// HALBORN

BWarelabs - Staking

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **February 7th, 2022 - February 25th, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	02/07/2022	Roberto Reigada
0.2	Document Updates	02/25/2022	Roberto Reigada
0.3	Draft Review	02/25/2022	Gabi Urrutia
1.0	Remediation Plan	03/01/2022	Roberto Reigada
1.1	Remediation Plan Review	03/01/2022	Gabi Urrutia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

BWarelabs engaged Halborn to conduct a security audit on their staking smart contracts beginning on February 7th, 2022 and ending on February 21st, 2022. The security assessment was scoped to the smart contracts provided in the GitHub repository bwarelabs/bware-staking.

1.2 AUDIT SUMMARY

The team at Halborn was provided three 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 within the smart contracts

In summary, Halborn identified some security risks that were addressed by BWarelabs team.

1.3 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 bridge 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
- Graphing out functionality and contract logic/connectivity/functions (solgraph)
- Manual assessment of use and safety for the critical Solidity variables and functions in scope to identify any arithmetic related vulnerability classes
- Manual testing by custom scripts
- Scanning of solidity files for vulnerabilities, security hotspots or bugs. (MythX)
- Static Analysis of security for scoped contract, and imported functions. (Slither)
- Testnet deployment (Brownie, Remix IDE)

RISK METHODOLOGY:

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

RISK SCALE - LIKELIHOOD

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

RISK SCALE - IMPACT

5 - May cause devastating and unrecoverable impact or loss.
4 - May cause a significant level of impact or loss.

3 - May cause a partial impact or loss to many.
2 - May cause temporary impact or loss.
1 - May cause minimal or un-noticeable impact.

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

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL

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

1.4 SCOPE

IN-SCOPE:

The security assessment was scoped to the following smart contracts:

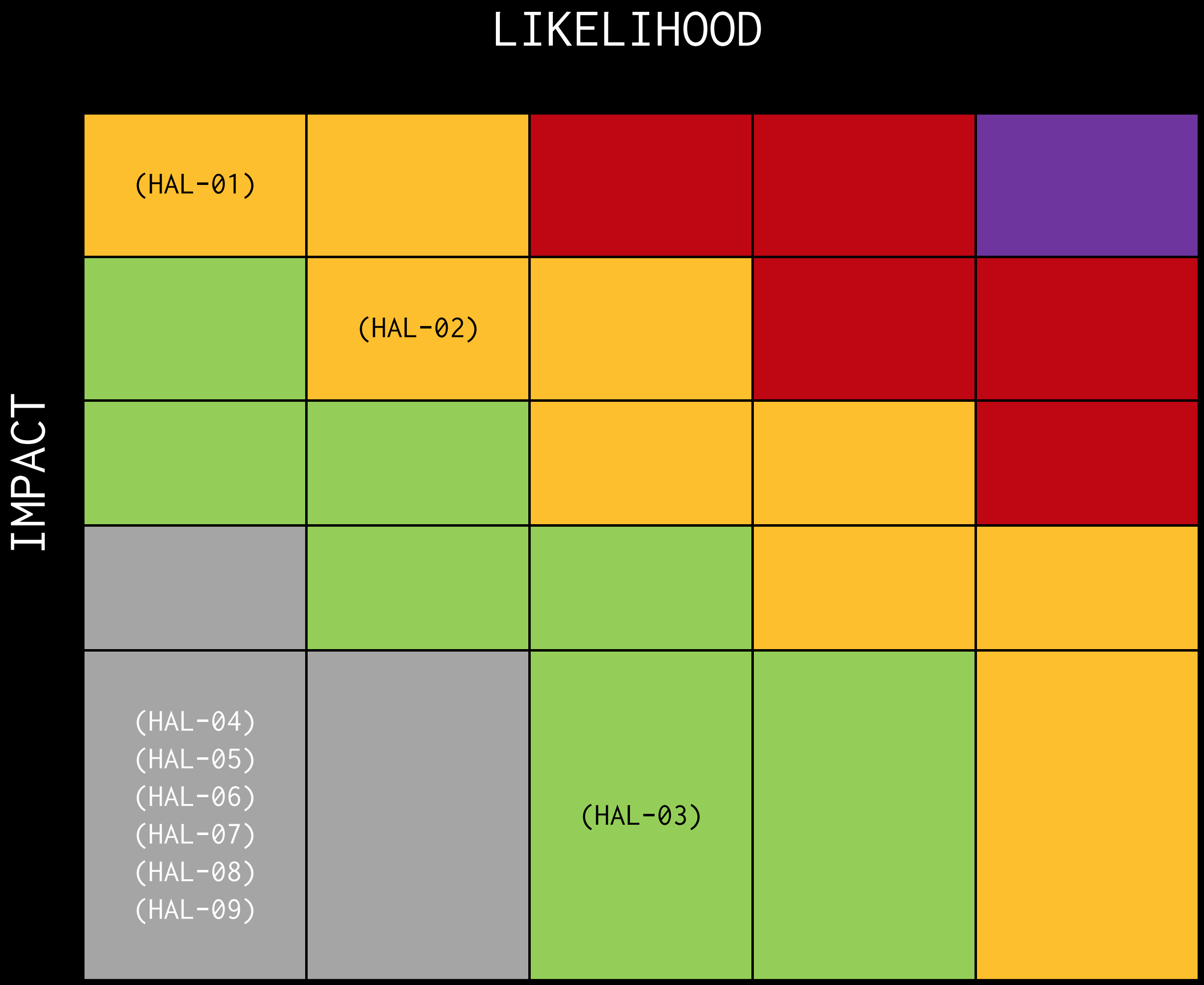
- Staking.sol
- StakingData.sol
- StakingUtility.sol

Commit ID: 8c84ab4f673f3805a961a41ac3f3369ee70eb393

Fixed ID: 151cb13628d9718e76d5cebdbcffbf5de844b197

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	2	1	6



SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL01 - FUNCTION SETUNLOCKEPOCH HAS NO MAXIMUM PERIOD LIMIT	Medium	SOLVED - 03/01/2022
HAL02 - FLOATING PRAGMA	Medium	RISK ACCEPTED
HAL03 - FUNCTION GETFORKEDDELEGATIONSCOUNT DISPLAYS A WRONG COUNTER AFTER TRYJOINFRAGMENT CALL	Low	RISK ACCEPTED
HAL04 - INCONSISTENT BALANCE WHEN SUPPLYING TRANSFER-ON-FEE OR DEFLATIONARY TOKENS	Informational	SOLVED - 03/01/2022
HAL05 - INCOMPATIBILITY WITH TOKENS THAT DO NOT HAVE 18 DECIMALS	Informational	SOLVED - 03/01/2022
HAL06 - UNNEEDED INITIALIZATION OF UINT256 VARIABLES TO 0	Informational	SOLVED - 03/01/2022
HAL07 - USING ++I CONSUMES LESS GAS THAN I++ IN LOOPS	Informational	SOLVED - 03/01/2022
HAL08 - STATE VARIABLE MISSING IMMUTABLE MODIFIER	Informational	SOLVED - 03/01/2022
HAL09 - FUNCTION ISCOMPLETE CAN BE REMOVED OR DECLARED EXTERNAL	Informational	SOLVED - 03/01/2022

FINDINGS & TECH DETAILS

3.1 (HAL-01) FUNCTION SETUNLOCKEPOCH HAS NO MAXIMUM PERIOD LIMIT - MEDIUM

Description:

In the contract Staking, the function setUnlockEpoch() defines the amount of time that a user will have to wait before he can receive back his previously staked tokens:

Listing 1: Staking.sol

```
    function setUnlockEpoch(uint256 period) external onlyRole(
        OWNER_ROLE) {
        _unlockEpoch = period;
    }
```

Listing 2: Staking.sol (Line 551)

```
540     function _lockTokens(address admin, uint256 amount) private {
541         Admin storage adminObj = _adminRegistry[admin];
542         uint256 epoch = _unlockEpoch;
543         if (adminObj.lockedTokens > 0) {
544             epoch = MathUtils.weightedAverage(
545                 MathUtils.diffOrZero(adminObj.unlockTime, block.
                        timestamp),
546                 adminObj.lockedTokens,
547                 epoch,
548                 amount
549             );
550         }
551         adminObj.unlockTime = block.timestamp.add(epoch);
552         adminObj.lockedTokens = adminObj.lockedTokens.add(amount);
553     }
```

As this function has no maximum period limit, a malicious owner could call this function with a very high period value causing that the users would never be able to retrieve back his staked tokens.

Risk Level:

Likelihood - 1
Impact - 5

Recommendation:

It is recommended to add a require statement that sets a maximum of time that the tokens can be locked, for example 1 year.

Remediation Plan:

SOLVED: The BWarelabs team added the suggested require statement. Tokens cannot be locked now for more than a year:

Listing 3: Staking.sol (Line 124)

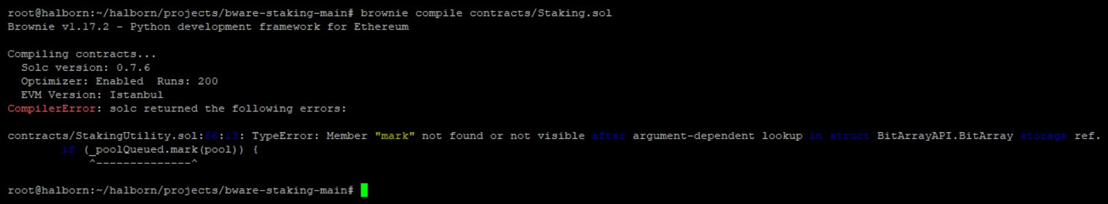
```
123 function setUnlockEpoch(uint256 period) external onlyRole(
        OWNER_ROLE) {
124     require(period <= 365 days, "Value too large for the unlocking
            period");
125     _unlockEpoch = period;
126 }
```

3.2 (HAL-02) FLOATING PRAGMA - MEDIUM

Description:

Contracts should be deployed with the same compiler version and flags used during development and testing. Locking the pragma helps to ensure that contracts do not accidentally get deployed using another pragma. For example, an outdated pragma version might introduce bugs that affect the contract system negatively or recently released pragma versions may have unknown security vulnerabilities.

In this case, the contracts do not compile with the ^0.7.0 version:



```
root@halborn:~/halborn/projects/bware-staking-main# brownie compile contracts/Staking.sol
Brownie v1.17.2 - Python development framework for Ethereum

Compiling contracts...
  Solc version: 0.7.6
  Optimizer: Enabled  Runs: 200
  EVM Version: Istanbul
CompilerError: solc returned the following errors:

contracts/StakingUtility.sol:66:13: TypeError: Member "mark" not found or not visible after argument-dependent lookup in struct BitArrayAPI.BitArray storage ref.
        if (_poolQueued.mark(pool)) {
            ^--------------^

root@halborn:~/halborn/projects/bware-staking-main#
```

Code Location:

Listing 4: Staking.sol

```
pragma solidity >=0.6.12 <0.8.0;
```

Listing 5: StakingData.sol

```
pragma solidity >=0.6.12 <0.8.0;
```

Listing 6: StakingUtility.sol

```
pragma solidity =0.6.12 <0.8.0;
```

Risk Level:

Likelihood - 2
Impact - 4

Recommendation:

Consider locking the pragma in all the contracts to the 0.6.12 version. It is not recommended to use a floating pragma in production.

Remediation Plan:

RISK ACCEPTED: The BWarelabs team accepts this risk as they plan to upgrade all the contracts in the future to version ^0.8.0.

3.3 (HAL-03) FUNCTION GETFORKEDDELEGATIONSCOUNT DISPLAYS A WRONG COUNTER AFTER TRYJOINFRAGMENT CALL - LOW

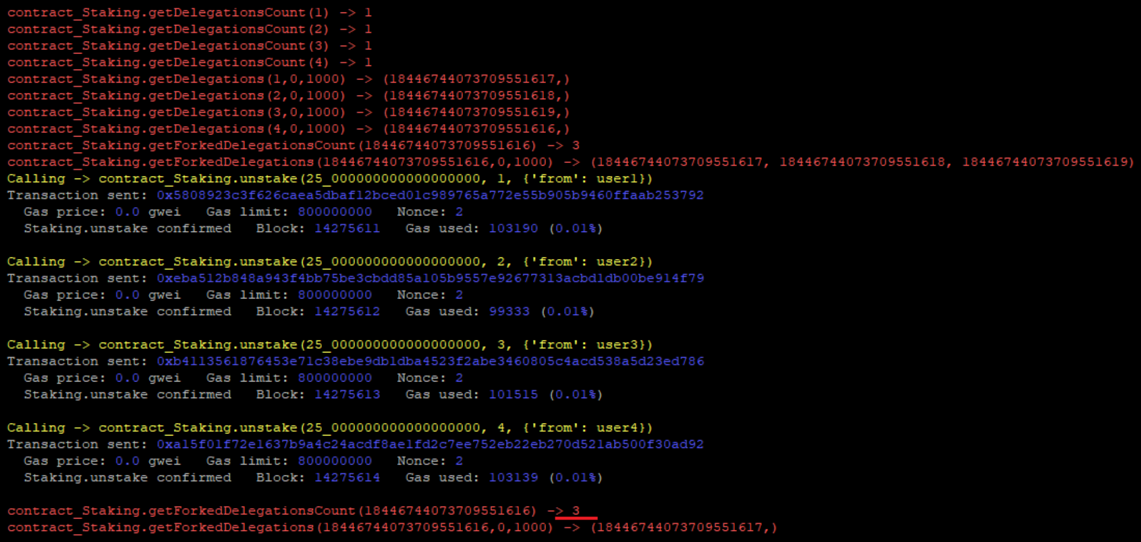
Description:

In the contract Staking, the function getForkedDelegationsCount() is used to display the amount of forked/sons for a delegation id:

Listing 7: Staking.sol

```
    function getForkedDelegationsCount(uint128 id) external view
        returns (uint256) {
        return _nonceForkedId[id];
    }
```

This value is displayed correctly initially, but once _tryJoinFragment() function is called, the value is not displayed correctly anymore:



```
contract_Staking.getDelegationsCount(1) -> 1
contract_Staking.getDelegationsCount(2) -> 1
contract_Staking.getDelegationsCount(3) -> 1
contract_Staking.getDelegationsCount(4) -> 1
contract_Staking.getDelegations(1,0,1000) -> (18446744073709551617,)
contract_Staking.getDelegations(2,0,1000) -> (18446744073709551618,)
contract_Staking.getDelegations(3,0,1000) -> (18446744073709551619,)
contract_Staking.getDelegations(4,0,1000) -> (18446744073709551616,)
contract_Staking.getForkedDelegationsCount(18446744073709551616) -> 3
contract_Staking.getForkedDelegations(18446744073709551616,0,1000) -> (18446744073709551617, 18446744073709551618, 18446744073709551619)
Calling -> contract_Staking.unstake(25_000000000000000000, 1, {'from': user1})
Transaction sent: 0x5808923c3f626caea5dbaf12bced01c989765a772e55b905b9460ffaab253792
  Gas price: 0.0 gwei   Gas limit: 800000000   Nonce: 2
  Staking.unstake confirmed   Block: 14275611   Gas used: 103190 (0.01%)

Calling -> contract_Staking.unstake(25_000000000000000000, 2, {'from': user2})
Transaction sent: 0xeba512b848a943f4bb75be3cbdd85a105b9557e92677313acbd1db00be914f79
  Gas price: 0.0 gwei   Gas limit: 800000000   Nonce: 2
  Staking.unstake confirmed   Block: 14275612   Gas used: 99333 (0.01%)

Calling -> contract_Staking.unstake(25_000000000000000000, 3, {'from': user3})
Transaction sent: 0xb4113561876453e71c38ebe9db1dba4523f2abe3460805c4acd538a5d23ed786
  Gas price: 0.0 gwei   Gas limit: 800000000   Nonce: 2
  Staking.unstake confirmed   Block: 14275613   Gas used: 101515 (0.01%)

Calling -> contract_Staking.unstake(25_000000000000000000, 4, {'from': user4})
Transaction sent: 0xa15f01f72e1637b9a4c24acdf8ae1fd2c7ee752eb22eb270d521ab500f30ad92
  Gas price: 0.0 gwei   Gas limit: 800000000   Nonce: 2
  Staking.unstake confirmed   Block: 14275614   Gas used: 103139 (0.01%)

contract_Staking.getForkedDelegationsCount(18446744073709551616) -> 3
contract_Staking.getForkedDelegations(18446744073709551616,0,1000) -> (18446744073709551617,)
```

This happens because _nonceForkedId[id] is not decreased when the _tryJoinFragment() function is called.

Risk Level:

Likelihood - 3
Impact - 1

Recommendation:

It is recommended to fix the _tryJoinFragment() function logic so the getForkedDelegationsCount() view function always displays the appropriate amount of forked/sons for a delegation id.

Remediation Plan:

RISK ACCEPTED: The BWarelabs team accepts this risk.

3.4 (HAL-04) INCONSISTENT BALANCE WHEN SUPPLYING TRANSFER-ON-FEE OR DEFLATIONARY TOKENS - INFORMATIONAL

Description:

In the contract Staking, the stake(), increaseReserve(), createNewPool() and _createDelegation() functions assume that the amount of _currencyToken is transferred to the smart contract after calling _currencyToken.safeTransferFrom(_msgSender(), address(this), amount) (and thus it updates the states variables accordingly). For example:

Listing 8: Staking.sol - stake (Lines 245,248)

```
239 function stake(uint256 amount, uint128 pool) external override {
240     _onlyAdmin(pool, TYPE_POOL);
241     require(_acceptingTokens(pool), "Stake: can stake only to live
            unjailed pools");
242     require(getCapacity(pool) >= amount, "Stake: not enough
          capacity to top up stake");
243 
244     _updateRewardsPool(pool);
245     _currencyToken.safeTransferFrom(_msgSender(), address(this),
          amount);
246 
247     Pool storage poolObj = _poolRegistry[pool];
248     poolObj.staked = poolObj.staked.add(amount);
249 
250     emit Stake(pool, amount);
251     mergePendingAssets(1);
252 }
```

However, this may not be true if the _currencyToken is a transfer-on-fee token or a deflationary/rebasing token, causing the received amount to be less than the accounted amount in the different state variables.

The Risk Level was set to informational as BWarelabs team will use

BWARE tokens in the smart contract which are not transfer-on-fee or deflationary.

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Get the actual received amount by calculating the difference of token balance before and after the transfer. For example:

Listing 9: Staking.sol - stake fixed (Lines 7,9,12,14)

```
 1 function stake(uint256 amount, uint128 pool) external override {
 2     _onlyAdmin(pool, TYPE_POOL);
 3     require(_acceptingTokens(pool), "Stake: can stake only to live unjailed pools");
 4     require(getCapacity(pool) >= amount, "Stake: not enough capacity to top up stake");
 5 
 6     _updateRewardsPool(pool);
 7     uint256 balanceBefore = _currencyToken.balanceOf(address(this));
 8     _currencyToken.safeTransferFrom(_msgSender(), address(this), amount);
 9     uint256 receivedAmount = _currencyToken.balanceOf(address(this)) - balanceBefore;
10 
11     Pool storage poolObj = _poolRegistry[pool];
12     poolObj.staked = poolObj.staked.add(receivedAmount);
13 
14     emit Stake(pool, receivedAmount);
15     mergePendingAssets(1);
16 }
```

Remediation Plan:

SOLVED: The BWarelabs team added _strictTransferFrom() which checks the balance before and after the token transfer:

Listing 10: Staking.sol

```
function _strictTransferFrom(uint256 amount) private returns (
    uint256) {
    uint256 balanceBefore = _currencyToken.balanceOf(address(this)
        );
    _currencyToken.safeTransferFrom(_msgSender(), address(this),
        amount);
    return _currencyToken.balanceOf(address(this)) - balanceBefore
        ;
}
```

3.5 (HAL-05) INCOMPATIBILITY WITH TOKENS THAT DO NOT HAVE 18 DECIMALS - INFORMATIONAL

Description:

In the contract Staking the function setMinDeposit() assumes that the _currencyToken has 18 decimals. In the case that, other token like USDC (which has 6 decimals) was being used as the _currencyToken it would not be possible to use this function as if it was called it would set the minimum deposit to 1,000,000,000,000,000 USDC.

Code Location:

Listing 11: Staking.sol (Line 118)

```
117 function setMinDeposit(uint256 amount, bool typeDeposit) public
        onlyRole(OWNER_ROLE) {
118     require(amount >= 10 ** 18, "Required deposit should be of non
            -zero tokens");
119     _requiredDeposit[typeDeposit] = amount;
120 }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to check the _currencyToken decimals as shown below:

Listing 12: Staking.sol (Line 118)

```
117 function setMinDeposit(uint256 amount, bool typeDeposit) public
        onlyRole(OWNER_ROLE) {
118     decimals = _currencyToken.decimals();
119     require(amount >= 10 ** decimals, "Required deposit should be
            of non-zero tokens");
120     _requiredDeposit[typeDeposit] = amount;
121 }
```

Remediation Plan:

SOLVED: The BWarelabs team now checks the token decimals in the setMinDeposit() function:

Listing 13: Staking.sol (Line 119)

```
118 function setMinDeposit(uint256 amount, bool typeDeposit) public
        onlyRole(OWNER_ROLE) {
119     require(amount >= 10 ** uint256(_currencyToken.decimals()), "
            Required deposit should be of non-zero tokens");
120     _requiredDeposit[typeDeposit] = amount;
121 }
```

3.6 (HAL-06) UNNEEDED INITIALIZATION OF UINT256 VARIABLES TO 0 - INFORMATIONAL

Description:

As `i` is an `uint256`, it is already initialized to 0. `uint256 i = 0` reassigns the 0 to `i` which wastes gas.

Code Location:

Staking.sol
- Line 303: `for (uint256 i = 0; i < workload; i++){`
- Line 438: `for (uint256 i = 0; i < ids.length; i++){`
- Line 456: `for (uint256 i = 0; i < ids.length; i++){`
- Line 473: `for (uint256 i = 0; i < ids.length; i++){`
- Line 630: `for (uint256 i = 0; i < ids.length; i++){`
- Line 638: `for (uint256 i = 0; i < ids.length; i++){`
- Line 697: `for (uint256 i = 0; i < ids.length; i++){`
- Line 718: `for (uint256 i = 0; i < ids.length; i++){`

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to not initialize uint256 variables to 0 to save some gas. For example, use instead:
`for (uint256 i; i < workload; ++i){`.

Remediation Plan:

SOLVED: The BWarelabs team removed the initialization to zero for all the mentioned iterator variables.

3.7 (HAL-07) USING ++I CONSUMES LESS GAS THAN I++ IN LOOPS - INFORMATIONAL

Description:

In the loop below, the variable i is incremented using i++. It is known that, in loops, using ++i costs less gas per iteration than i++.

Code Location:

Staking.sol
- Line 156: `for (uint256 index = start; index < start.add(end); index ++){`
- Line 159: `assets[count++] = id;`
- Line 186: `for (uint128 index = uint128(start.add(1)); index <= start.add(end); index++){`
- Line 188: `assets[count++] = id + index;`
- Line 303: `for (uint256 i = 0; i < workload; i++){`
- Line 317: `for (; capacity > 0 && i < workload; i++){`
- Line 400: `for (; workload > 0; workload--){`
- Line 438: `for (uint256 i = 0; i < ids.length; i++){`
- Line 456: `for (uint256 i = 0; i < ids.length; i++){`
- Line 473: `for (uint256 i = 0; i < ids.length; i++){`
- Line 630: `for (uint256 i = 0; i < ids.length; i++){`
- Line 638: `for (uint256 i = 0; i < ids.length; i++){`
- Line 697: `for (uint256 i = 0; i < ids.length; i++){`
- Line 701: `ids[start++] = ids[i];`
- Line 703: `end++;`
- Line 718: `for (uint256 i = 0; i < ids.length; i++){`

Proof of Concept:

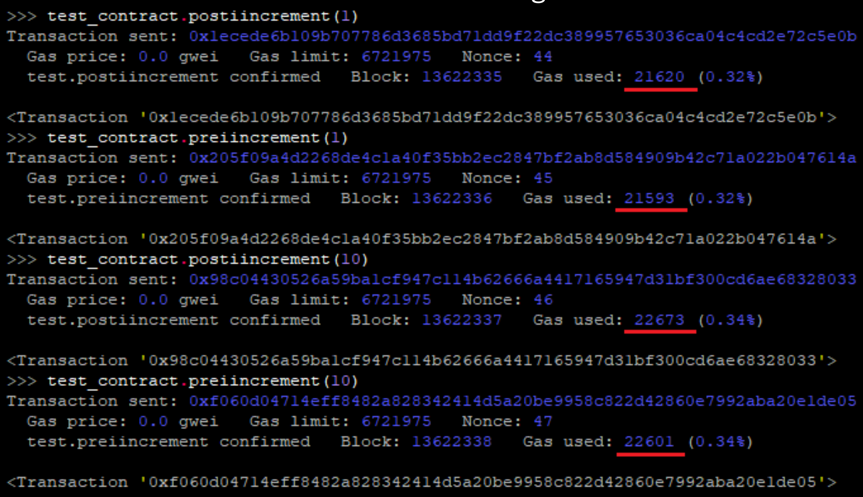
For example, based in the following test contract:

Listing 14: Test.sol

```solidity
//SPDX-License-Identifier: MIT
pragma solidity 0.8.9;

contract test {
    function postiincrement(uint256 iterations) public {
        for (uint256 i = 0; i < iterations; i++) {
        }
    }
    function preiincrement(uint256 iterations) public {
        for (uint256 i = 0; i < iterations; ++i) {
        }
    }
}
```

We can see the difference in the gas costs:



```
>>> test_contract.postiincrement(1)
Transaction sent: 0x1ecede6b109b707786d3685bd71dd9f22dc389957653036ca04c4cd2e72c5e0b
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 44
  test.postiincrement confirmed   Block: 13622335   Gas used: 21620 (0.32%)

<Transaction '0x1ecede6b109b707786d3685bd71dd9f22dc389957653036ca04c4cd2e72c5e0b'>
>>> test_contract.preiincrement(1)
Transaction sent: 0x205f09a4d2268de4c1a40f35bb2ec2847bf2ab8d584909b42c71a022b047614a
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 45
  test.preiincrement confirmed   Block: 13622336   Gas used: 21593 (0.32%)

<Transaction '0x205f09a4d2268de4c1a40f35bb2ec2847bf2ab8d584909b42c71a022b047614a'>
>>> test_contract.postiincrement(10)
Transaction sent: 0x98c04430526a59ba1cf947c114b62666a4417165947d31bf300cd6ae68328033
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 46
  test.postiincrement confirmed   Block: 13622337   Gas used: 22673 (0.34%)

<Transaction '0x98c04430526a59ba1cf947c114b62666a4417165947d31bf300cd6ae68328033'>
>>> test_contract.preiincrement(10)
Transaction sent: 0xf060d04714eff8482a828342414d5a20be9958c822d42860e7992aba20e1de05
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 47
  test.preiincrement confirmed   Block: 13622338   Gas used: 22601 (0.34%)

<Transaction '0xf060d04714eff8482a828342414d5a20be9958c822d42860e7992aba20e1de05'>
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to use ++i instead of i++ to increment the value of an uint variable inside a loop. This does not only apply to the iterator variable. It also applies to increments done inside the loop code block.

Remediation Plan:

SOLVED: The BWarelabs team now uses ++i instead of i++ inside loops to save some gas.

3.8 (HAL-08) STATE VARIABLE MISSING IMMUTABLE MODIFIER - INFORMATIONAL

Description:

In the contract StakingData, the state variable _currencyToken can be declared as immutable to reduce the gas costs.

The immutable keyword was added to Solidity in 0.6.5. State variables can be marked immutable which causes them to be read-only, but only assignable in the constructor.

Code Location:

Listing 15: StakingData.sol

```
IERC20 public _currencyToken;
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to add the immutable modifier to the _currencyToken state variable.

Remediation Plan:

SOLVED: The BWarelabs team added the immutable modifier to the _currencyToken state variable.

3.9 (HAL-09) FUNCTION ISCOMPLETE CAN BE REMOVED OR DECLARED EXTERNAL - INFORMATIONAL

Description:

In the contract Staking, the function isComplete() is marked as public but it is never directly called within the same contract. On the other hand, there is also an internal function with the same code: _isComplete().

Code Location:

Listing 16: Staking.sol

```
    function isComplete(uint128 pool) public view returns (bool) {
        return getCapacity(pool) == 0;
    }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to remove this function or to mark it as external to reduce the gas costs.

Remediation Plan:

SOLVED: The BWarelabs team removed the isComplete() public function.

AUTOMATED TESTING

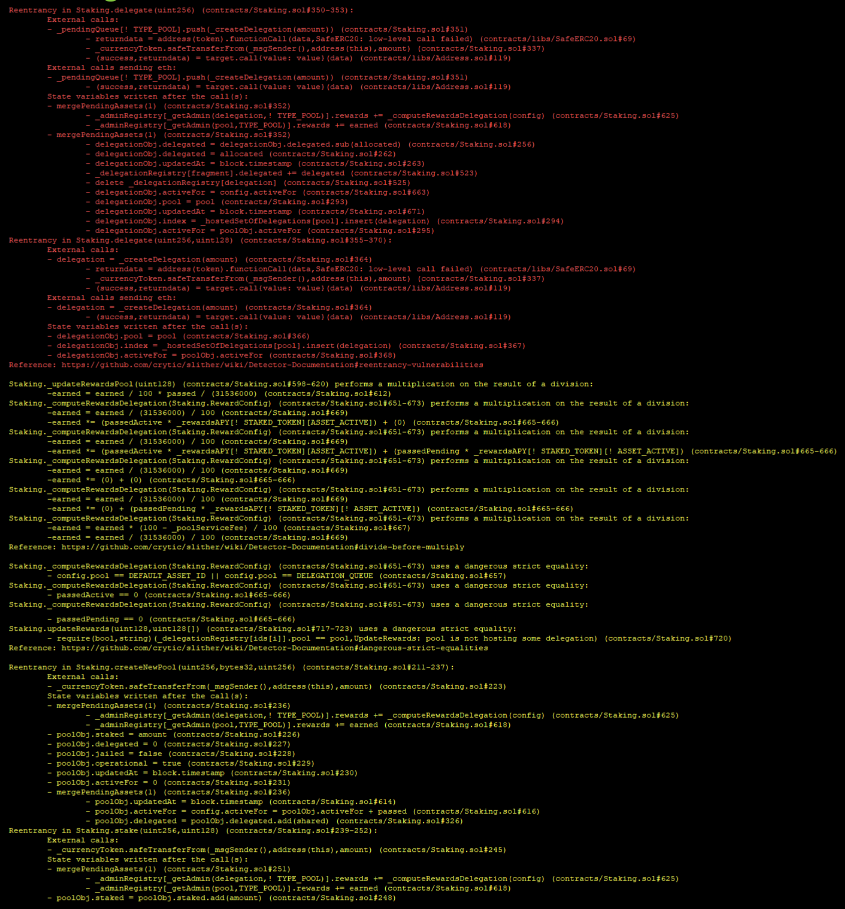
4.1 STATIC ANALYSIS REPORT

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' APIs across the entire code-base.

Slither results:

Staking.sol



```
Reentrancy in Staking.delegate(uint256) (contracts/Staking.sol#350-353):
        External calls:
        - _pendingQueue[! TYPE_POOL].push(_createDelegation(amount)) (contracts/Staking.sol#351)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (contracts/libs/SafeERC20.sol#69)
                - _currencyToken.safeTransferFrom(_msgSender(),address(this),amount) (contracts/Staking.sol#337)
                - (success,returndata) = target.call{value: value}(data) (contracts/libs/Address.sol#119)
        External calls sending eth:
        - _pendingQueue[! TYPE_POOL].push(_createDelegation(amount)) (contracts/Staking.sol#351)
                - (success,returndata) = target.call{value: value}(data) (contracts/libs/Address.sol#119)
        State variables written after the call(s):
        - mergePendingAssets(1) (contracts/Staking.sol#352)
                - _adminRegistry[_getAdmin(delegation,! TYPE_POOL)].rewards += _computeRewardsDelegation(config) (contracts/Staking.sol#625)
                - _adminRegistry[_getAdmin(pool,TYPE_POOL)].rewards += earned (contracts/Staking.sol#618)
        - mergePendingAssets(1) (contracts/Staking.sol#352)
                - delegationObj.delegated = delegationObj.delegated.sub(allocated) (contracts/Staking.sol#256)
                - delegationObj.delegated = allocated (contracts/Staking.sol#262)
                - delegationObj.updatedAt = block.timestamp (contracts/Staking.sol#263)
                - _delegationRegistry[fragment].delegated += delegated (contracts/Staking.sol#523)
                - delete _delegationRegistry[delegation] (contracts/Staking.sol#525)
                - delegationObj.activeFor = config.activeFor (contracts/Staking.sol#663)
                - delegationObj.pool = pool (contracts/Staking.sol#293)
                - delegationObj.updatedAt = block.timestamp (contracts/Staking.sol#671)
                - delegationObj.index = _hostedSetOfDelegations[pool].insert(delegation) (contracts/Staking.sol#294)
                - delegationObj.activeFor = poolObj.activeFor (contracts/Staking.sol#295)
Reentrancy in Staking.delegate(uint256,uint128) (contracts/Staking.sol#355-370):
        External calls:
        - delegation = _createDelegation(amount) (contracts/Staking.sol#364)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (contracts/libs/SafeERC20.sol#69)
                - _currencyToken.safeTransferFrom(_msgSender(),address(this),amount) (contracts/Staking.sol#337)
                - (success,returndata) = target.call{value: value}(data) (contracts/libs/Address.sol#119)
        External calls sending eth:
        - delegation = _createDelegation(amount) (contracts/Staking.sol#364)
                - (success,returndata) = target.call{value: value}(data) (contracts/libs/Address.sol#119)
        State variables written after the call(s):
        - delegationObj.pool = pool (contracts/Staking.sol#366)
        - delegationObj.index = _hostedSetOfDelegations[pool].insert(delegation) (contracts/Staking.sol#367)
        - delegationObj.activeFor = poolObj.activeFor (contracts/Staking.sol#368)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities

Staking._updateRewardsPool(uint128) (contracts/Staking.sol#598-620) performs a multiplication on the result of a division:
        -earned = earned / 100 * passed / (31536000) (contracts/Staking.sol#612)
Staking._computeRewardsDelegation(Staking.RewardConfig) (contracts/Staking.sol#651-673) performs a multiplication on the result of a division:
        -earned = earned / (31536000) / 100 (contracts/Staking.sol#669)
        -earned *= (passedActive * _rewardsAPY[! STAKED_TOKEN][ASSET_ACTIVE]) + (0) (contracts/Staking.sol#665-666)
Staking._computeRewardsDelegation(Staking.RewardConfig) (contracts/Staking.sol#651-673) performs a multiplication on the result of a division:
        -earned = earned / (31536000) / 100 (contracts/Staking.sol#669)
        -earned *= (passedActive * _rewardsAPY[! STAKED_TOKEN][ASSET_ACTIVE]) + (passedPending * _rewardsAPY[! STAKED_TOKEN][! ASSET_ACTIVE]) (contracts/Staking.sol#665-666)
Staking._computeRewardsDelegation(Staking.RewardConfig) (contracts/Staking.sol#651-673) performs a multiplication on the result of a division:
        -earned = earned / (31536000) / 100 (contracts/Staking.sol#669)
        -earned *= (0) + (0) (contracts/Staking.sol#665-666)
Staking._computeRewardsDelegation(Staking.RewardConfig) (contracts/Staking.sol#651-673) performs a multiplication on the result of a division:
        -earned = earned / (31536000) / 100 (contracts/Staking.sol#669)
        -earned *= (0) + (passedPending * _rewardsAPY[! STAKED_TOKEN][! ASSET_ACTIVE]) (contracts/Staking.sol#665-666)
Staking._computeRewardsDelegation(Staking.RewardConfig) (contracts/Staking.sol#651-673) performs a multiplication on the result of a division:
        -earned = earned * (100 - _poolServiceFee) / 100 (contracts/Staking.sol#667)
        -earned = earned / (31536000) / 100 (contracts/Staking.sol#669)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

Staking._computeRewardsDelegation(Staking.RewardConfig) (contracts/Staking.sol#651-673) uses a dangerous strict equality:
        - config.pool == DEFAULT_ASSET_ID || config.pool == DELEGATION_QUEUE (contracts/Staking.sol#657)
Staking._computeRewardsDelegation(Staking.RewardConfig) (contracts/Staking.sol#651-673) uses a dangerous strict equality:
        - passedActive == 0 (contracts/Staking.sol#665-666)
Staking._computeRewardsDelegation(Staking.RewardConfig) (contracts/Staking.sol#651-673) uses a dangerous strict equality:

        - passedPending == 0 (contracts/Staking.sol#665-666)
Staking.updateRewards(uint128,uint128[]) (contracts/Staking.sol#717-723) uses a dangerous strict equality:
        - require(bool,string)(_delegationRegistry[ids[i]].pool == pool,UpdateRewards: pool is not hosting some delegation) (contracts/Staking.sol#720)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in Staking.createNewPool(uint256,bytes32,uint256) (contracts/Staking.sol#211-237):
        External calls:
        - _currencyToken.safeTransferFrom(_msgSender(),address(this),amount) (contracts/Staking.sol#223)
        State variables written after the call(s):
        - mergePendingAssets(1) (contracts/Staking.sol#236)
                - _adminRegistry[_getAdmin(delegation,! TYPE_POOL)].rewards += _computeRewardsDelegation(config) (contracts/Staking.sol#625)
                - _adminRegistry[_getAdmin(pool,TYPE_POOL)].rewards += earned (contracts/Staking.sol#618)
        - poolObj.staked = amount (contracts/Staking.sol#226)
        - poolObj.delegated = 0 (contracts/Staking.sol#227)
        - poolObj.jailed = false (contracts/Staking.sol#228)
        - poolObj.operational = true (contracts/Staking.sol#229)
        - poolObj.updatedAt = block.timestamp (contracts/Staking.sol#230)
        - poolObj.activeFor = 0 (contracts/Staking.sol#231)
        - mergePendingAssets(1) (contracts/Staking.sol#236)
                - poolObj.updatedAt = block.timestamp (contracts/Staking.sol#614)
                - poolObj.activeFor = config.activeFor = poolObj.activeFor + passed (contracts/Staking.sol#616)
                - poolObj.delegated = poolObj.delegated.add(shared) (contracts/Staking.sol#326)
Reentrancy in Staking.stake(uint256,uint128) (contracts/Staking.sol#239-252):
        External calls:
        - _currencyToken.safeTransferFrom(_msgSender(),address(this),amount) (contracts/Staking.sol#245)
        State variables written after the call(s):
        - mergePendingAssets(1) (contracts/Staking.sol#251)
                - _adminRegistry[_getAdmin(delegation,! TYPE_POOL)].rewards += _computeRewardsDelegation(config) (contracts/Staking.sol#625)
                - _adminRegistry[_getAdmin(pool,TYPE_POOL)].rewards += earned (contracts/Staking.sol#618)
        - poolObj.staked = poolObj.staked.add(amount) (contracts/Staking.sol#248)
        - mergePendingAssets(1) (contracts/Staking.sol#251)
                - poolObj.updatedAt = block.timestamp (contracts/Staking.sol#614)
                - poolObj.activeFor = config.activeFor = poolObj.activeFor + passed (contracts/Staking.sol#616)
                - poolObj.delegated = poolObj.delegated.add(shared) (contracts/Staking.sol#326)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

Staking._updateRewardsPool(uint128).config (contracts/Staking.sol#599) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

Reentrancy in Staking._createDelegation(uint256) (contracts/Staking.sol#333-348):
        External calls:
        - _currencyToken.safeTransferFrom(_msgSender(),address(this),amount) (contracts/Staking.sol#337)
        State variables written after the call(s):
        - delegationObj.pool = DELEGATION_QUEUE (contracts/Staking.sol#340)
        - delegationObj.index = 0 (contracts/Staking.sol#341)
        - delegationObj.delegated = amount (contracts/Staking.sol#342)
        - delegationObj.updatedAt = block.timestamp (contracts/Staking.sol#343)
        - delegationObj.activeFor = 0 (contracts/Staking.sol#344)
Reentrancy in Staking.createNewPool(uint256,bytes32,uint256) (contracts/Staking.sol#211-237):
        External calls:
        - _currencyToken.safeTransferFrom(_msgSender(),address(this),amount) (contracts/Staking.sol#223)
        State variables written after the call(s):
        - mergePendingAssets(1) (contracts/Staking.sol#236)
                - delegationObj.delegated = delegationObj.delegated.sub(allocated) (contracts/Staking.sol#256)
                - delegationObj.delegated = allocated (contracts/Staking.sol#262)
                - delegationObj.updatedAt = block.timestamp (contracts/Staking.sol#263)
                - _delegationRegistry[fragment].delegated += delegated (contracts/Staking.sol#523)
                - delete _delegationRegistry[delegation] (contracts/Staking.sol#525)
                - delegationObj.activeFor = config.activeFor (contracts/Staking.sol#663)
                - delegationObj.pool = pool (contracts/Staking.sol#293)
                - delegationObj.updatedAt = block.timestamp (contracts/Staking.sol#671)
                - delegationObj.index = _hostedSetOfDelegations[pool].insert(delegation) (contracts/Staking.sol#294)
                - delegationObj.activeFor = poolObj.activeFor (contracts/Staking.sol#295)
        - mergePendingAssets(1) (contracts/Staking.sol#236)
                - delete _hostedForkDelegation[pool][parent] (contracts/Staking.sol#536)
                - _hostedForkDelegation[pool][parent] = delegation (contracts/Staking.sol#520)
        - mergePendingAssets(1) (contracts/Staking.sol#236)
                - id + (1 + _nonceForkedId[id] ++) (contracts/StakingUtility.sol#62)
Reentrancy in Staking.delegate(uint256) (contracts/Staking.sol#350-353):
        External calls:
        - _pendingQueue[! TYPE_POOL].push(_createDelegation(amount)) (contracts/Staking.sol#351)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (contracts/libs/SafeERC20.sol#69)
                - _currencyToken.safeTransferFrom(_msgSender(),address(this),amount) (contracts/Staking.sol#337)
                - (success,returndata) = target.call{value: value}(data) (contracts/libs/Address.sol#119)
        External calls sending eth:
        - _pendingQueue[! TYPE_POOL].push(_createDelegation(amount)) (contracts/Staking.sol#351)
                - (success,returndata) = target.call{value: value}(data) (contracts/libs/Address.sol#119)
        State variables written after the call(s):
        - mergePendingAssets(1) (contracts/Staking.sol#352)
                - delete _hostedForkDelegation[pool][parent] (contracts/Staking.sol#536)
                - _hostedForkDelegation[pool][parent] = delegation (contracts/Staking.sol#520)
        - mergePendingAssets(1) (contracts/Staking.sol#352)
                - id + (1 + _nonceForkedId[id] ++) (contracts/StakingUtility.sol#62)
        - mergePendingAssets(1) (contracts/Staking.sol#352)
                - poolObj.updatedAt = block.timestamp (contracts/Staking.sol#614)
                - poolObj.activeFor = config.activeFor = poolObj.activeFor + passed (contracts/Staking.sol#616)
                - poolObj.delegated = poolObj.delegated.add(shared) (contracts/Staking.sol#326)
Reentrancy in Staking.increaseReserve(uint256) (contracts/Staking.sol#130-133):
        External calls:
        - _currencyToken.safeTransferFrom(_msgSender(),address(this),amount) (contracts/Staking.sol#131)
        State variables written after the call(s):
        - _reserveRewards = _reserveRewards.add(amount) (contracts/Staking.sol#132)
Reentrancy in Staking.stake(uint256,uint128) (contracts/Staking.sol#239-252):
        External calls:
        - _currencyToken.safeTransferFrom(_msgSender(),address(this),amount) (contracts/Staking.sol#245)
        State variables written after the call(s):
        - mergePendingAssets(1) (contracts/Staking.sol#251)
                - delegationObj.delegated = delegationObj.delegated.sub(allocated) (contracts/Staking.sol#256)
                - delegationObj.delegated = allocated (contracts/Staking.sol#262)
                - delegationObj.updatedAt = block.timestamp (contracts/Staking.sol#263)
                - _delegationRegistry[fragment].delegated += delegated (contracts/Staking.sol#523)
                - delete _delegationRegistry[delegation] (contracts/Staking.sol#525)
                - delegationObj.activeFor = config.activeFor (contracts/Staking.sol#663)
                - delegationObj.pool = pool (contracts/Staking.sol#293)
                - delegationObj.updatedAt = block.timestamp (contracts/Staking.sol#671)
                - delegationObj.index = _hostedSetOfDelegations[pool].insert(delegation) (contracts/Staking.sol#294)
                - delegationObj.activeFor = poolObj.activeFor (contracts/Staking.sol#295)
        - mergePendingAssets(1) (contracts/Staking.sol#251)
                - delete _hostedForkDelegation[pool][parent] (contracts/Staking.sol#536)
                - _hostedForkDelegation[pool][parent] = delegation (contracts/Staking.sol#520)
        - mergePendingAssets(1) (contracts/Staking.sol#251)
                - id + (1 + _nonceForkedId[id] ++) (contracts/StakingUtility.sol#62)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in Staking._createDelegation(uint256) (contracts/Staking.sol#333-348):
        External calls:
        - _currencyToken.safeTransferFrom(_msgSender(),address(this),amount) (contracts/Staking.sol#337)
        Event emitted after the call(s):
        - Delegate(_msgSender(),amount,delegation) (contracts/Staking.sol#346)
Reentrancy in Staking.createNewPool(uint256,bytes32,uint256) (contracts/Staking.sol#211-237):
        External calls:
        - _currencyToken.safeTransferFrom(_msgSender(),address(this),amount) (contracts/Staking.sol#223)
        Event emitted after the call(s):
        - CreateNewPool(_msgSender(),amount,pool,id) (contracts/Staking.sol#234)
        - ForkFragment(id,allocated) (contracts/Staking.sol#258)
                - mergePendingAssets(1) (contracts/Staking.sol#236)
        - JoinFragment(delegation,fragment) (contracts/Staking.sol#526)
                - mergePendingAssets(1) (contracts/Staking.sol#236)
        - ShareToPool(pool,delegation) (contracts/Staking.sol#296)
                - mergePendingAssets(1) (contracts/Staking.sol#236)
Reentrancy in Staking.delegate(uint256) (contracts/Staking.sol#350-353):
        External calls:
        - _pendingQueue[! TYPE_POOL].push(_createDelegation(amount)) (contracts/Staking.sol#351)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (contracts/libs/SafeERC20.sol#69)
                - _currencyToken.safeTransferFrom(_msgSender(),address(this),amount) (contracts/Staking.sol#337)
                - (success,returndata) = target.call{value: value}(data) (contracts/libs/Address.sol#119)
        External calls sending eth:
        - _pendingQueue[! TYPE_POOL].push(_createDelegation(amount)) (contracts/Staking.sol#351)
                - (success,returndata) = target.call{value: value}(data) (contracts/libs/Address.sol#119)
        Event emitted after the call(s):
        - ForkFragment(id,allocated) (contracts/Staking.sol#258)
                - mergePendingAssets(1) (contracts/Staking.sol#352)
        - JoinFragment(delegation,fragment) (contracts/Staking.sol#526)
                - mergePendingAssets(1) (contracts/Staking.sol#352)
        - ShareToPool(pool,delegation) (contracts/Staking.sol#296)
                - mergePendingAssets(1) (contracts/Staking.sol#352)
Reentrancy in Staking.delegate(uint256,uint128) (contracts/Staking.sol#355-370):
        External calls:
        - delegation = _createDelegation(amount) (contracts/Staking.sol#364)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (contracts/libs/SafeERC20.sol#69)
                - _currencyToken.safeTransferFrom(_msgSender(),address(this),amount) (contracts/Staking.sol#337)
                - (success,returndata) = target.call{value: value}(data) (contracts/libs/Address.sol#119)
        External calls sending eth:
        - delegation = _createDelegation(amount) (contracts/Staking.sol#364)
                - (success,returndata) = target.call{value: value}(data) (contracts/libs/Address.sol#119)
        Event emitted after the call(s):
        - ShareToPool(pool,delegation) (contracts/Staking.sol#369)
Reentrancy in Staking.stake(uint256,uint128) (contracts/Staking.sol#239-252):
        External calls:
        - _currencyToken.safeTransferFrom(_msgSender(),address(this),amount) (contracts/Staking.sol#245)
        Event emitted after the call(s):
        - ForkFragment(id,allocated) (contracts/Staking.sol#258)
                - mergePendingAssets(1) (contracts/Staking.sol#251)
        - JoinFragment(delegation,fragment) (contracts/Staking.sol#526)
                - mergePendingAssets(1) (contracts/Staking.sol#251)
        - ShareToPool(pool,delegation) (contracts/Staking.sol#296)
                - mergePendingAssets(1) (contracts/Staking.sol#251)
        - Stake(pool,amount) (contracts/Staking.sol#250)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Staking.getAdminAssets(address,bool,uint256,uint256) (contracts/Staking.sol#145-165) uses timestamp for comparisons
        Dangerous comparisons:
        - getTokens(id,typeOf,typeOf) > 0 (contracts/Staking.sol#158)
Staking.getForkedDelegations(uint128,uint256,uint256) (contracts/Staking.sol#179-194) uses timestamp for comparisons
        Dangerous comparisons:
        - getTokens(id + index,! TYPE_POOL,! STAKED_TOKEN) > 0 (contracts/Staking.sol#187)
Staking.createNewPool(uint256,bytes32,uint256) (contracts/Staking.sol#211-237) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)((nodeObj.pool == 0 || poolObj.staked == 0) && (nodeObj.operational || _reservingNodeOff),Stake: node is not operational or already reserved by pool) (contracts/Staking.sol#217-219)
Staking.withdrawTokens(uint256) (contracts/Staking.sol#555-565) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(adminObj.unlockTime <= block.timestamp,Withdraw: unlocking period has not expired yet) (contracts/Staking.sol#557)
Staking._updateRewardsDelegation(uint128[],bool) (contracts/Staking.sol#636-649) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp.diffOrZero(delegationObj.updatedAt) <= 86400 (contracts/Staking.sol#640)
Staking._computeRewardsDelegation(Staking.RewardConfig) (contracts/Staking.sol#651-673) uses timestamp for comparisons
        Dangerous comparisons:
        - earned == 0 (contracts/Staking.sol#654)
        - config.pool == DEFAULT_ASSET_ID || config.pool == DELEGATION_QUEUE (contracts/Staking.sol#657)
        - passedActive > 0 && ! config.deleted (contracts/Staking.sol#662)
        - passedActive == 0 (contracts/Staking.sol#665-666)
        - passedPending == 0 (contracts/Staking.sol#665-666)
        - passedPending == 0 (contracts/Staking.sol#665-666)
Staking.getDelegationsOutdated(uint128,uint256,uint256,uint256) (contracts/Staking.sol#682-709) uses timestamp for comparisons
        Dangerous comparisons:
        - before >= block.timestamp && pool != DELEGATION_QUEUE (contracts/Staking.sol#693)
Staking.updateRewards(uint128,uint128[]) (contracts/Staking.sol#717-723) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(_delegationRegistry[ids[i]].pool == pool,UpdateRewards: pool is not hosting some delegation) (contracts/Staking.sol#720)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Staking.getAdminAssets(address,bool,uint256,uint256) (contracts/Staking.sol#145-165) uses assembly
        - INLINE ASM (contracts/Staking.sol#163)
Staking.getForkedDelegations(uint128,uint256,uint256) (contracts/Staking.sol#179-194) uses assembly
        - INLINE ASM (contracts/Staking.sol#192)
```

```
Staking.getDelegationsOutdated(uint128,uint256,uint256,uint256) (contracts/Staking.sol#682-709) uses assembly
	- INLINE ASM (contracts/Staking.sol#707)
Address.isContract(address) (contracts/libs/Address.sol#26-35) uses assembly
	- INLINE ASM (contracts/libs/Address.sol#33)
Address._verifyCallResult(bool,bytes,string) (contracts/libs/Address.sol#171-188) uses assembly
	- INLINE ASM (contracts/libs/Address.sol#180-183)
SetAPI.getSet(SetAPI.Set,uint256,uint256) (contracts/libs/SetAPI.sol#56-73) uses assembly
	- INLINE ASM (contracts/libs/SetAPI.sol#71)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
	- Version used: ['0.6.12', '>=0.6.0<0.8.0', '>=0.6.12<0.8.0', '>=0.6.2<0.8.0']
	- >=0.6.12<0.8.0 (contracts/IStaking.sol#3)
	- >=0.6.12<0.8.0 (contracts/IStakingData.sol#3)
	- 0.6.12 (contracts/Staking.sol#3)
	- ABIEncoderV2 (contracts/StakingData.sol#2)
	- 0.6.12 (contracts/StakingData.sol#3)
	- 0.6.12 (contracts/StakingUtility.sol#3)
	- >=0.6.2<0.8.0 (contracts/libs/Address.sol#3)
	- >=0.6.12<0.8.0 (contracts/libs/BitArrayAPI.sol#3)
	- >=0.6.12<0.8.0 (contracts/libs/HeapAPI.sol#3)
	- >=0.6.12<0.8.0 (contracts/libs/MathUtils.sol#3)
	- >=0.6.0<0.8.0 (contracts/libs/SafeERC20.sol#3)
	- >=0.6.0<0.8.0 (contracts/libs/SafeMath.sol#3)
	- >=0.6.12<0.8.0 (contracts/libs/SetAPI.sol#3)
	- >=0.6.12<0.8.0 (contracts/utils/AccessControl.sol#3)
	- >=0.6.0<0.8.0 (contracts/utils/Context.sol#3)
	- >=0.6.0<0.8.0 (contracts/utils/IERC20.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Staking._tryJoinFragment(uint128,uint128,uint256) (contracts/Staking.sol#515-531) has costly operations inside a loop:
	- delete _delegationRegistry[delegation] (contracts/Staking.sol#525)
Staking.undelegate(uint128[]) (contracts/Staking.sol#465-484) has costly operations inside a loop:
	- delete _delegationRegistry[ids[i]] (contracts/Staking.sol#480)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop

Address.functionCall(address,bytes) (contracts/libs/Address.sol#79-81) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (contracts/libs/Address.sol#104-106) is never used and should be removed
Address.functionDelegateCall(address,bytes) (contracts/libs/Address.sol#153-155) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (contracts/libs/Address.sol#163-169) is never used and should be removed
Address.functionStaticCall(address,bytes) (contracts/libs/Address.sol#129-131) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (contracts/libs/Address.sol#139-145) is never used and should be removed
Address.sendValue(address,uint256) (contracts/libs/Address.sol#53-59) is never used and should be removed
Context._msgData() (contracts/utils/Context.sol#20-24) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (contracts/libs/SafeERC20.sol#37-46) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (contracts/libs/SafeERC20.sol#53-56) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (contracts/libs/SafeERC20.sol#48-51) is never used and should be removed
SafeMath.mod(uint256,uint256) (contracts/libs/SafeMath.sol#139-141) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (contracts/libs/SafeMath.sol#155-158) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version>=0.6.12<0.8.0 (contracts/IStaking.sol#3) is too complex
Pragma version>=0.6.12<0.8.0 (contracts/IStakingData.sol#3) is too complex
Pragma version>=0.6.2<0.8.0 (contracts/libs/Address.sol#3) is too complex
Pragma version>=0.6.12<0.8.0 (contracts/libs/BitArrayAPI.sol#3) is too complex
Pragma version>=0.6.12<0.8.0 (contracts/libs/HeapAPI.sol#3) is too complex
Pragma version>=0.6.12<0.8.0 (contracts/libs/MathUtils.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (contracts/libs/SafeERC20.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (contracts/libs/SafeMath.sol#3) is too complex
Pragma version>=0.6.12<0.8.0 (contracts/libs/SetAPI.sol#3) is too complex
Pragma version>=0.6.12<0.8.0 (contracts/utils/AccessControl.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (contracts/utils/Context.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (contracts/utils/IERC20.sol#3) is too complex
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (contracts/libs/Address.sol#53-59):
	- (success) = recipient.call{value: amount}() (contracts/libs/Address.sol#57)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (contracts/libs/Address.sol#114-121):
	- (success,returndata) = target.call{value: value}(data) (contracts/libs/Address.sol#119)
Low level call in Address.functionStaticCall(address,bytes,string) (contracts/libs/Address.sol#139-145):
	- (success,returndata) = target.staticcall(data) (contracts/libs/Address.sol#143)
Low level call in Address.functionDelegateCall(address,bytes,string) (contracts/libs/Address.sol#163-169):
	- (success,returndata) = target.delegatecall(data) (contracts/libs/Address.sol#167)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Variable StakingData._currencyToken (contracts/StakingData.sol#17) is not in mixedCase
Variable StakingData._reserveRewards (contracts/StakingData.sol#20) is not in mixedCase
Variable StakingData._poolCapacity (contracts/StakingData.sol#23) is not in mixedCase
Variable StakingData._unlockEpoch (contracts/StakingData.sol#26) is not in mixedCase
Variable StakingData._poolServiceFee (contracts/StakingData.sol#29) is not in mixedCase
Variable StakingData._requiredDeposit (contracts/StakingData.sol#32) is not in mixedCase
Variable StakingData._rewardsAPY (contracts/StakingData.sol#35) is not in mixedCase
Variable StakingData._reservingNodeOff (contracts/StakingData.sol#38) is not in mixedCase
Variable StakingData._noncePrimaryId (contracts/StakingData.sol#41) is not in mixedCase
Variable StakingData._nonceForkedId (contracts/StakingData.sol#44) is not in mixedCase
Variable StakingData._adminOfAsset (contracts/StakingData.sol#47) is not in mixedCase
Variable StakingData._adminRegistry (contracts/StakingData.sol#50) is not in mixedCase
Variable StakingData._poolRegistry (contracts/StakingData.sol#52) is not in mixedCase
Variable StakingData._delegationRegistry (contracts/StakingData.sol#54) is not in mixedCase
Variable StakingData._hostedSetOfDelegations (contracts/StakingData.sol#57) is not in mixedCase
Variable StakingData._hostedForkDelegation (contracts/StakingData.sol#60) is not in mixedCase
Variable StakingData._pendingQueue (contracts/StakingData.sol#63) is not in mixedCase
Variable StakingData._poolQueued (contracts/StakingData.sol#66) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (contracts/utils/Context.sol#21)" inContext (contracts/utils/Context.sol#15-26)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

Variable StakingData._currencyToken (contracts/StakingData.sol#17) is too similar to Staking.constructor(address,uint256,uint256,uint256).currencyToken_ (contracts/Staking.sol#44)
Variable StakingData._poolCapacity (contracts/StakingData.sol#23) is too similar to Staking.constructor(address,uint256,uint256,uint256).poolCapacity_ (contracts/Staking.sol#45)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

StakingUtility.ONLY_ADMIN_ACCESS (contracts/StakingUtility.sol#29) is never used in Staking (contracts/Staking.sol#11-736)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

isComplete(uint128) should be declared external:
	- Staking.isComplete(uint128) (contracts/Staking.sol#89-91)
grantRole(bytes32,address) should be declared external:
	- AccessControl.grantRole(bytes32,address) (contracts/utils/AccessControl.sol#135-139)
revokeRole(bytes32,address) should be declared external:
	- AccessControl.revokeRole(bytes32,address) (contracts/utils/AccessControl.sol#150-154)
renounceRole(bytes32,address) should be declared external:
	- AccessControl.renounceRole(bytes32,address) (contracts/utils/AccessControl.sol#170-174)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

StakingData.sol

```
SetAPI.getSet(SetAPI.Set,uint256,uint256) (contracts/libs/SetAPI.sol#56-73) uses assembly
	- INLINE ASM (contracts/libs/SetAPI.sol#71)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
	- Version used: ['0.6.12', '>=0.6.0<0.8.0', '>=0.6.12<0.8.0']
	- >=0.6.12<0.8.0 (contracts/IStakingData.sol#3)
	- ABIEncoderV2 (contracts/StakingData.sol#2)
	- 0.6.12 (contracts/StakingData.sol#3)
	- >=0.6.12<0.8.0 (contracts/libs/BitArrayAPI.sol#3)
	- >=0.6.12<0.8.0 (contracts/libs/HeapAPI.sol#3)
	- >=0.6.12<0.8.0 (contracts/libs/SetAPI.sol#3)
	- >=0.6.0<0.8.0 (contracts/utils/IERC20.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

BitArrayAPI.mark(BitArrayAPI.BitArray,uint256) (contracts/libs/BitArrayAPI.sol#13-20) is never used and should be removed
BitArrayAPI.unmark(BitArrayAPI.BitArray,uint256) (contracts/libs/BitArrayAPI.sol#22-28) is never used and should be removed
HeapAPI.getQueue(HeapAPI.Heap) (contracts/libs/HeapAPI.sol#63-65) is never used and should be removed
HeapAPI.getQueue(HeapAPI.Heap,uint256,uint256) (contracts/libs/HeapAPI.sol#67-76) is never used and should be removed
HeapAPI.pop(HeapAPI.Heap) (contracts/libs/HeapAPI.sol#29-57) is never used and should be removed
HeapAPI.push(HeapAPI.Heap,uint256) (contracts/libs/HeapAPI.sol#15-27) is never used and should be removed
HeapAPI.sizeOf(HeapAPI.Heap) (contracts/libs/HeapAPI.sol#59-61) is never used and should be removed
SetAPI.getSet(SetAPI.Set,uint256,uint256) (contracts/libs/SetAPI.sol#56-73) is never used and should be removed
SetAPI.insert(SetAPI.Set,uint128) (contracts/libs/SetAPI.sol#29-37) is never used and should be removed
SetAPI.remove(SetAPI.Set) (contracts/libs/SetAPI.sol#39-45) is never used and should be removed
SetAPI.remove(SetAPI.Set,uint128) (contracts/libs/SetAPI.sol#47-50) is never used and should be removed
SetAPI.setup(SetAPI.Set,uint256) (contracts/libs/SetAPI.sol#22-27) is never used and should be removed
SetAPI.sizeOf(SetAPI.Set) (contracts/libs/SetAPI.sol#52-54) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version>=0.6.12<0.8.0 (contracts/IStakingData.sol#3) is too complex
Pragma version>=0.6.12<0.8.0 (contracts/libs/BitArrayAPI.sol#3) is too complex
Pragma version>=0.6.12<0.8.0 (contracts/libs/HeapAPI.sol#3) is too complex
Pragma version>=0.6.12<0.8.0 (contracts/libs/SetAPI.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (contracts/utils/IERC20.sol#3) is too complex
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Variable StakingData._currencyToken (contracts/StakingData.sol#17) is not in mixedCase
Variable StakingData._reserveRewards (contracts/StakingData.sol#20) is not in mixedCase
Variable StakingData._poolCapacity (contracts/StakingData.sol#23) is not in mixedCase
Variable StakingData._unlockEpoch (contracts/StakingData.sol#26) is not in mixedCase
Variable StakingData._poolServiceFee (contracts/StakingData.sol#29) is not in mixedCase
Variable StakingData._requiredDeposit (contracts/StakingData.sol#32) is not in mixedCase
Variable StakingData._rewardsAPY (contracts/StakingData.sol#35) is not in mixedCase
Variable StakingData._reservingNodeOff (contracts/StakingData.sol#38) is not in mixedCase
Variable StakingData._noncePrimaryId (contracts/StakingData.sol#41) is not in mixedCase
Variable StakingData._nonceForkedId (contracts/StakingData.sol#44) is not in mixedCase
Variable StakingData._adminOfAsset (contracts/StakingData.sol#47) is not in mixedCase
Variable StakingData._adminRegistry (contracts/StakingData.sol#50) is not in mixedCase
Variable StakingData._poolRegistry (contracts/StakingData.sol#52) is not in mixedCase
Variable StakingData._delegationRegistry (contracts/StakingData.sol#54) is not in mixedCase
Variable StakingData._hostedSetOfDelegations (contracts/StakingData.sol#57) is not in mixedCase
Variable StakingData._hostedForkDelegation (contracts/StakingData.sol#60) is not in mixedCase
Variable StakingData._pendingQueue (contracts/StakingData.sol#63) is not in mixedCase
Variable StakingData._poolQueued (contracts/StakingData.sol#66) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

StakingData._poolCapacity (contracts/StakingData.sol#23) should be constant
StakingData._poolServiceFee (contracts/StakingData.sol#29) should be constant
StakingData._reserveRewards (contracts/StakingData.sol#20) should be constant
StakingData._reservingNodeOff (contracts/StakingData.sol#38) should be constant
StakingData._unlockEpoch (contracts/StakingData.sol#26) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
```

StakingUtility.sol

```
SetAPI.getSet(SetAPI.Set,uint256,uint256) (contracts/libs/SetAPI.sol#56-73) uses assembly
        - INLINE ASM (contracts/libs/SetAPI.sol#71)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
        - Version used: ['0.6.12', '>=0.6.0<0.8.0', '>=0.6.12<0.8.0']
        - >=0.6.12<0.8.0 (contracts/IStakingData.sol#3)
        - ABIEncoderV2 (contracts/StakingData.sol#2)
        - 0.6.12 (contracts/StakingData.sol#3)
        - 0.6.12 (contracts/StakingUtility.sol#3)
        - >=0.6.12<0.8.0 (contracts/libs/BitArrayAPI.sol#3)
        - >=0.6.12<0.8.0 (contracts/libs/HeapAPI.sol#3)
        - >=0.6.0<0.8.0 (contracts/libs/SafeMath.sol#3)
        - >=0.6.12<0.8.0 (contracts/libs/SetAPI.sol#3)
        - >=0.6.0<0.8.0 (contracts/utils/IERC20.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

BitArrayAPI.mark(BitArrayAPI.BitArray,uint256) (contracts/libs/BitArrayAPI.sol#13-20) is never used and should be removed
BitArrayAPI.unmark(BitArrayAPI.BitArray,uint256) (contracts/libs/BitArrayAPI.sol#22-28) is never used and should be removed
HeapAPI.getQueue(HeapAPI.Heap) (contracts/libs/HeapAPI.sol#63-65) is never used and should be removed
HeapAPI.getQueue(HeapAPI.Heap,uint256,uint256) (contracts/libs/HeapAPI.sol#67-76) is never used and should be removed
HeapAPI.pop(HeapAPI.Heap) (contracts/libs/HeapAPI.sol#29-57) is never used and should be removed
HeapAPI.push(HeapAPI.Heap,uint256) (contracts/libs/HeapAPI.sol#15-27) is never used and should be removed
HeapAPI.sizeOf(HeapAPI.Heap) (contracts/libs/HeapAPI.sol#59-61) is never used and should be removed
SafeMath.add(uint256,uint256) (contracts/libs/SafeMath.sol#29-34) is never used and should be removed
SafeMath.div(uint256,uint256) (contracts/libs/SafeMath.sol#103-105) is never used and should be removed
SafeMath.div(uint256,uint256,string) (contracts/libs/SafeMath.sol#119-125) is never used and should be removed
SafeMath.mod(uint256,uint256) (contracts/libs/SafeMath.sol#139-141) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (contracts/libs/SafeMath.sol#155-158) is never used and should be removed
SafeMath.mul(uint256,uint256) (contracts/libs/SafeMath.sol#77-89) is never used and should be removed
SafeMath.sub(uint256,uint256) (contracts/libs/SafeMath.sol#46-48) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (contracts/libs/SafeMath.sol#60-65) is never used and should be removed
SetAPI.getSet(SetAPI.Set,uint256,uint256) (contracts/libs/SetAPI.sol#56-73) is never used and should be removed
SetAPI.insert(SetAPI.Set,uint128) (contracts/libs/SetAPI.sol#29-37) is never used and should be removed
SetAPI.remove(SetAPI.Set) (contracts/libs/SetAPI.sol#39-45) is never used and should be removed
SetAPI.remove(SetAPI.Set,uint128) (contracts/libs/SetAPI.sol#47-50) is never used and should be removed
SetAPI.setup(SetAPI.Set,uint256) (contracts/libs/SetAPI.sol#22-27) is never used and should be removed
SetAPI.sizeOf(SetAPI.Set) (contracts/libs/SetAPI.sol#52-54) is never used and should be removed
StakingUtility._createAssetId(address,bool) (contracts/StakingUtility.sol#36-47) is never used and should be removed
StakingUtility._forkDelegationId(uint128) (contracts/StakingUtility.sol#60-63) is never used and should be removed
StakingUtility._getAdmin(uint128,bool) (contracts/StakingUtility.sol#53-58) is never used and should be removed
StakingUtility._getParentDelegation(uint128) (contracts/StakingUtility.sol#49-51) is never used and should be removed
StakingUtility._guardedPoolPop(uint128) (contracts/StakingUtility.sol#71-74) is never used and should be removed
StakingUtility._guardedPoolPush(uint128) (contracts/StakingUtility.sol#65-69) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version>=0.6.12<0.8.0 (contracts/IStakingData.sol#3) is too complex
Pragma version>=0.6.12<0.8.0 (contracts/libs/BitArrayAPI.sol#3) is too complex
Pragma version>=0.6.12<0.8.0 (contracts/libs/HeapAPI.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (contracts/libs/SafeMath.sol#3) is too complex
Pragma version>=0.6.12<0.8.0 (contracts/libs/SetAPI.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (contracts/utils/IERC20.sol#3) is too complex
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Variable StakingData._currencyToken (contracts/StakingData.sol#17) is not in mixedCase
Variable StakingData._reserveRewards (contracts/StakingData.sol#20) is not in mixedCase
Variable StakingData._poolCapacity (contracts/StakingData.sol#23) is not in mixedCase
Variable StakingData._unlockEpoch (contracts/StakingData.sol#26) is not in mixedCase
Variable StakingData._poolServiceFee (contracts/StakingData.sol#29) is not in mixedCase
Variable StakingData._requiredDeposit (contracts/StakingData.sol#32) is not in mixedCase
Variable StakingData._rewardsAPY (contracts/StakingData.sol#35) is not in mixedCase
Variable StakingData._reservingNodeOff (contracts/StakingData.sol#38) is not in mixedCase
Variable StakingData._noncePrimaryId (contracts/StakingData.sol#41) is not in mixedCase
Variable StakingData._nonceForkedId (contracts/StakingData.sol#44) is not in mixedCase
Variable StakingData._adminOfAsset (contracts/StakingData.sol#47) is not in mixedCase
Variable StakingData._adminRegistry (contracts/StakingData.sol#50) is not in mixedCase
Variable StakingData._poolRegistry (contracts/StakingData.sol#52) is not in mixedCase
Variable StakingData._delegationRegistry (contracts/StakingData.sol#54) is not in mixedCase
Variable StakingData._hostedSetOfDelegations (contracts/StakingData.sol#57) is not in mixedCase
Variable StakingData._hostedForkDelegation (contracts/StakingData.sol#60) is not in mixedCase
Variable StakingData._pendingQueue (contracts/StakingData.sol#63) is not in mixedCase
Variable StakingData._poolQueued (contracts/StakingData.sol#66) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

StakingData._reservingNodeOff (contracts/StakingData.sol#38) is never used in StakingUtility (contracts/StakingUtility.sol#8-75)
StakingData._hostedSetOfDelegations (contracts/StakingData.sol#57) is never used in StakingUtility (contracts/StakingUtility.sol#8-75)
StakingData._hostedForkDelegation (contracts/StakingData.sol#60) is never used in StakingUtility (contracts/StakingUtility.sol#8-75)
StakingUtility.DEFAULT_ASSET_ID (contracts/StakingUtility.sol#12) is never used in StakingUtility (contracts/StakingUtility.sol#8-75)
StakingUtility.DELEGATION_QUEUE (contracts/StakingUtility.sol#14) is never used in StakingUtility (contracts/StakingUtility.sol#8-75)
StakingUtility.REMOVED_HOSTED_DELEGATION (contracts/StakingUtility.sol#16) is never used in StakingUtility (contracts/StakingUtility.sol#8-75)
StakingUtility.STAKED_TOKEN (contracts/StakingUtility.sol#21) is never used in StakingUtility (contracts/StakingUtility.sol#8-75)
StakingUtility.ASSET_ACTIVE (contracts/StakingUtility.sol#23) is never used in StakingUtility (contracts/StakingUtility.sol#8-75)
StakingUtility.ONLY_ADMIN_ACCESS (contracts/StakingUtility.sol#29) is never used in StakingUtility (contracts/StakingUtility.sol#8-75)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

StakingData._poolCapacity (contracts/StakingData.sol#23) should be constant
StakingData._poolServiceFee (contracts/StakingData.sol#29) should be constant
StakingData._reserveRewards (contracts/StakingData.sol#20) should be constant
StakingData._reservingNodeOff (contracts/StakingData.sol#38) should be constant
StakingData._unlockEpoch (contracts/StakingData.sol#26) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
```

- No major issues found by Slither.
- The reentrancies flagged by Slither are false positives. The _currencyToken can only be set during the contract deployment by the owner and should never be a malicious token.

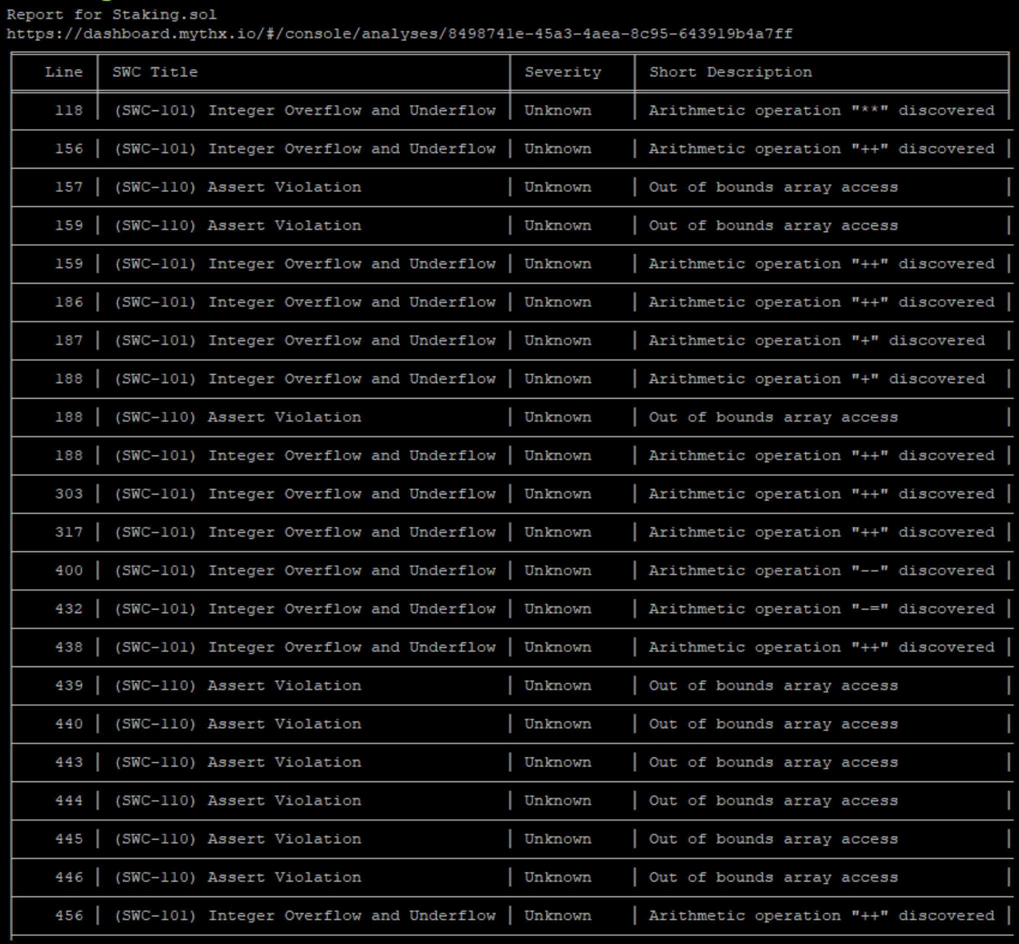
4.2 AUTOMATED SECURITY SCAN

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 all the contracts and sent the compiled results to the analyzers to locate any vulnerabilities.

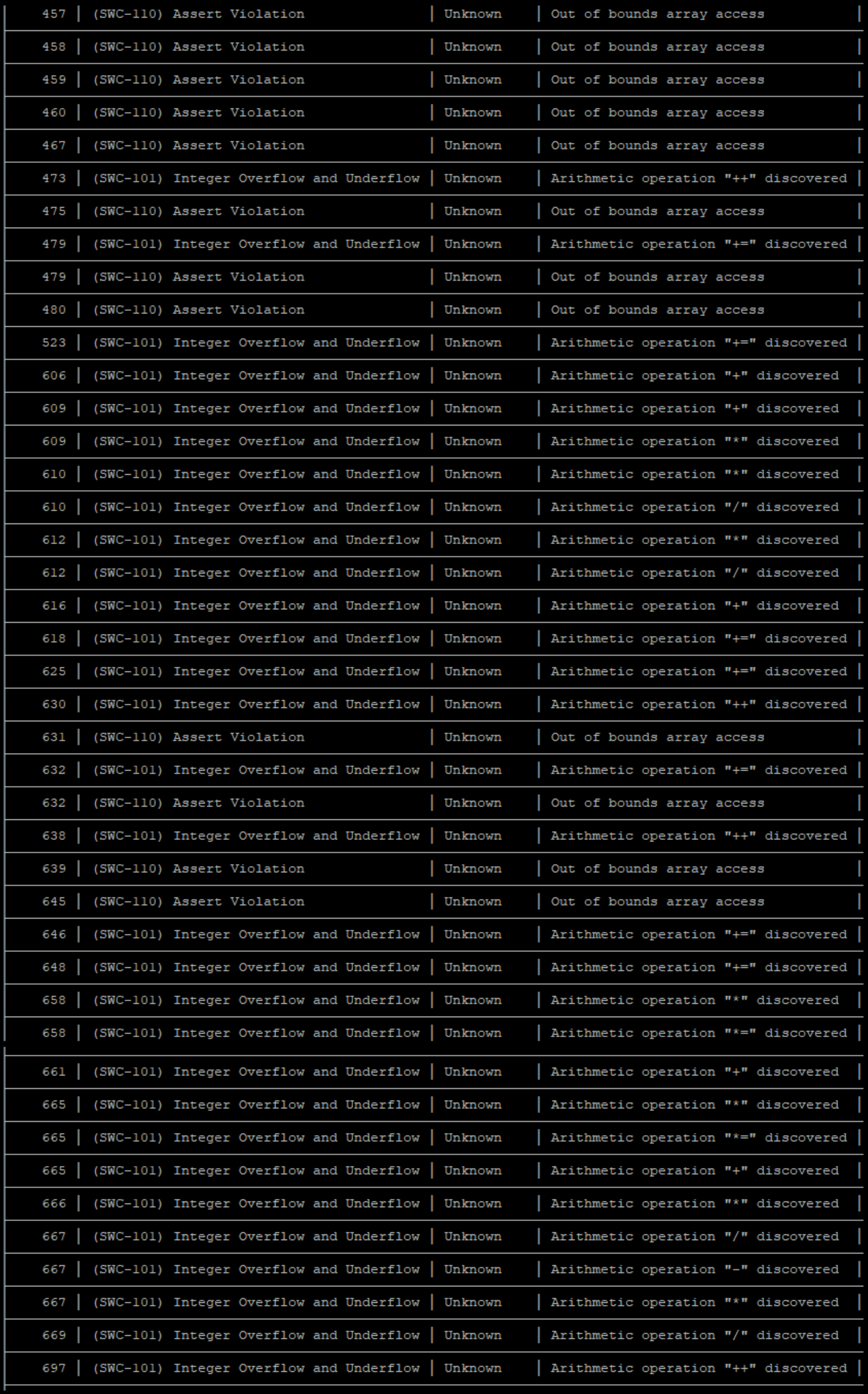
MythX results:

Staking.sol

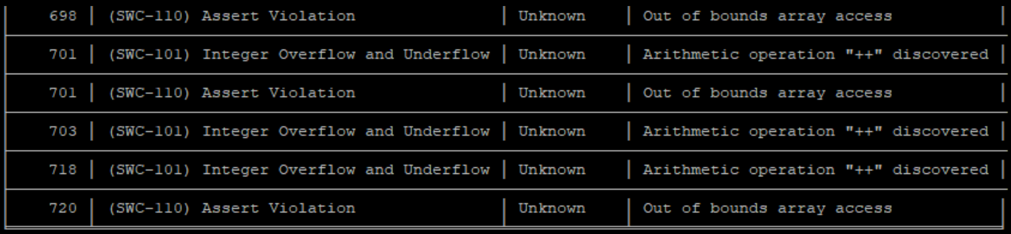


```
Report for Staking.sol
https://dashboard.mythx.io/#/console/analyses/8498741e-45a3-4aea-8c95-643919b4a7ff

  Line | SWC Title                               | Severity | Short Description
   118 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "**" discovered |
   156 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "++" discovered |
   157 | (SWC-110) Assert Violation               | Unknown  | Out of bounds array access           |
   159 | (SWC-110) Assert Violation               | Unknown  | Out of bounds array access           |
   159 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "++" discovered |
   186 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "++" discovered |
   187 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "+" discovered  |
   188 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "+" discovered  |
   188 | (SWC-110) Assert Violation               | Unknown  | Out of bounds array access           |
   188 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "++" discovered |
   303 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "++" discovered |
   317 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "++" discovered |
   400 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "--" discovered |
   432 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "-=" discovered |
   438 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "++" discovered |
   439 | (SWC-110) Assert Violation               | Unknown  | Out of bounds array access           |
   440 | (SWC-110) Assert Violation               | Unknown  | Out of bounds array access           |
   443 | (SWC-110) Assert Violation               | Unknown  | Out of bounds array access           |
   444 | (SWC-110) Assert Violation               | Unknown  | Out of bounds array access           |
   445 | (SWC-110) Assert Violation               | Unknown  | Out of bounds array access           |
   446 | (SWC-110) Assert Violation               | Unknown  | Out of bounds array access           |
   456 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "++" discovered |
```

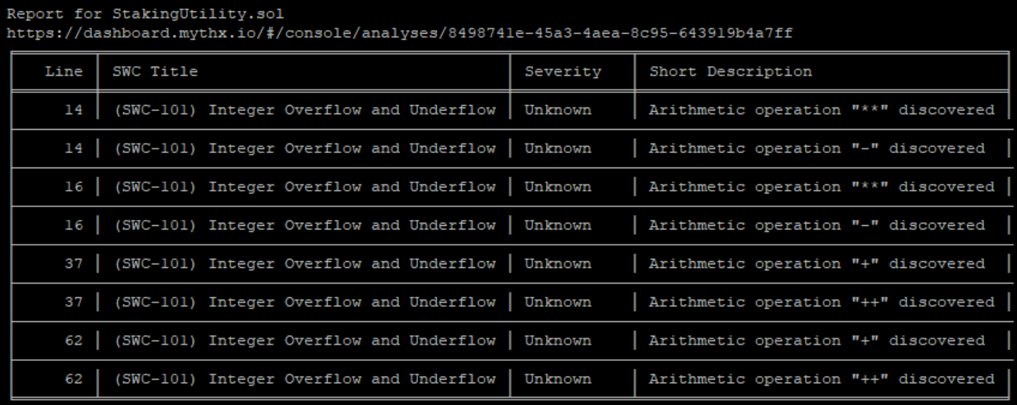


```
| 457 | (SWC-110) Assert Violation                 | Unknown | Out of bounds array access           |
| 458 | (SWC-110) Assert Violation                 | Unknown | Out of bounds array access           |
| 459 | (SWC-110) Assert Violation                 | Unknown | Out of bounds array access           |
| 460 | (SWC-110) Assert Violation                 | Unknown | Out of bounds array access           |
| 467 | (SWC-110) Assert Violation                 | Unknown | Out of bounds array access           |
| 473 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "++" discovered |
| 475 | (SWC-110) Assert Violation                 | Unknown | Out of bounds array access           |
| 479 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "+=" discovered |
| 479 | (SWC-110) Assert Violation                 | Unknown | Out of bounds array access           |
| 480 | (SWC-110) Assert Violation                 | Unknown | Out of bounds array access           |
| 523 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "+=" discovered |
| 606 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "+" discovered  |
| 609 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "+" discovered  |
| 609 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "*" discovered  |
| 610 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "*" discovered  |
| 610 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "/" discovered  |
| 612 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "*" discovered  |
| 612 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "/" discovered  |
| 616 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "+" discovered  |
| 618 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "+=" discovered |
| 625 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "+=" discovered |
| 630 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "++" discovered |
| 631 | (SWC-110) Assert Violation                 | Unknown | Out of bounds array access           |
| 632 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "+=" discovered |
| 632 | (SWC-110) Assert Violation                 | Unknown | Out of bounds array access           |
| 638 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "++" discovered |
| 639 | (SWC-110) Assert Violation                 | Unknown | Out of bounds array access           |
| 645 | (SWC-110) Assert Violation                 | Unknown | Out of bounds array access           |
| 646 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "+=" discovered |
| 648 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "+=" discovered |
| 658 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "*" discovered  |
| 658 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "*=" discovered |
| 661 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "+" discovered  |
| 665 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "*" discovered  |
| 665 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "*=" discovered |
| 665 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "+" discovered  |
| 666 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "*" discovered  |
| 667 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "/" discovered  |
| 667 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "-" discovered  |
| 667 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "*" discovered  |
| 669 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "/" discovered  |
| 697 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "++" discovered |
```



```
|  698 | (SWC-110) Assert Violation                 | Unknown  | Out of bounds array access           |
|  701 | (SWC-101) Integer Overflow and Underflow   | Unknown  | Arithmetic operation "++" discovered |
|  701 | (SWC-110) Assert Violation                 | Unknown  | Out of bounds array access           |
|  703 | (SWC-101) Integer Overflow and Underflow   | Unknown  | Arithmetic operation "++" discovered |
|  718 | (SWC-101) Integer Overflow and Underflow   | Unknown  | Arithmetic operation "++" discovered |
|  720 | (SWC-110) Assert Violation                 | Unknown  | Out of bounds array access           |
```

StakingData.sol



```
Report for StakingUtility.sol
https://dashboard.mythx.io/#/console/analyses/8498741e-45a3-4aea-8c95-643919b4a7ff
| Line | SWC Title                                  | Severity | Short Description                    |
|   14 | (SWC-101) Integer Overflow and Underflow   | Unknown  | Arithmetic operation "**" discovered |
|   14 | (SWC-101) Integer Overflow and Underflow   | Unknown  | Arithmetic operation "-" discovered  |
|   16 | (SWC-101) Integer Overflow and Underflow   | Unknown  | Arithmetic operation "**" discovered |
|   16 | (SWC-101) Integer Overflow and Underflow   | Unknown  | Arithmetic operation "-" discovered  |
|   37 | (SWC-101) Integer Overflow and Underflow   | Unknown  | Arithmetic operation "+" discovered  |
|   37 | (SWC-101) Integer Overflow and Underflow   | Unknown  | Arithmetic operation "++" discovered |
|   62 | (SWC-101) Integer Overflow and Underflow   | Unknown  | Arithmetic operation "+" discovered  |
|   62 | (SWC-101) Integer Overflow and Underflow   | Unknown  | Arithmetic operation "++" discovered |
```

StakingUtility.sol

No issues found by MythX in this smart contract.

- The Integer Overflows and Underflows flagged by MythX were checked individually and were determined to be mathematically impossible.
- Assert violations are false positives.

THANK YOU FOR CHOOSING

// HALBORN