DepositsAdapter._previousEpochBalance(DepositsAdapter.DeferredDeposit) (src/staking
DepositsAdapter._storeDeposit(DepositsAdapter.DeferredDeposit,DepositsAdapter.Defer
Initializable._disableInitializers() (lib/openzeppelin-contracts/contracts-upgradea
Initializable._getInitializedVersion() (lib/openzeppelin-contracts/contracts-upgrad
Initializable._isInitializing() (lib/openzeppelin-contracts/contracts-upgradeable/p
MathUpgradeable.average(uint256,uint256) (lib/openzeppelin-contracts/contracts-upgr
MathUpgradeable.ceilDiv(uint256,uint256) (lib/openzeppelin-contracts/contracts-upgr
MathUpgradeable.log10(uint256) (lib/openzeppelin-contracts/contracts-upgradeable/ut
MathUpgradeable.log10(uint256,MathUpgradeable.Rounding) (lib/openzeppelin-contracts
MathUpgradeable.log2(uint256) (lib/openzeppelin-contracts/contracts-upgradeable/uti
MathUpgradeable.log2(uint256,MathUpgradeable.Rounding) (lib/openzeppelin-contracts/
MathUpgradeable.log256(uint256) (lib/openzeppelin-contracts/contracts-upgradeable/u
MathUpgradeable.log256(uint256,MathUpgradeable.Rounding) (lib/openzeppelin-contract
MathUpgradeable.min(uint256,uint256) (lib/openzeppelin-contracts/contracts-upgradea
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
MathUpgradeable.mulDiv(uint256,uint256,uint256,MathUpgradeable.Rounding) (lib/openz
```

AUTOMATED TESTING

```
SafeCastUpgradeable.toInt128(int256) (lib/openzeppelin-contracts/contracts-upgrade
SafeCastUpgradeable.toInt136(int256) (lib/openzeppelin-contracts/contracts-upgrade
SafeCastUpgradeable.toInt144(int256) (lib/openzeppelin-contracts/contracts-upgrade
SafeCastUpgradeable.toInt152(int256) (lib/openzeppelin-contracts/contracts-upgrade
SafeCastUpgradeable.toInt16(int256) (lib/openzeppelin-contracts/contracts-upgraded
SafeCastUpgradeable.toInt160(int256) (lib/openzeppelin-contracts/contracts-upgrade
SafeCastUpgradeable.toInt168(int256) (lib/openzeppelin-contracts/contracts-upgrade
SafeCastUpgradeable.toInt176(int256) (lib/openzeppelin-contracts/contracts-upgrade
SafeCastUpgradeable.toInt184(int256) (lib/openzeppelin-contracts/contracts-upgrade
SafeCastUpgradeable.toInt192(int256) (lib/openzeppelin-contracts/contracts-upgrade
SafeCastUpgradeable.toInt200(int256) (lib/openzeppelin-contracts/contracts-upgrade
SafeCastUpgradeable.toInt208(int256) (lib/openzeppelin-contracts/contracts-upgrade
SafeCastUpgradeable.toInt216(int256) (lib/openzeppelin-contracts/contracts-upgrade
SafeCastUpgradeable.toInt224(int256) (lib/openzeppelin-contracts/contracts-upgrade
SafeCastUpgradeable.toInt232(int256) (lib/openzeppelin-contracts/contracts-upgrade
SafeCastUpgradeable.toInt24(int256) (lib/openzeppelin-contracts/contracts-upgraded
SafeCastUpgradeable.toInt240(int256) (lib/openzeppelin-contracts/contracts-upgrade
SafeCastUpgradeable.toInt248(int256) (lib/openzeppelin-contracts/contracts-upgrade
SafeCastUpgradeable.toInt256(uint256) (lib/openzeppelin-contracts/contracts-upgrad
SafeCastUpgradeable.toInt32(int256) (lib/openzeppelin-contracts/contracts-upgraded
SafeCastUpgradeable.toInt40(int256) (lib/openzeppelin-contracts/contracts-upgraded
SafeCastUpgradeable.toInt48(int256) (lib/openzeppelin-contracts/contracts-upgraded
SafeCastUpgradeable.toInt56(int256) (lib/openzeppelin-contracts/contracts-upgraded
SafeCastUpgradeable.toInt64(int256) (lib/openzeppelin-contracts/contracts-upgraded
SafeCastUpgradeable.toInt72(int256) (lib/openzeppelin-contracts/contracts-upgraded
SafeCastUpgradeable.toInt8(int256) (lib/openzeppelin-contracts/contracts-upgradeab
SafeCastUpgradeable.toInt80(int256) (lib/openzeppelin-contracts/contracts-upgraded
SafeCastUpgradeable.toInt88(int256) (lib/openzeppelin-contracts/contracts-upgraded
SafeCastUpgradeable.toInt96(int256) (lib/openzeppelin-contracts/contracts-upgraded
SafeCastUpgradeable.toUint104(uint256) (lib/openzeppelin-contracts/contracts-upgra
SafeCastUpgradeable.toUint112(uint256) (lib/openzeppelin-contracts/contracts-upgra
SafeCastUpgradeable.toUint120(uint256) (lib/openzeppelin-contracts/contracts-upgra
SafeCastUpgradeable.toUint128(uint256) (lib/openzeppelin-contracts/contracts-upgra
SafeCastUpgradeable.toUint136(uint256) (lib/openzeppelin-contracts/contracts-upgra
SafeCastUpgradeable.toUint144(uint256) (lib/openzeppelin-contracts/contracts-upgra
SafeCastUpgradeable.toUint152(uint256) (lib/openzeppelin-contracts/contracts-upgra
SafeCastUpgradeable.toUint16(uint256) (lib/openzeppelin-contracts/contracts-upgrad
SafeCastUpgradeable.toUint160(uint256) (lib/openzeppelin-contracts/contracts-upgra
SafeCastUpgradeable.toUint168(uint256) (lib/openzeppelin-contracts/contracts-upgra
SafeCastUpgradeable.toUint176(uint256) (lib/openzeppelin-contracts/contracts-upgra
SafeCastUpgradeable.toUint184(uint256) (lib/openzeppelin-contracts/contracts-upgra
SafeCastUpgradeable.toUint192(uint256) (lib/openzeppelin-contracts/contracts-upgra
SafeCastUpgradeable.toUint200(uint256) (lib/openzeppelin-contracts/contracts-upgra
SafeCastUpgradeable.toUint208(uint256) (lib/openzeppelin-contracts/contracts-upgra
SafeCastUpgradeable.toUint216(uint256) (lib/openzeppelin-contracts/contracts-upgra
SafeCastUpgradeable.toUint224(uint256) (lib/openzeppelin-contracts/contracts-upgra
```

- TransfersAdapter.sol

```
TransfersAdapter._strictTransferFrom(address,IERC20Upgradeable,uint256) (src/stakin
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#arbitrary-

StakingStorage._depositedRewardsFund (src/staking/StakingStorage.sol#71) is never i
        - StakingStorage.depositedRewardsFund() (src/staking/StakingStorage.sol#86-
StakingStorage._poolsArchive (src/staking/StakingStorage.sol#74) is never initializ
        - StakingStorage.poolMetadata(bytes32) (src/staking/StakingStorage.sol#93-9
        - StakingStorage.isPoolOwner(address,bytes32) (src/staking/StakingStorage.s
        - StakingStorage.poolExists(bytes32) (src/staking/StakingStorage.sol#107-10
        - StakingStorage.poolOperative(bytes32) (src/staking/StakingStorage.sol#114
        - StakingStorage.poolJailed(bytes32) (src/staking/StakingStorage.sol#129-13
        - StakingStorage.getPoolStake(bytes32) (src/staking/StakingStorage.sol#137-
        - StakingStorage.getDelegatorStake(address,bytes32) (src/staking/StakingSto
        - StakingStorage.getDelegatorCurrency(address,bytes32,IERC20Upgradeable) (s
        - StakingStorage.getDelegatorPendingWithdrawal(address,bytes32) (src/stakin
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitiali

MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-up
        - inverse = (3 * denominator) ^ 2 (lib/openzeppelin-contracts/contracts-upg
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-up
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contract
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-up
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contract
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-up
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contract
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-up
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contract
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-up
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contract
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-up
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contract
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
        - prod0 = prod0 / twos (lib/openzeppelin-contracts/contracts-upgradeable/ut
        - result = prod0 * inverse (lib/openzeppelin-contracts/contracts-upgradeabl
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-bef

TransfersAdapter._strictTransferFrom(address,IERC20Upgradeable,uint256) (src/stakin
        - require(bool,string)(erc20.balanceOf(address(this)) == previousBalance +
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-

TransfersAdapter._strictTransferFrom(address,StakingStorage.Currency[]).it (src/sta
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitiali
```

```
DepositsAdapter._loadDeposit(DepositsAdapter.DeferredDeposit).currentEpoch (src/staki
        - StakingEpochManager.currentEpoch() (src/staking/epochs/StakingEpochManager.
DepositsAdapter._previousEpochBalance(DepositsAdapter.DeferredDeposit).currentEpoch (
        - StakingEpochManager.currentEpoch() (src/staking/epochs/StakingEpochManager.
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variab

StakingEpochManager._endEpoch() (src/staking/epochs/StakingEpochManager.sol#69-76) us
        Dangerous comparisons:
        - require(bool,string)(block.timestamp >= getEpochEarliestEndTime(),E301) (sr
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timest

AddressUpgradeable._revert(bytes,string) (lib/openzeppelin-contracts/contracts-upgrad
        - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressU
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts
        - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/Mat
        - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/Mat
        - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/Mat
StakingStorage.filterPoolIdsIdle(bytes32[]) (src/staking/StakingStorage.sol#185-197)
        - INLINE ASM (src/staking/StakingStorage.sol#194-196)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usa

Different versions of Solidity are used:
        - Version used: ['^0.8.0', '^0.8.1', '^0.8.11', '^0.8.2']
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/token/ERC20/IERC20
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/token/ERC20/extens
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/token/ERC20/extens
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/token/ERC20/utils/
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/MathUpg
        - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/SafeCas
        - ^0.8.1 (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgra
        - ^0.8.11 (src/staking/StakingStorage.sol#2)
        - ^0.8.11 (src/staking/config/Configurable.sol#2)
        - ^0.8.11 (src/staking/config/StakingConstants.sol#2)
        - ^0.8.11 (src/staking/deposits/DepositsAdapter.sol#2)
        - ^0.8.11 (src/staking/deposits/TransfersAdapter.sol#2)
        - ^0.8.11 (src/staking/epochs/StakingEpochManager.sol#2)
        - ^0.8.2 (lib/openzeppelin-contracts/contracts-upgradeable/proxy/utils/Initia
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pr

AddressUpgradeable._revert(bytes,string) (lib/openzeppelin-contracts/contracts-upgrad
AddressUpgradeable.functionCall(address,bytes) (lib/openzeppelin-contracts/contracts-
AddressUpgradeable.functionCall(address,bytes,string) (lib/openzeppelin-contracts/con
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (lib/openzeppelin-con
AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (lib/openzeppe
AddressUpgradeable.functionStaticCall(address,bytes) (lib/openzeppelin-contracts/cont
AddressUpgradeable.functionStaticCall(address,bytes,string) (lib/openzeppelin-contrac
AddressUpgradeable.isContract(address) (lib/openzeppelin-contracts/contracts-upgradea
AddressUpgradeable.sendValue(address,uint256) (lib/openzeppelin-contracts/contracts-u
```

```
DepositsAdapter._storeDeposit(DepositsAdapter.DeferredDeposit,DepositsAdapter.Defer
Initializable._disableInitializers() (lib/openzeppelin-contracts/contracts-upgradea
Initializable._getInitializedVersion() (lib/openzeppelin-contracts/contracts-upgrad
Initializable._isInitializing() (lib/openzeppelin-contracts/contracts-upgradeable/p
MathUpgradeable.average(uint256,uint256) (lib/openzeppelin-contracts/contracts-upgr
MathUpgradeable.ceilDiv(uint256,uint256) (lib/openzeppelin-contracts/contracts-upgr
MathUpgradeable.log10(uint256) (lib/openzeppelin-contracts/contracts-upgradeable/ut
MathUpgradeable.log10(uint256,MathUpgradeable.Rounding) (lib/openzeppelin-contracts
MathUpgradeable.log2(uint256) (lib/openzeppelin-contracts/contracts-upgradeable/uti
MathUpgradeable.log2(uint256,MathUpgradeable.Rounding) (lib/openzeppelin-contracts/
MathUpgradeable.log256(uint256) (lib/openzeppelin-contracts/contracts-upgradeable/u
MathUpgradeable.log256(uint256,MathUpgradeable.Rounding) (lib/openzeppelin-contract
MathUpgradeable.min(uint256,uint256) (lib/openzeppelin-contracts/contracts-upgradea
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
MathUpgradeable.mulDiv(uint256,uint256,uint256,MathUpgradeable.Rounding) (lib/openz
MathUpgradeable.sqrt(uint256) (lib/openzeppelin-contracts/contracts-upgradeable/uti
MathUpgradeable.sqrt(uint256,MathUpgradeable.Rounding) (lib/openzeppelin-contracts/
SafeCastUpgradeable.toInt104(int256) (lib/openzeppelin-contracts/contracts-upgradea
SafeCastUpgradeable.toInt112(int256) (lib/openzeppelin-contracts/contracts-upgradea
SafeCastUpgradeable.toInt120(int256) (lib/openzeppelin-contracts/contracts-upgradea
SafeCastUpgradeable.toInt128(int256) (lib/openzeppelin-contracts/contracts-upgradea
SafeCastUpgradeable.toInt136(int256) (lib/openzeppelin-contracts/contracts-upgradea
SafeCastUpgradeable.toInt144(int256) (lib/openzeppelin-contracts/contracts-upgradea
SafeCastUpgradeable.toInt152(int256) (lib/openzeppelin-contracts/contracts-upgradea
SafeCastUpgradeable.toInt16(int256) (lib/openzeppelin-contracts/contracts-upgradeab
SafeCastUpgradeable.toInt160(int256) (lib/openzeppelin-contracts/contracts-upgradea
SafeCastUpgradeable.toInt168(int256) (lib/openzeppelin-contracts/contracts-upgradea
SafeCastUpgradeable.toInt176(int256) (lib/openzeppelin-contracts/contracts-upgradea
SafeCastUpgradeable.toInt184(int256) (lib/openzeppelin-contracts/contracts-upgradea
SafeCastUpgradeable.toInt192(int256) (lib/openzeppelin-contracts/contracts-upgradea
SafeCastUpgradeable.toInt200(int256) (lib/openzeppelin-contracts/contracts-upgradea
SafeCastUpgradeable.toInt208(int256) (lib/openzeppelin-contracts/contracts-upgradea
SafeCastUpgradeable.toInt216(int256) (lib/openzeppelin-contracts/contracts-upgradea
SafeCastUpgradeable.toInt224(int256) (lib/openzeppelin-contracts/contracts-upgradea
SafeCastUpgradeable.toInt232(int256) (lib/openzeppelin-contracts/contracts-upgradea
SafeCastUpgradeable.toInt24(int256) (lib/openzeppelin-contracts/contracts-upgradeab
SafeCastUpgradeable.toInt240(int256) (lib/openzeppelin-contracts/contracts-upgradea
SafeCastUpgradeable.toInt248(int256) (lib/openzeppelin-contracts/contracts-upgradea
SafeCastUpgradeable.toInt256(uint256) (lib/openzeppelin-contracts/contracts-upgrade
SafeCastUpgradeable.toInt32(int256) (lib/openzeppelin-contracts/contracts-upgradeab
SafeCastUpgradeable.toInt40(int256) (lib/openzeppelin-contracts/contracts-upgradeab
SafeCastUpgradeable.toInt48(int256) (lib/openzeppelin-contracts/contracts-upgradeab
SafeCastUpgradeable.toInt56(int256) (lib/openzeppelin-contracts/contracts-upgradeab
SafeCastUpgradeable.toInt64(int256) (lib/openzeppelin-contracts/contracts-upgradeab
SafeCastUpgradeable.toInt72(int256) (lib/openzeppelin-contracts/contracts-upgradeab
SafeCastUpgradeable.toInt8(int256) (lib/openzeppelin-contracts/contracts-upgradeabl
SafeCastUpgradeable.toInt80(int256) (lib/openzeppelin-contracts/contracts-upgradeab
SafeCastUpgradeable.toInt88(int256) (lib/openzeppelin-contracts/contracts-upgradeab
```

- StakingConstants.sol

```
Pragma version^0.8.11 (src/staking/config/StakingConstants.sol#2) allows old versio
solc-0.8.11 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-

Variable StakingConstants.OWNER_STAKE_INFLUENCE_DEN (src/staking/config/StakingCons
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-n
```

- Configurable.sol

```
Configurable._setConfig(bytes32,uint256) (src/staking/config/Configurable.sol#26-30
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.11 (src/staking/config/Configurable.sol#2) allows old versions
solc-0.8.11 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-

Variable Configurable.__gap (src/staking/config/Configurable.sol#36) is not in mixe
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformanc

Configurable.__gap (src/staking/config/Configurable.sol#36) is never used in Config
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-sta
```

The above output was reviewed by Halborn and the findings were either included in the report or confirmed to be false positives.

8.2 AUTOMATED SECURITY SCAN

Description:

Halborn used automated security scanners to assist with detection of well-known security issues and to identify low-hanging fruits on the targets for this engagement. Among the tools used was MythX, a security analysis service for Ethereum smart contracts. MythX performed a scan on the smart contracts and sent the compiled results to the analyzers in order to locate any vulnerabilities.

Results:

Report for src/staking/Staking.sol
https://dashboard.mythx.io/#/console/analyses/fd75e263-e339-4277-8f59-f8ed29c591cc

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

Report for src/staking/StakingStorage.sol
https://dashboard.mythx.io/#/console/analyses/ad3620d4-49fc-46fe-bd89-bc9efe6d64cc

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

Report for src/staking/deposits/DelegationsAdapter.sol
https://dashboard.mythx.io/#/console/analyses/b7531772-cdcc-41ac-b954-4ec9664f4ccb

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

Report for src/staking/deposits/DepositsAdapter.sol
https://dashboard.mythx.io/#/console/analyses/c4dacf8c-55e8-433e-a623-301bbfea8ccc

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

Report for src/staking/deposits/TransfersAdapter.sol
https://dashboard.mythx.io/#/console/analyses/e3128c2a-0b9e-44a8-bb9c-01d879867dab

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

Report for src/staking/epochs/StakingEpochManager.sol
https://dashboard.mythx.io/#/console/analyses/258652f7-e218-4f17-a130-2443edf56c58

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

Report for src/staking/rewards/DelegatorRewardsAdapter.sol
https://dashboard.mythx.io/#/console/analyses/fe639c1c-272c-4113-98db-44335732b5f6

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

Report for src/staking/rewards/Finalizer.sol
https://dashboard.mythx.io/#/console/analyses/d49fef8b-d5d1-490d-a00f-3c95979b4bcd

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

All the issues flagged by MythX were found to be either false positives or issues already reported.

THANK YOU FOR CHOOSING

// HALBORN