



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	4
CONTACTS	5
1 EXECUTIVE OVERVIEW	6
1.1 INTRODUCTION	7
1.2 AUDIT SUMMARY	7
1.3 SCOPE	8
1.4 TEST APPROACH & METHODOLOGY	9
2 RISK METHODOLOGY	10
2.1 EXPLOITABILITY	11
2.2 IMPACT	12
2.3 SEVERITY COEFFICIENT	14
3 ASSESSMENT SUMMARY & FINDINGS OVERVIEW	16
4 FINDINGS & TECH DETAILS	17
4.1 (HAL-01) UNUSABLE STAKING POOL AFTER POOL OWNER UNSTAKE - LOW(2.5)	19
Description	19
Code Location	19
Proof Of Concept	20
BVSS	22
Recommendation	22
Remediation Plan	22
4.2 (HAL-02) NOT ALL ROLES ARE SET UP ON INITIALIZATION - LOW(2.5)	23
Description	23
Code Location	23

BVSS	24
Recommendation	24
Remediation Plan	24
4.3 (HAL-03) USE CUSTOM ERRORS TO SAVE GAS - INFORMATIONAL(0.0)	25
Description	25
Code Location	25
BVSS	25
Recommendation	25
Remediation Plan	25
4.4 (HAL-04) INEFFICIENT CONDITION IN THE FINALIZEPOOL INTERNAL FUNCTION - INFORMATIONAL(0.0)	26
Description	26
Code Location	26
BVSS	26
Proof of Concept	27
Recommendation	27
Remediation Plan	27
5 RECOMMENDATIONS OVERVIEW	27
6 RE-TEST	30
6.1 BWARE01 - THERE IS NO CAP FOR THE POOL OWNERS STAKE	31
Description	31
Code Location	31
Recommendation	34
Remediation Plan	34
7 MANUAL TESTING	35
7.1 TESTING METHODOLOGY	36

7.2 TESTS	37
Unit testing of all the components involved in the protocol	37
Integration Testing of use cases integrating all the stakeholders	37
Edge Case Scenario	40
OVERVIEW OF THE RESULTS OF THE TESTS PERFORMED	41
8 AUTOMATED TESTING	41
8.1 STATIC ANALYSIS REPORT	43
Description	43
Slither Results	44
8.2 AUTOMATED SECURITY SCAN	72
Description	72
Results	73

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 were 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)	1
	Specific (AO:S)	0.2
Attack Cost (AC)	Low (AC:L)	1
	Medium (AC:M)	0.67
	High (AC:H)	0.33
Attack Complexity (AX)	Low (AX:L)	1
	Medium (AX:M)	0.67
	High (AX:H)	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)	1
	Partial (R:P)	0.5
	Full (R:F)	0.25
Scope (s)	Changed (S:C)	1.25
	Unchanged (S:U)	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

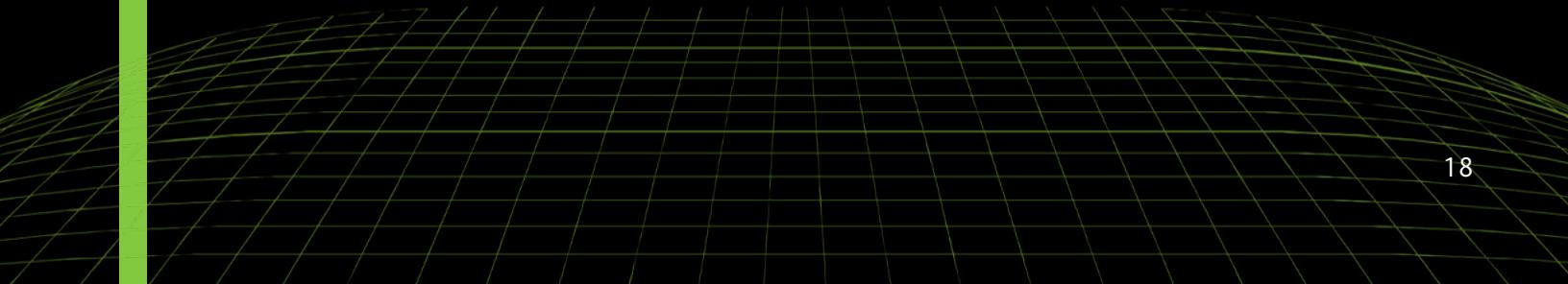
CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	0	2	2

EXECUTIVE OVERVIEW

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

Code Section - StakingStorage.sol#L116

Listing 2: StakingStorage.sol (Line 116)

```

114     function poolOperative(bytes32 poolId) public view returns (
115         bool) {
116         Pool storage poolPtr = _poolsArchive[poolId];
117         return poolExists(poolId) && poolPtr.delegations[poolPtr.
118         meta.owner].deposited.nextEpochBalance != 0;
119     }

```

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

```

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

```

[0] VM::startPrank(alice: [0x61A1D7fd8C9bbd82932D99FD47bD2581C23b08c])
|_ ← ()
[22529] MockERC20::approve(Staking: [0x01cb6c0007A0d23cC00c5A448Ce4452502b73aC9], 10000000000000000000)
|_ emit Approval(owner: alice: [0x61A1D7fd8C9bbd82932D99FD47bD2581C23b08c], spender: Staking: [0x01cb6c0007A0d23cC00
|_ ← true
[10136] Staking::delegate(0x61a1d7fd8c9bbd82932d99fd47bd2581c23b08c00000000000000000000000000000000, [0x2efD08c8Fc1050Ed24f5
|_ [585] MockERC20::balanceOf(Staking: [0x01cb6c0007A0d23cC00c5A448Ce4452502b73aC9]) [staticcall]
|_ ← 30000000000000000000000000000000
|_ [4659] MockERC20::transferFrom(alice: [0x61A1D7fd8C9bbd82932D99FD47bD2581C23b08c], Staking: [0x01cb6c0007A0d23cC0
|_ emit Approval(owner: alice: [0x61A1D7fd8C9bbd82932D99FD47bD2581C23b08c], spender: Staking: [0x01cb6c0007A0d23cC0
|_ emit Transfer(from: alice: [0x61A1D7fd8C9bbd82932D99FD47bD2581C23b08c], to: Staking: [0x01cb6c0007A0d23cC00c5
|_ ← true
|_ [585] MockERC20::balanceOf(Staking: [0x01cb6c0007A0d23cC00c5A448Ce4452502b73aC9]) [staticcall]
|_ ← 30100000000000000000000000000000
|_ ← "E104"
|_ ← "E104"

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

ling tests:
ountered 1 failing test in test/MyTest2.t.sol:MyTest2
IL 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:

A0: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:

A0: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:

A0: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.

```

└ [33031] Staking::finalizePools([0x61a1d7fd8c9bbd82932d99dfd47bd2581c23b08c0000000000000000100000001], [1000
|   └ emit Event(totalStake: 61000000000000000000000000000000, ownerStake: 60000000000000000000000000000000, poolsToFinalizePre
|   └ emit Event(totalStake: 61000000000000000000000000000000, ownerStake: 60000000000000000000000000000000, poolsToFinalizePre
|   └ emit Event(totalStake: 61000000000000000000000000000000, ownerStake: 60000000000000000000000000000000, poolsToFinalizePre
|   └ emit FinalizedPool(poolId: 0x61a1d7fd8c9bbd82932d99dfd47bd2581c23b08c0000000000000000100000001, perfor
|   └ ↵ () 
└ [32776] Staking::finalizePools([0x61a1d7fd8c9bbd82932d99dfd47bd2581c23b08c0000000000000000100000001], [1000
|   └ emit Event(totalStake: 61000000000000000000000000000000, ownerStake: 60000000000000000000000000000000, poolsToFinalizePrevi
|   └ emit Event(totalStake: 61000000000000000000000000000000, ownerStake: 60000000000000000000000000000000, poolsToFinalizePrevi
|   └ emit Event(totalStake: 61000000000000000000000000000000, ownerStake: 60000000000000000000000000000000, poolsToFinalizePrevi
|   └ emit FinalizedPool(poolId: 0x61a1d7fd8c9bbd82932d99dfd47bd2581c23b08c0000000000000000100000001, performa
|   └ ↵ () 
--- ...

```

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) /
177             PERCENTAGE_SCALING_FACTOR;
178         baseReward = (baseReward * performance) /
179             PERCENTAGE_SCALING_FACTOR;
180     }

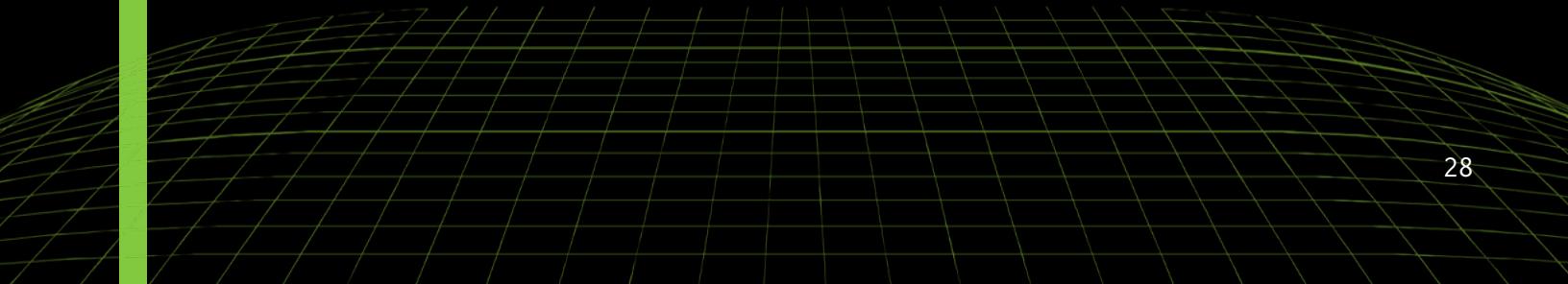
```

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,
145         uint16 ownerBaseApy) private {
146         require(performance <= PERCENTAGE_SCALING_FACTOR &&
147             ownerBaseApy <= MAX_OWNER_BASE_APY, "E401");
148
149         uint64 currentEpoch = currentEpoch();
150         CumulativeReward memory cumulativeReward =
151             poolCumulativeReward(poolId, currentEpoch);
152         if (cumulativeReward.finalized) {
153             // pool already finalized for the previous epoch
154             return;
155         }
156         Pool storage poolPtr = _poolsArchive[poolId];
157         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.toInt248(
```

```
189             ((poolReward - commissionReward) *
190             TOKEN_FRACTION_SCALING_FACTOR) / totalStake
191         );
192         // set cumulative reward at this epoch and implicitly mark
193         // the pool as finalized
194         _poolCumulativeReward[poolId][currentEpoch] =
195         cumulativeReward;
196         // reward the pool owner its commission + base rewards
197         _unclaimedRewards[poolPtr.meta.owner] += (baseReward +
198         commissionReward);
199         // remove this pool from the finalization accounting of
200         // previous epoch
201         _poolsToFinalizePreviousEpoch -= 1;
202         emit FinalizedPool(poolId, performance, ownerBaseApy,
203         cumulativeReward, poolReward, baseReward);
204     }
```

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 ----> 30000000000000000000000000000000
y token ----> 30000000000000000000000000000000
```

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 undelегates.
13. We run more epochs.
14. Edgar can claim.
15. We run more epochs.
16. Edgar undelегates 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: 15000000000000000000000000000000
Alice T token: 15000000000000000000000000000000
Alice BWR token: 0
Edgar X token: 15000000000000000000000000000000
Edgar Y token: 15000000000000000000000000000000
Edgar BWR token: 50000000000000000000000000000000
-----
--- STAGE 1 ---
-----
Alice X token: 12000000000000000000000000000000
Alice T token: 12000000000000000000000000000000
Alice BWR token: 0
Edgar X token: 15000000000000000000000000000000
Edgar Y token: 15000000000000000000000000000000
Edgar BWR token: 0
pool terminated? - true
-----
--- STAGE 2 ---
-----
Alice X token: 15000000000000000000000000000000
Alice T token: 15000000000000000000000000000000
Alice BWR token: 573487881981032659
Edgar X token: 15000000000000000000000000000000
Edgar Y token: 15000000000000000000000000000000
Edgar BWR token: 50000000000000000000000000000000
```

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_tt2C() (gas: 3168345)
Logs:
Alice X token: 30000000000000000000000000000000
Alice T token: 30000000000000000000000000000000
Alice BWR token: 852515068493150674
Edgar X token: 30000000000000000000000000000000
Edgar Y token: 30000000000000000000000000000000
Edgar BWR token: 845285563751317167
Bobby X token: 0
Bobby Y token: 0
Bobby BWR token: 19000000000000000000000000000000
```

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.

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_poolJailedPoCC() (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-upgr
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-i
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-i
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-i
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-i
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-i
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-i
Finalizer._finalizePool(bytes32,uint16,uint16) (src/staking/rewards/Finalizer.sol#144-
    - baseReward = (((ownerStake * ownerBaseApy) / PERCENTAGE_SCALING_FACTOR) * pi
Finalizer._finalizePool(bytes32,uint16,uint16) (src/staking/rewards/Finalizer.sol#144-
    - baseReward = (((ownerStake * ownerBaseApy) / PERCENTAGE_SCALING_FACTOR) * pi
    - 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_S
Finalizer.computePoolEpochRewards(uint256,uint256) (src/staking/rewards/Finalizer.sol#
    - earningStake = (ownerStake * (totalStake - earningStake)) / saturationCap (: 
    - 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 + pledge
Finalizer.computePoolEpochRewards(uint256,uint256) (src/staking/rewards/Finalizer.sol#
    - (((earningStake * getConfig(POOL_EARNING_STAKE_APY)) / PERCENTAGE_SCALING_F
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

DelegatorRewardsAdapter._computeDelegatorReward(address,bytes32) (src/staking/rewards,
    - delegation.currentEpoch == 0 (src/staking/rewards/DelegatorRewardsAdapter.sol#144-
DelegatorRewardsAdapter._computeDelegatorReward(address,bytes32) (src/staking/rewards,
    - delegation.currentEpoch == currentEpoch (src/staking/rewards/DelegatorRewards

```



```

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/stak
    - delegation.currencies[currency.erc20] -= currency.amount (src/stak
- _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-level)
  - erc20.safeTransferFrom(account,address(this),amount) (src/staking/Staking.sol#21)
  - (success,returndata) = target.call{value: value}(data) (lib/openzeppelin-solidity/contracts/token/erc20/SafeERC20.sol#10)
External calls sending eth:
- _strictTransferFrom(_msgSender(),currencies) (src/staking/Staking.sol#21)
  - (success,returndata) = target.call{value: value}(data) (lib/openzeppelin-solidity/contracts/token/erc20/SafeERC20.sol#10)
Event emitted after the call(s):
- CollectDelegatorRewards(delegator,poolId,rewards) (src/staking/rewards/Distributor.sol#14)
  - _stake(poolId,_msgSender(),currencies) (src/staking/Staking.sol#21)
- 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/Staking.sol#21)
  - returndata = address(token).functionCall(data, SafeERC20: low-level)
  - erc20.safeTransferFrom(account,address(this),amount) (src/staking/Staking.sol#21)
  - (success,returndata) = target.call{value: value}(data) (lib/openzeppelin-solidity/contracts/token/erc20/SafeERC20.sol#10)
External calls sending eth:
- _strictTransferFrom(_msgSender(),rewardsToken(),amount) (src/staking/Staking.sol#21)
  - (success,returndata) = target.call{value: value}(data) (lib/openzeppelin-solidity/contracts/token/erc20/SafeERC20.sol#10)
Event emitted after the call(s):
- DepositRewards(amount) (src/staking/Staking.sol#148)
Reentrancy in Staking.executePendingWithdrawal(bytes32) (src/staking/Staking.sol#21)
External calls:
- _strictTransferTo(_msgSender(),withdrawal.currencies[it].erc20,withdrawal.amount)
  - returndata = address(token).functionCall(data, SafeERC20: low-level)
  - erc20.safeTransfer(account,amount) (src/staking/deposits/Transfers.sol#10)
  - (success,returndata) = target.call{value: value}(data) (lib/openzeppelin-solidity/contracts/token/erc20/SafeERC20.sol#10)
External calls sending eth:
- _strictTransferTo(_msgSender(),withdrawal.currencies[it].erc20,withdrawal.amount)
  - (success,returndata) = target.call{value: value}(data) (lib/openzeppelin-solidity/contracts/token/erc20/SafeERC20.sol#10)
Event emitted after the call(s):
- ExecutePendingWithdrawal(_msgSender(),poolId) (src/staking/Staking.sol#31)
Reentrancy in Staking.unjail(bytes32) (src/staking/Staking.sol#358-365):
External calls:
- depositRewards(getConfig(POOL_UNJAILING_FEE)) (src/staking/Staking.sol#31)
  - returndata = address(token).functionCall(data, SafeERC20: low-level)
  - erc20.safeTransferFrom(account,address(this),amount) (src/staking/Staking.sol#31)
  - (success,returndata) = target.call{value: value}(data) (lib/openzeppelin-solidity/contracts/token/erc20/SafeERC20.sol#10)
External calls sending eth:
- depositRewards(getConfig(POOL_UNJAILING_FEE)) (src/staking/Staking.sol#31)
  - (success,returndata) = target.call{value: value}(data) (lib/openzeppelin-solidity/contracts/token/erc20/SafeERC20.sol#10)
Event emitted after the call(s):
- UnjailPool(poolId) (src/staking/Staking.sol#364)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy

```

```

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/AddressUpgra
StringsUpgradeable.toString(uint256) (lib/openzeppelin-contracts/contracts-upgradea
    - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/StringUpgra
    - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/StringUpgra
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac
    - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/MathUpg
    - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/MathUpg
    - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/MathUpg
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-upgradeable

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/AccessControlUpg
- ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/access/IAccessControlUp
- ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/security/PausableUpg
- ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/token/ERC20/IERC20Upg
- ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/ContextUpgradab
- ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/StringsUpgradab

```

```

- ^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/cont
AddressUpgradeable.functionStaticCall(address,bytes,string) (lib/openzeppelin-contro
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/util
ERC165Upgradeable.__ERC165_init() (lib/openzeppelin-contracts/contracts-upgradeable,
ERC165Upgradeable.__ERC165_init_unchained() (lib/openzeppelin-contracts/contracts-u
Initializable._disableInitializers() (lib/openzeppelin-contracts/contracts-upgradeab
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/
```

- StakingStorage.sol

```

StakingStorage._depositedRewardsFund (src/staking/StakingSTORAGE.sol#71) is never initialized
  - StakingStorage.depositedRewardsFund() (src/staking/StakingSTORAGE.sol#86-100)
StakingStorage._poolsArchive (src/staking/StakingSTORAGE.sol#74) is never initialized
  - StakingStorage.poolMetadata(bytes32) (src/staking/StakingSTORAGE.sol#93-97)
  - StakingStorage.isPoolOwner(address,bytes32) (src/staking/StakingSTORAGE.sol#108-112)
  - StakingStorage.poolExists(bytes32) (src/staking/StakingSTORAGE.sol#107-110)
  - StakingStorage.poolOperative(bytes32) (src/staking/StakingSTORAGE.sol#114-118)
  - StakingStorage.poolJailed(bytes32) (src/staking/StakingSTORAGE.sol#129-133)
  - StakingStorage.getPoolStake(bytes32) (src/staking/StakingSTORAGE.sol#137-141)
  - StakingStorage.getDelegatorStake(address,bytes32) (src/staking/StakingSTORAGE.sol#145-149)
  - StakingStorage.getDelegatorCurrency(address,bytes32,IERC20Upgradeable) (src/staking/StakingSTORAGE.sol#153-157)
  - StakingStorage.getDelegatorPendingWithdrawal(address,bytes32) (src/staking/StakingSTORAGE.sol#161-165)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitializ
```

```

MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts/math/MathUpgradeable.sol#10-14)
  - denominator = denominator / twos (lib/openzeppelin-contracts/contracts/math/MathUpgradeable.sol#11-12)
  - inverse = (3 * denominator) ^ 2 (lib/openzeppelin-contracts/contracts/math/MathUpgradeable.sol#13-14)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts/math/MathUpgradeable.sol#18-22)
  - denominator = denominator / twos (lib/openzeppelin-contracts/contracts/math/MathUpgradeable.sol#19-20)
  - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contracts/math/MathUpgradeable.sol#21-22)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts/math/MathUpgradeable.sol#26-30)
  - denominator = denominator / twos (lib/openzeppelin-contracts/contracts/math/MathUpgradeable.sol#27-28)
  - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contracts/math/MathUpgradeable.sol#29-30)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts/math/MathUpgradeable.sol#34-38)
  - denominator = denominator / twos (lib/openzeppelin-contracts/contracts/math/MathUpgradeable.sol#35-36)
  - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contracts/math/MathUpgradeable.sol#37-38)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts/math/MathUpgradeable.sol#42-46)
  - denominator = denominator / twos (lib/openzeppelin-contracts/contracts/math/MathUpgradeable.sol#43-44)
  - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contracts/math/MathUpgradeable.sol#45-46)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts/math/MathUpgradeable.sol#50-54)
  - denominator = denominator / twos (lib/openzeppelin-contracts/contracts/math/MathUpgradeable.sol#51-52)
  - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contracts/math/MathUpgradeable.sol#53-54)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts/math/MathUpgradeable.sol#58-62)
  - denominator = denominator / twos (lib/openzeppelin-contracts/contracts/math/MathUpgradeable.sol#59-60)
  - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contracts/math/MathUpgradeable.sol#61-62)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts/math/MathUpgradeable.sol#66-70)
  - denominator = denominator / twos (lib/openzeppelin-contracts/contracts/math/MathUpgradeable.sol#67-68)
  - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contracts/math/MathUpgradeable.sol#69-70)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts/math/MathUpgradeable.sol#74-78)
  - denominator = denominator / twos (lib/openzeppelin-contracts/contracts/math/MathUpgradeable.sol#75-76)
  - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contracts/math/MathUpgradeable.sol#77-78)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts/math/MathUpgradeable.sol#82-86)
  - denominator = denominator / twos (lib/openzeppelin-contracts/contracts/math/MathUpgradeable.sol#83-84)
  - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contracts/math/MathUpgradeable.sol#85-86)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts/math/MathUpgradeable.sol#90-94)
  - denominator = denominator / twos (lib/openzeppelin-contracts/contracts/math/MathUpgradeable.sol#91-92)
  - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contracts/math/MathUpgradeable.sol#93-94)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiplication
```

```

DepositsAdapter._loadDeposit(DepositsAdapter.DeferredDeposit).currentEpoch (src/staking/epochs/DepositsAdapter.sol#10-14)
  - StakingEpochManager.currentEpoch() (src/staking/epochs/StakingEpochManager.sol#11-15)
DepositsAdapter._previousEpochBalance(DepositsAdapter.DeferredDeposit).currentEpoch (src/staking/epochs/DepositsAdapter.sol#18-22)
  - StakingEpochManager.currentEpoch() (src/staking/epochs/StakingEpochManager.sol#19-23)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-use

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-comparisons
```

```

AddressUpgradeable._revert(bytes,string) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#194-196)
    - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#194-196)
        - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/MathUpgradeable.sol#194-196)
            - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/MathUpgradeable.sol#194-196)
                - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/MathUpgradeable.sol#194-196)
                    - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/MathUpgradeable.sol#194-196)
                        - INLINE ASM (src/staking/StakingSTORAGE.sol#185-196)
                            - INLINE ASM (src/staking/StakingSTORAGE.sol#194-196)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-instrumentation

```

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.sol#194-196)
- ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/MathUpgradeable.sol#194-196)
- ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/MathUpgradeable.sol#194-196)
- ^0.8.1 (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/MathUpgradeable.sol#194-196)
- ^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/Initializable.sol#194-196)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#different-solidity-versions>

```

AddressUpgradeable._revert(bytes,string) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#194-196)
AddressUpgradeable.functionCall(address,bytes) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#194-196)
AddressUpgradeable.functionCall(address,bytes,string) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#194-196)
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#194-196)
AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#194-196)
AddressUpgradeable.functionStaticCall(address,bytes) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#194-196)
AddressUpgradeable.functionStaticCall(address,bytes,string) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#194-196)
AddressUpgradeable.isContract(address) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#194-196)
AddressUpgradeable.sendValue(address,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#194-196)
AddressUpgradeable.verifyCallResult(bool,bytes,string) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#194-196)
AddressUpgradeable.verifyCallResultFromTarget(address,bool,bytes,string) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#194-196)
Configurable._setConfig(bytes32,uint256) (src/staking/config/Configurable.sol#26-31)
DepositsAdapter._decreaseNextBalance(DepositsAdapter.DeferredDeposit,uint256) (src/staking/deposits/DepositsAdapter.sol#194-196)
DepositsAdapter._increaseNextBalance(DepositsAdapter.DeferredDeposit,uint256) (src/staking/deposits/DepositsAdapter.sol#194-196)
DepositsAdapter._previousEpochBalance(DepositsAdapter.DeferredDeposit) (src/staking/deposits/DepositsAdapter.sol#194-196)
DepositsAdapter._storeDeposit(DepositsAdapter.DeferredDeposit,DepositsAdapter.DeferredDeposit) (src/staking/deposits/DepositsAdapter.sol#194-196)
Initializable._disableInitializers() (lib/openzeppelin-contracts/contracts-upgradeable/utils/Initializable.sol#194-196)
Initializable._getInitializedVersion() (lib/openzeppelin-contracts/contracts-upgradeable/utils/Initializable.sol#194-196)
Initializable._isInitializing() (lib/openzeppelin-contracts/contracts-upgradeable/utils/Initializable.sol#194-196)
MathUpgradeable.average(uint256,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#194-196)
MathUpgradeable.ceilDiv(uint256,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#194-196)
MathUpgradeable.log10(uint256) (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#194-196)
MathUpgradeable.log10(uint256,MathUpgradeable.Rounding) (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#194-196)

```


- 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
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/staking/deposits/DepositsAdapter.sol#11-14)
    - StakingEpochManager.currentEpoch() (src/staking/epochs/StakingEpochManager.sol#11-14)
DepositsAdapter._previousEpochBalance(DepositsAdapter.DeferredDeposit).currentEpoch (src/staking/deposits/DepositsAdapter.sol#15-18)
    - StakingEpochManager.currentEpoch() (src/staking/epochs/StakingEpochManager.sol#15-18)
Finalizer._finalizePool(bytes32,uint16,uint16).currentEpoch (src/staking/rewards/finalizer.sol#1-4)
    - StakingEpochManager.currentEpoch() (src/staking/epochs/StakingEpochManager.sol#1-4)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-use-after-initialization

StakingEpochManager._endEpoch() (src/staking/epochs/StakingEpochManager.sol#69-76) :
    Dangerous comparisons:
        - require(bool,string)(block.timestamp >= getEpochEarliestEndTime(),E301) (src/staking/epochs/StakingEpochManager.sol#69-76)
Finalizer.unclaimedRewards(address) (src/staking/rewards/finalizer.sol#75-81) uses dangerous comparison:
    Dangerous comparisons:
        - rewards != 0 (src/staking/rewards/finalizer.sol#78)
Finalizer._finalizePool(bytes32,uint16,uint16) (src/staking/rewards/finalizer.sol#1-4) :
    Dangerous comparisons:
        - require(bool,string)({_depositedRewardsFund >= (poolReward + baseReward),E301}) (src/staking/rewards/finalizer.sol#1-4)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-time-comparisons

AddressUpgradeable._revert(bytes,string) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#1-1)
    - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#1-1)
StringsUpgradeable.toString(uint256) (lib/openzeppelin-contracts/contracts-upgradeable/utils/StringUpgradeable.sol#1-1)
    - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/StringUpgradeable.sol#1-1)
    - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/StringUpgradeable.sol#1-1)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#1-1)
    - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/MathUpgradeable.sol#1-1)
    - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/MathUpgradeable.sol#1-1)
    - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/MathUpgradeable.sol#1-1)
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-use-after-initialization

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/AccessControl.sol#1-1)
    - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/access/IAccessControl.sol#1-1)
    - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/security/Pausable.sol#1-1)
    - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/token/ERC20/IERC20.sol#1-1)
    - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/ContextUpgradeabilityAware.sol#1-1)
    - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/StringsUpgradeable.sol#1-1)
    - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/introspection/IERC165.sol#1-1)
    - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/introspection/Introspectable.sol#1-1)
    - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/MathUpgradeable.sol#1-1)
    - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/MathUpgradeable.sol#1-1)
    - ^0.8.1 (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#1-1)
    - ^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/Initializable.sol#1-1)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-solidity-versions

```

```
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
```

- DelegatorRewardsAdapter.sol

```

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-upg
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contract
- denominator = denominator / twos (lib/openzeppelin-contracts/contracts-upg
- inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contract
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contract
- denominator = denominator / twos (lib/openzeppelin-contracts/contracts-upg
- inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contract
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contract
- denominator = denominator / twos (lib/openzeppelin-contracts/contracts-upg
- inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contract
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contract
- denominator = denominator / twos (lib/openzeppelin-contracts/contracts-upg
- inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contract
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contract
- denominator = denominator / twos (lib/openzeppelin-contracts/contracts-upg
- inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contract
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contract
- denominator = denominator / twos (lib/openzeppelin-contracts/contracts-upg
- inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contract
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contract
- prod0 = prod0 / twos (lib/openzeppelin-contracts/contracts-upgradeable/ut
- result = prod0 * inverse (lib/openzeppelin-contracts/contracts-upgradeable
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: 

```

```

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

StakingEpochManager._endEpoch() (src/staking/epochs/StakingEpochManager.sol#69-76)
  Dangerous comparisons:
    - require(bool,string)(block.timestamp >= getEpochEarliestEndTime(),E301) (function)
      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#106-115)
  Dangerous comparisons:
    - require(bool,string)({_depositedRewardsFund >= (poolReward + baseReward),E301}) (function)
      Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-time

AddressUpgradeable._revert(bytes,string) (lib/openzeppelin-contracts/contracts-upgradeable/AddressUpgradeable.sol#106-115)
  - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#106-115)
    StringsUpgradeable.toString(uint256) (lib/openzeppelin-contracts/contracts-upgradeable/utils/StringsUpgradeable.sol#106-115)
    - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/StringUpgradeability.sol#106-115)
    - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/StringUpgradeability.sol#106-115)
    MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#106-115)
    - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/MathUpgradeability.sol#106-115)
    - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/MathUpgradeability.sol#106-115)
    - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/MathUpgradeability.sol#106-115)
    StakingStorage.filterPoolIdsIdle(bytes32[])
      - INLINE ASM (src/staking/StakingStorage.sol#194-196)
      Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-upgradeability

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/AccessControl.sol#106-115)
  - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/access/IAccessControl.sol#106-115)
  - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/security/Pausable.sol#106-115)
  - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/token/ERC20/IERC20.sol#106-115)
  - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/ContextUpgradeability.sol#106-115)
  - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/StringsUpgradeability.sol#106-115)
  - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/introspect/IntrospectibleUpgradeability.sol#106-115)
  - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/introspect/IntrospectibleUpgradeability.sol#106-115)
  - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/MathUpgradeability.sol#106-115)
  - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/MathUpgradeability.sol#106-115)
  - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/MathUpgradeability.sol#106-115)
  - ^0.8.1 (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeability.sol#106-115)
  - ^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/InitUpgradeability.sol#106-115)
  Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-solidity-versions
```

```
AccessControlUpgradeable.__AccessControl_init() (lib/openzeppelin-contracts/contracts-upgradeable/access-control/AccessControl.sol#10-11)
AccessControlUpgradeable.__AccessControl_init_unchained() (lib/openzeppelin-contracts/contracts-upgradeable/access-control/AccessControl.sol#10-11)
AccessControlUpgradeable._setRoleAdmin(bytes32,bytes32) (lib/openzeppelin-contracts/contracts-upgradeable/access-control/AccessControl.sol#10-11)
AccessControlUpgradeable._setupRole(bytes32,address) (lib/openzeppelin-contracts/contracts-upgradeable/access-control/AccessControl.sol#10-11)
AddressUpgradeable._revert(bytes,string) (lib/openzeppelin-contracts/contracts-upgradeable/address/Address.sol#10-11)
AddressUpgradeable.functionCall(address,bytes) (lib/openzeppelin-contracts/contracts-upgradeable/address/Address.sol#10-11)
AddressUpgradeable.functionCall(address,bytes,string) (lib/openzeppelin-contracts/contracts-upgradeable/address/Address.sol#10-11)
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/address/Address.sol#10-11)
AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (lib/openzeppelin-contracts/contracts-upgradeable/address/Address.sol#10-11)
AddressUpgradeable.functionStaticCall(address,bytes) (lib/openzeppelin-contracts/contracts-upgradeable/address/Address.sol#10-11)
AddressUpgradeable.functionStaticCall(address,bytes,string) (lib/openzeppelin-contracts/contracts-upgradeable/address/Address.sol#10-11)
AddressUpgradeable.isContract(address) (lib/openzeppelin-contracts/contracts-upgradeable/address/Address.sol#10-11)
AddressUpgradeable.sendValue(address,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/address/Address.sol#10-11)
AddressUpgradeable.verifyCallResult(bool,bytes,string) (lib/openzeppelin-contracts/contracts-upgradeable/address/Address.sol#10-11)
AddressUpgradeable.verifyCallResultFromTarget(address,bool,bytes,string) (lib/openzeppelin-contracts/contracts-upgradeable/address/Address.sol#10-11)
Configurable._setConfig(bytes32,uint256) (src/staking/config/Configurable.sol#26-30)
ContextUpgradeable.__Context_init() (lib/openzeppelin-contracts/contracts-upgradeable/context/Context.sol#10-11)
ContextUpgradeable.__Context_init_unchained() (lib/openzeppelin-contracts/contracts-upgradeable/context/Context.sol#10-11)
ContextUpgradeable._msgData() (lib/openzeppelin-contracts/contracts-upgradeable/utility/Context.sol#10-11)
DelegatorRewardsAdapter._collectDelegatorRewards(address,bytes32) (src/staking/rewards/DelegatorRewardsAdapter.sol#10-11)
DelegatorRewardsAdapter._collectDelegatorRewardsMany(address,bytes32[]) (src/staking/rewards/DelegatorRewardsAdapter.sol#10-11)
DepositsAdapter._decreaseNextBalance(DepositsAdapter.DeferredDeposit,uint256) (src/staking/deposits/DepositsAdapter.sol#10-11)
DepositsAdapter._increaseNextBalance(DepositsAdapter.DeferredDeposit,uint256) (src/staking/deposits/DepositsAdapter.sol#10-11)
DepositsAdapter._storeDeposit(DepositsAdapter.DeferredDeposit,DepositsAdapter.Defer) (src/staking/deposits/DepositsAdapter.sol#10-11)
ERC165Upgradeable.__ERC165_init() (lib/openzeppelin-contracts/contracts-upgradeable/introspection/ERC165Upgradeable.sol#10-11)
ERC165Upgradeable.__ERC165_init_unchained() (lib/openzeppelin-contracts/contracts-upgradeable/introspection/ERC165Upgradeable.sol#10-11)
Finalizer._markPoolAsCreated(bytes32) (src/staking/rewards/Finalizer.sol#86-89)
Finalizer._markPoolAsTerminated(bytes32) (src/staking/rewards/Finalizer.sol#96-101)
Initializable._disableInitializers() (lib/openzeppelin-contracts/contracts-upgradeable/initi...
Initializable._getInitializedVersion() (lib/openzeppelin-contracts/contracts-upgradeable/initi...
Initializable._isInitializing() (lib/openzeppelin-contracts/contracts-upgradeable/initi...
MathUpgradeable.average(uint256,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#10-11)
MathUpgradeable.ceilDiv(uint256,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#10-11)
MathUpgradeable.log10(uint256) (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#10-11)
MathUpgradeable.log10(uint256,MathUpgradeable.Rounding) (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#10-11)
MathUpgradeable.log2(uint256) (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#10-11)
MathUpgradeable.log2(uint256,MathUpgradeable.Rounding) (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#10-11)
MathUpgradeable.log256(uint256) (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#10-11)
MathUpgradeable.log256(uint256,MathUpgradeable.Rounding) (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#10-11)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#10-11)
MathUpgradeable.mulDiv(uint256,uint256,uint256,MathUpgradeable.Rounding) (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#10-11)
MathUpgradeable.sqrt(uint256) (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#10-11)
MathUpgradeable.sqrt(uint256,MathUpgradeable.Rounding) (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#10-11)
PausableUpgradeable.__Pausable_init() (lib/openzeppelin-contracts/contracts-upgradeable/pausability/PausableUpgradeable.sol#10-11)
PausableUpgradeable.__Pausable_init_unchained() (lib/openzeppelin-contracts/contracts-upgradeable/pausability/PausableUpgradeable.sol#10-11)
PausableUpgradeable._pause() (lib/openzeppelin-contracts/contracts-upgradeable/security/PausableUpgradeable.sol#10-11)
PausableUpgradeable._requirePaused() (lib/openzeppelin-contracts/contracts-upgradeable/security/PausableUpgradeable.sol#10-11)
```

- StakingEpochManager.sol

`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-upgradeable/utils/AddressUpgradeable.sol:10)
    - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol:10)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol:10)
    - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol:10)
    - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol:10)
    - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol:10)
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/MathUtil.sol](#))
- ^0.8.1 ([lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeability.sol](#))
- ^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/Initializable.sol](#))

```

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/openze
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(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-contr
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 mixed mode
Function StakingEpochManager.__StakingEpochManager_init_unchained(uint256) (src/staking/epochs/StakingEpochManager.sol#78)
Variable StakingEpochManager.__gap (src/staking/epochs/StakingEpochManager.sol#78)
Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-detector>

Variable StakingConstants.OWNER_STAKE_INFLUENCE_DEN (src/staking/config/StakingConstants.sol#8)
Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#variable-references>

StakingConstants.OWNER_ROLE (src/staking/config/StakingConstants.sol#8) is never used
StakingConstants.PAUSER_ROLE (src/staking/config/StakingConstants.sol#11) is never used
StakingConstants.FINALIZER_ROLE (src/staking/config/StakingConstants.sol#14) is never used
StakingConstants.JAILER_ROLE (src/staking/config/StakingConstants.sol#17) is never used
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) is never used
StakingConstants.POOL_SATURATION_CAP (src/staking/config/StakingConstants.sol#35) is never used
StakingConstants.OWNER_STAKE_INFLUENCE_NUM (src/staking/config/StakingConstants.sol#36)
StakingConstants.OWNER_STAKE_INFLUENCE_DEN (src/staking/config/StakingConstants.sol#37)
StakingConstants.WITHDRAWAL_LOCK_TIME (src/staking/config/StakingConstants.sol#42)
StakingConstants.POOL_COMMISSION_FEE_COOLDOWN (src/staking/config/StakingConstants.sol#43)
StakingConstants.POOL_UNJAILING_FEE (src/staking/config/StakingConstants.sol#48) is never used
StakingConstants.PERCENTAGE_SCALING_FACTOR (src/staking/config/StakingConstants.sol#50)
StakingConstants.MAX_OWNER_BASE_APY (src/staking/config/StakingConstants.sol#56) is never used
StakingConstants.MIN_POOL_COMMISSION_FEE (src/staking/config/StakingConstants.sol#57)
StakingConstants.TOKEN_FRACTION_SCALING_FACTOR (src/staking/config/StakingConstants.sol#58)
StakingConstants.ONE_YEAR_SECS (src/staking/config/StakingConstants.sol#68) is never used
StakingEpochManager.__gap (src/staking/epochs/StakingEpochManager.sol#78) is never used
Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables>

- DelegationsAdapter.sol


```
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#1)
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#1)
        - inverse = (3 * denominator) ^ 2 (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#1)
      MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#1)
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#1)
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#1)
      MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#1)
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#1)
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#1)
      MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#1)
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#1)
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#1)
      MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#1)
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#1)
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#1)
      MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#1)
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#1)
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#1)
      MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#1)
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#1)
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#1)
      MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#1)
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#1)
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#1)
      MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#1)
        - prod0 = prod0 / twos (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#1)
        - result = prod0 * inverse (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#1)
      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/staking/rewards/Finalizer.sol#1)
      Finalizer._finalizePool(bytes32,uint16,uint16) (src/staking/rewards/Finalizer.sol#1)
        - poolReward = (poolReward * performance) / PERCENTAGE_SCALING_FACTOR (src/staking/rewards/Finalizer.sol#1)
        - commissionReward = (poolReward * poolPtr.meta.commissionFee) / PERCENTAGE_SCALING_FACTOR (src/staking/rewards/Finalizer.sol#1)
      Finalizer.computePoolEpochRewards(uint256,uint256) (src/staking/rewards/Finalizer.sol#1)
        - earningStake = (ownerStake * (totalStake - earningStake)) / saturationCap (src/staking/rewards/Finalizer.sol#1)
        - earningStake = (earningStake * pledgeFactorNum) / pledgeFactorDen (src/staking/rewards/Finalizer.sol#1)
      Finalizer.computePoolEpochRewards(uint256,uint256) (src/staking/rewards/Finalizer.sol#1)
        - earningStake = (earningStake * pledgeFactorNum) / pledgeFactorDen (src/staking/rewards/Finalizer.sol#1)
        - earningStake = (earningStake * pledgeFactorDen) / (pledgeFactorDen + pledgedStake) (src/staking/rewards/Finalizer.sol#1)
      Finalizer.computePoolEpochRewards(uint256,uint256) (src/staking/rewards/Finalizer.sol#1)
        - (((earningStake * getConfig(POOL_EARNING_STAKE_APY)) / PERCENTAGE_SCALING_FACTOR) * performance) / PERCENTAGE_SCALING_FACTOR (src/staking/rewards/Finalizer.sol#1)
      Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiplication
```

DelegatorRewardsAdapter._collectDelegatorRewardsMany(address,bytes32[]).it (src/staking/rewards/DelegatorRewardsAdapter.sol#1)
 DelegationsAdapter._updateDelegationCurrencies(bytes32,address,StakingStorage.Currency[]).it (src/staking/rewards/DelegationsAdapter.sol#1)
 Finalizer.finalizePools(bytes32[],uint16[],uint16[]).it (src/staking/rewards/Finalizer.sol#1)
 DelegatorRewardsAdapter.computeDelegatorReward(address,bytes32[]).it (src/staking/rewards/DelegatorRewardsAdapter.sol#1)
 DelegationsAdapter.validateCurrencies(StakingStorage.Currency[]).it (src/staking/rewards/DelegationsAdapter.sol#1)
 Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-variable>

```

DepositsAdapter._loadDeposit(DepositsAdapter.DeferredDeposit).currentEpoch (src/staking/epochs/StakingEpochManager.sol#106-115)
    - StakingEpochManager.currentEpoch() (src/staking/epochs/StakingEpochManager.sol#106-115)
DepositsAdapter._previousEpochBalance(DepositsAdapter.DeferredDeposit).currentEpoch (src/staking/epochs/StakingEpochManager.sol#106-115)
    - StakingEpochManager.currentEpoch() (src/staking/epochs/StakingEpochManager.sol#106-115)
DelegatorRewardsAdapter._computeDelegatorRewardOverInterval(bytes32,uint256,uint64,bytes32) (src/staking/rewards/Finalizer.sol#75-81)
    - Finalizer.endEpoch() (src/staking/rewards/Finalizer.sol#75-81)
DelegatorRewardsAdapter._computeDelegatorReward(address,bytes32).currentEpoch (src/staking/rewards/Finalizer.sol#75-81)
    - StakingEpochManager.currentEpoch() (src/staking/epochs/StakingEpochManager.sol#106-115)
Finalizer._finalizePool(bytes32,uint16,uint16).currentEpoch (src/staking/rewards/Finalizer.sol#75-81)
    - StakingEpochManager.currentEpoch() (src/staking/epochs/StakingEpochManager.sol#106-115)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-use

StakingEpochManager._endEpoch() (src/staking/epochs/StakingEpochManager.sol#69-76)
    Dangerous comparisons:
        - require(bool,string)(block.timestamp >= getEpochEarliestEndTime(),E301) (src/staking/epochs/StakingEpochManager.sol#69-76)
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#75-81)
    Dangerous comparisons:
        - require(bool,string)({_depositedRewardsFund >= (poolReward + baseReward),E301}) (src/staking/rewards/Finalizer.sol#75-81)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-time-use

AddressUpgradeable._revert(bytes,string) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#194-196)
    - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#194-196)
StringsUpgradeable.toString(uint256) (lib/openzeppelin-contracts/contracts-upgradeable/utils/StringUpgradeable.sol#194-196)
    - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/StringUpgradeable.sol#194-196)
    - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/StringUpgradeable.sol#194-196)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#194-196)
    - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#194-196)
    - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#194-196)
    - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#194-196)
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-use

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/AccessControl.sol#194-196)
    - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/access/IAccessControl.sol#194-196)
    - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/security/Pausable.sol#194-196)
    - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/token/ERC20/IERC20.sol#194-196)
    - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/ContextUpgradeability.sol#194-196)
    - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/StringsUpgradeability.sol#194-196)
    - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/introspect/Introspectible.sol#194-196)
    - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/introspect/IntrospectionUpgradeability.sol#194-196)
    - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/MathUpgradeability.sol#194-196)
    - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/MathUpgradeability.sol#194-196)
    - ^0.8.1 (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeability.sol#194-196)
    - ^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/InitReference: <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/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/contractAccessControlUpgradeable.__AccessControl_init_unchained() (lib/openzeppelin-contractAccessControlUpgradeable._setRoleAdmin(bytes32,bytes32) (lib/openzeppelin-contractsAccessControlUpgradeable._setupRole(bytes32,address) (lib/openzeppelin-contracts/coAddressUpgradeable._revert(bytes,string) (lib/openzeppelin-contracts/contracts-upgrAddressUpgradeable.functionCall(address,bytes) (lib/openzeppelin-contracts/contractAddressUpgradeable.functionCall(address,bytes,string) (lib/openzeppelin-contracts/cAddressUpgradeable.functionCallWithValue(address,bytes,uint256) (lib/openzeppelin-cAddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (lib/openzeAddressUpgradeable.functionStaticCall(address,bytes) (lib/openzeppelin-contracts/coAddressUpgradeable.functionStaticCall(address,bytes,string) (lib/openzeppelin-contAddressUpgradeable.isContract(address) (lib/openzeppelin-contracts/contracts-upgradAddressUpgradeable.sendValue(address,uint256) (lib/openzeppelin-contracts/contractsAddressUpgradeable.verifyCallResult(bool,bytes,string) (lib/openzeppelin-contracts/AddressUpgradeable.verifyCallResultFromTarget(address,bool,bytes,string) (lib/openzConfigurable._setConfig(bytes32,uint256) (src/staking/config/Configurable.sol#26-30ContextUpgradeable.__Context_init() (lib/openzeppelin-contracts/contracts-upgradeabContextUpgradeable.__Context_init_unchained() (lib/openzeppelin-contracts/contractsContextUpgradeable._msgData() (lib/openzeppelin-contracts/contracts-upgradeable/utiDelegationsAdapter._initializePoolOwner(bytes32,address,StakingStorage.Currency[]])DelegationsAdapter._stake(bytes32,address,StakingStorage.Currency[]]) (src/staking/dDelegationsAdapter._unstake(bytes32,address,StakingStorage.Currency[]]) (src/stakingDelegationsAdapter._updateDeferredDeposit(DepositsAdapter.DeferredDeposit,uint256,bDelegationsAdapter._updateDelegation(bytes32,address,StakingStorage.Currency[],boolDelegationsAdapter._updateDelegationCurrencies(bytes32,address,StakingStorage.CurreDelegationsAdapter.validateCurrencies(StakingStorage.Currency[]]) (src/staking/deposDelegatorRewardsAdapter._collectDelegatorRewards(address,bytes32) (src/staking/rewaDelegatorRewardsAdapter._collectDelegatorRewardsMany(address,bytes32[]]) (src/stakinDepositsAdapter._decreaseNextBalance(DepositsAdapter.DeferredDeposit,uint256) (src/DepositsAdapter._increaseNextBalance(DepositsAdapter.DeferredDeposit,uint256) (src/DepositsAdapter._storeDeposit(DepositsAdapter.DeferredDeposit,DepositsAdapter.DeferERC165Upgradeable.__ERC165_init() (lib/openzeppelin-contracts/contracts-upgradeableERC165Upgradeable.__ERC165_init_unchained() (lib/openzeppelin-contracts/contracts-uFinalizer._markPoolAsCreated(bytes32) (src/staking/rewards/Finalizer.sol#86-89) isFinalizer._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/contracts-upgradeable/math/MathUpgradeable.sol#117)
    - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#117)
    - inverse = (3 * denominator) ^ 2 (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#117)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#117)
    - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#117)
    - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#117)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#117)
    - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#117)
    - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#117)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#117)
    - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#117)
    - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#117)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#117)
    - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#117)
    - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#117)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#117)
    - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#117)
    - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#117)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#117)
    - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#117)
    - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#117)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#117)
    - denominator = denominator / twos (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#117)
    - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#117)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-by-zero
```

```
DepositsAdapter._loadDeposit(DepositsAdapter.DeferredDeposit).currentEpoch (src/staking/epochs/StakingEpochManager.sol#104)
    - StakingEpochManager.currentEpoch() (src/staking/epochs/StakingEpochManager.sol#104)
DepositsAdapter._previousEpochBalance(DepositsAdapter.DeferredDeposit).currentEpoch (src/staking/epochs/StakingEpochManager.sol#104)
    - StakingEpochManager.currentEpoch() (src/staking/epochs/StakingEpochManager.sol#104)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-use-before-initialization
```

```
StakingEpochManager._endEpoch() (src/staking/epochs/StakingEpochManager.sol#69-76)
    Dangerous comparisons:
        - require(bool,string)(block.timestamp >= getEpochEarliestEndTime(),E301) (src/staking/epochs/StakingEpochManager.sol#70)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-time-comparisons
```

```
AddressUpgradeable._revert(bytes,string) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#117)
    - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#117)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#117)
    - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#117)
    - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#117)
    - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#117)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-use-after-scope
```

```
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/deposit  
DepositsAdapter._previousEpochBalance(DepositsAdapter.DeferredDeposit) (src/stakin  
DepositsAdapter._storeDeposit(DepositsAdapter.DeferredDeposit,DepositsAdapter.Defer  
Initializable._disableInitializers() (lib/openzeppelin-contracts/contracts-upgradeade  
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-upgradeade  
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contrac  
MathUpgradeable.mulDiv(uint256,uint256,uint256,MathUpgradeable.Rounding) (lib/openz
```


- TransfersAdapter.sol

```
TransfersAdapter._strictTransferFrom(address,IERC20Upgradeable,uint256) (src/stakin
Reference: 

```
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:

```
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: 

```
TransfersAdapter._strictTransferFrom\\\(address,IERC20Upgradeable,uint256\\\) \\\(src/stakin
 - require\\\(bool,string\\\)\\\(erc20.balanceOf\\\(address\\\(this\\\)\\\) == previousBalance +
Reference:

```
TransfersAdapter.\\\\_strictTransferFrom\\\\(address,StakingStorage.Currency\\\\[\\\\]\\\\).it \\\\(src/sta
Reference: 68
```


```


```


```


```

```

DepositsAdapter._loadDeposit(DepositsAdapter.DeferredDeposit).currentEpoch (src/staking/epochs/StakingEpochManager.sol#111-117)
    - StakingEpochManager.currentEpoch() (src/staking/epochs/StakingEpochManager.sol#111-117)
DepositsAdapter._previousEpochBalance(DepositsAdapter.DeferredDeposit).currentEpoch (src/staking/epochs/StakingEpochManager.sol#118-124)
    - StakingEpochManager.currentEpoch() (src/staking/epochs/StakingEpochManager.sol#118-124)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable

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

AddressUpgradeable._revert(bytes,string) (lib/openzeppelin-contracts/contracts-upgradeable/AddressUpgradeable.sol#100-106)
    - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#100-106)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/math/MathUpgradeable.sol#100-106)
    - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/MathUpgradeable.sol#100-106)
    - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/MathUpgradeable.sol#100-106)
    - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/MathUpgradeable.sol#100-106)
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-usages

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.sol#10-14)
    - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/token/ERC20/IERC20.sol#10-14)
    - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/token/ERC20/extensions/IERC20Permit.sol#10-14)
    - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/token/ERC20/extensions/IERC20Permit.sol#10-14)
    - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/token/ERC20/utils/IERC20.sol#10-14)
    - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/MathUpgradeable.sol#10-14)
    - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/MathUpgradeable.sol#10-14)
    - ^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/MathUpgradeable.sol#10-14)
    - ^0.8.1 (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#10-14)
    - ^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/Initializable.sol#10-14)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-practices

AddressUpgradeable._revert(bytes,string) (lib/openzeppelin-contracts/contracts-upgradeable/AddressUpgradeable.sol#100-106)
AddressUpgradeable.functionCall(address,bytes) (lib/openzeppelin-contracts/contracts-upgradeable/AddressUpgradeable.sol#107-113)
AddressUpgradeable.functionCall(address,bytes,string) (lib/openzeppelin-contracts/contracts-upgradeable/AddressUpgradeable.sol#114-120)
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/AddressUpgradeable.sol#121-127)
AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (lib/openzeppelin-contracts/contracts-upgradeable/AddressUpgradeable.sol#128-134)
AddressUpgradeable.functionStaticCall(address,bytes) (lib/openzeppelin-contracts/contracts-upgradeable/AddressUpgradeable.sol#135-141)
AddressUpgradeable.functionStaticCall(address,bytes,string) (lib/openzeppelin-contracts/contracts-upgradeable/AddressUpgradeable.sol#142-148)
AddressUpgradeable.isContract(address) (lib/openzeppelin-contracts/contracts-upgradeable/AddressUpgradeable.sol#149-155)
AddressUpgradeable.sendValue(address,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/AddressUpgradeable.sol#156-162)

```

```
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 versions
solc-0.8.11 is not recommended for deployment
Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-pragma-version>
 - Variable StakingConstants.OWNER_STAKE_INFLUENCE_DEN (src/staking/config/StakingConstants.sol#2) is not in mixed case
Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#variable-naming>
- 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-pragma-version>
 - Variable Configurable.__gap (src/staking/config/Configurable.sol#36) is not in mixed case
Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#conformance>
 - Configurable.__gap (src/staking/config/Configurable.sol#36) is never used in Configurable
Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable>

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