

**HACKEN**

# SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

**Customer:** Asymetrix  
**Date:** 26 Oct, 2023

This report may contain confidential information about IT systems and the intellectual property of the Customer, as well as information about potential vulnerabilities and methods of their exploitation.

The report can be disclosed publicly after prior consent by another Party. Any subsequent publication of this report shall be without mandatory consent.

## Document

<b>Name</b>	Smart Contract Code Review and Security Analysis Report for Asymetrix
<b>Approved By</b>	Luis Buendia   Senior Solidity SC Auditor at Hacken OÜ Grzegorz Trawiński   Lead Solidity SC Auditor at Hacken OU
<b>Tags</b>	ERC20 token; Staking; Yielding; Uniswap; Balancer; Oracle
<b>Platform</b>	EVM
<b>Language</b>	Solidity
<b>Methodology</b>	<a href="#">Link</a>
<b>Website</b>	<a href="https://asymetrix.io">https://asymetrix.io</a>
<b>Changelog</b>	11.10.2023 - Initial Review 21.10.2023 - Second Review 26.10.2023 - Third Review

## Table of contents

<b>Introduction</b>	<b>4</b>
<b>System Overview</b>	<b>4</b>
<b>Executive Summary</b>	<b>5</b>
<b>Risks</b>	<b>6</b>
<b>Findings</b>	<b>7</b>
Critical	7
High	7
H01. Counter Not Updated to User	7
H02. Claim Rewards May Revert for Non-boosted Positions	8
Medium	9
M01. Price Oracle Manipulation	9
M02. Usage of Deprecated ChainLink Oracle Function	10
M03. unstake and extendLock May Revert Due to Access Control Modifier	11
M04. Slippage Control Can Revert Due to Inaccurate Calculation	11
Low	13
L01. createVesting Position on Claim Transaction Can Revert	13
Informational	14
I01. Use Custom Errors	14
I02. Initialized Variable to Default Value	14
I03. Shift Instead of Divide to Save Gas	15
I04. Using Booleans for Storage Incurs Overhead	15
I05. validateStake Function Return Value Is Never Used	15
I06. Misleading NatSpec Documentation	16
I07. Centralization Risk	16
I08. Owner Can Renounce Itself	17
I09. Redeem Function Does Nothing	17
<b>Disclaimers</b>	<b>18</b>
<b>Appendix 1. Severity Definitions</b>	<b>19</b>
Risk Levels	19
Impact Levels	20
Likelihood Levels	20
Informational	20
<b>Appendix 2. Scope</b>	<b>21</b>

## Introduction

Hacken OÜ (Consultant) was contracted by Asymetrix (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of the Customer's smart contracts.

## System Overview

*Asymetrix protocol* is the decentralized, non-custodial protocol for asymmetric yield distribution generated from staking. The files in the scope:

- **StakePrizePoolv2** - Pool where users deposit stEth ERC20 token. Owner adds prizes manually. Also, rewards are distributed among all the contributors.
- **OracleUniswapV3** - Contract that given a TWAP period obtains the price of a token for a given pool. The protocol uses it to obtain the ASX price in terms of ETH.
- **OracleBalancerWeighted** - Contract that obtains the price of ASX in terms of ETH using the liquidity on a given balancer pool.
- **ASXPriceFeed** - Contract that returns the price of ASX by computing an average price between the uniswap and balancer oracles.
- **ValuerUniswapV3** - Contract that returns the price in USD of a uniswapv3 pool position.
- **ValuerBalancer** - Contract that returns the price in USD of a balancer pool position.
- **UniswapWrapper** - Contract that performs a swap on a uniswapv3 pool.
- **RewardsBooster** - Contract that enables users to stake their LP positions of ASX token and boost their rewards on the stake prize pool.
- **ValidatorUniswapV3** - Contract that validates if a position is legit on a given pool of uniswap v3.
- **ValidatorBalancerWeighted** - Contract that validates if a position is legit on a balancer pool.
- **ESASX** - Escrowed ASX. Is a non-tradable ERC20 token, used to reward protocol contributors.
- **ESASVesting** - Contract that handles the ESASX vesting tokens from users.

## Privileged roles

- The owner of each contract can set and change multiple configuration values and also access restricted functionalities.
- The manager/operator can access certain restricted functionalities.

## Executive Summary

The score measurement details can be found in the corresponding section of the [scoring methodology](#).

### Documentation quality

The total Documentation Quality score is **10** out of **10**.

- High level overview documentation has been created.
- Technical descriptions have been provided and NatSpec is extensive.

### Code quality

The total Code Quality score is **10** out of **10**.

- The code is structured and readable.
- The Gas model is optimized.

### Test coverage

Code coverage of the project is **97%** (branch coverage)

- Deployment and some basic user interactions are covered with tests.
- The extended code coverage was done after the audit.

### Security score

As a result of the audit, the code contains **1** medium severity issue. The security score is **9** out of **10**.

All found issues are displayed in the “Findings” section.

### Summary

According to the assessment, the Customer's smart contract has the following score: **9.7**. The system users should acknowledge all the risks summed up in the risks section of the report.



The final score 

*Table. The distribution of issues during the audit*

Review date	Low	Medium	High	Critical
11 October 2023	1	3	3	0
21 October 2023	0	1	0	0
26 October 2023	0	1	0	0



## Risks

- The protocol uses illiquid pools to compute token prices. Although the exploitation may not be direct it can represent an issue for the protocol if it is not treated properly. Furthermore, using the balance of a liquidity pool as an oracle is heavily discouraged.

## Findings

### Critical

No critical severity issues were found.

### High

#### H01. Counter Not Updated to User

Impact	Medium
Likelihood	High

The function `_createVestingPosition` from the `ESASXVesting.sol` contract does not properly handle the stored users' vesting positions. The function uses the `msg.sender` to account for the correct position and increments the counter of the address introduced as parameter.

```
function _createVestingPosition(address _user, uint256 _amount) private {
    uint256 _vestingPositionsCount = vestingPositionsCount[msg.sender];

    VestingPosition memory vestingPosition = VestingPosition({
        lockPeriod: vestingPeriod,
        amount: _amount,
        releasedAmount: 0,
        createdAt: uint32(block.timestamp)
    });

    vestingPositions[_user][_vestingPositionsCount] = vestingPosition;
    vestingPositionsCount[_user] += 1;
    totalVestedAmount += _amount;
    emit VestingPositionCreated(_vestingPositionsCount, _user, vestingPosition);
}
```

The external function `createVestingPosition` apparently should just be called by the `PrizePool` whenever a user has ESASX rewards. As ESASX is a non-tradable token, it is transferred from the Pool to the ESASX `Vesting` contract, creating a vesting position for the user.

So if a user has two times rewards of ESASX that can be vested, the second one will overwrite the previous one and so on, producing a loss of funds.

**Path:** `./contracts/vesting/ESASXVesting.sol : _createVestingPosition()`





**Path:** ./contracts/core/prize-pool/PrizePoolV2.sol : claim()

**Recommendation:** The fix should allow users that provided stEth without boost to claim their corresponding rewards.

**Found in:** 3d1b247ce4223ef2bc8ec51a2f893a18040b8b40

**Status:** **Fixed** (Revised commit: 876a5a463c42)

**Remediation:** The contract now allows to claim rewards for non-boosted positions because the `getBoost` function returns `zero` and `false` instead of reverting.

## ■ ■ Medium

### M01. Price Oracle Manipulation

Impact	Medium
Likelihood	Medium

The `latestAnswer` function of the `OracleBalancerWeighted.sol` contract returns the price of the ASX token in USD. The function relies on the available liquidity of the Balancer pool ASX/WETH to determine the price of ASX in terms of WETH. Then it uses ChainLink oracle to obtain the price of WETH in USD for the final calculation.

However, using directly the liquidity of a pool to compute prices is a risk as it can be easily manipulated. Furthermore, when dealing with illiquid pools.

The protocol uses this oracle with the Uniswap v3 oracle pool that implements the TWAP mechanism to avoid the previous scenario. Nevertheless, the final price is calculated using an arithmetic mean between the two oracle values, which can still result in a price manipulation.

**Proof of Concept:** The next test case illustrates the previous described behavior.

1. Obtain the calculated price of the `ASXPriceFeed.sol`
2. Swap 3 ether on the Balancer pool
3. Obtain again the manipulated price of the `ASXPriceFeed.sol`

```
PS C:\...\.asymetrix\feature-esASXVesting> forge test --match-test test_oracle_values
Running 1 test for tests/Test.t.sol:Deployer
[PASS] test_oracle_values() (gas: 463630)
Logs:
asxPfeed Before      232968727062082856
asxPfeed After      308171828483522490
```

As it is possible to observe the difference in price, even doing the arithmetic mean with Uniswap v3 result using TWAP, is about 0.7 USD per ASX in price with just 3 ETH.

**Path:** `./contracts/rewards-booster/OracleBalancerWeighted.sol:latestAnswer()`

**Recommendation:** There are many valid approaches to mitigate the issue. The main concept here is not to directly use the balance of a liquidity pool to compute a price. This same issue is present on the Eth oracles as well not included in the scope of the audit.

**Found in:** 3d1b247ce4223ef2bc8ec51a2f893a18040b8b40

**Status:** **Acknowledged**

**Remediation:** Asymetrix team considers that as there are only two on-chain price sources this design meets their requirements and ensures proper functionality.

## M02. Usage of Deprecated ChainLink Oracle Function

Impact	Medium
Likelihood	Medium

The `latestAnswer` ChainLink oracle function is deprecated. As the documentation specifies, it is strongly recommended not to use this function.

Reference:

<https://docs.chain.link/data-feeds/api-reference#latestanswer>

Using a deprecated function to obtain market prices is a serious hazard for the protocol.

**Path:** `./contracts/rewards-booster/oracles/OracleUniswapV3.sol:getLatestAnswer()`

`./contracts/rewards-booster/oracles/OracleBalancerWeighted.sol:latestAnswer()`

`./contracts/rewards-booster/oracles/OracleBalancerWeighted.sol:latestAnswer()`

**Recommendation:** Consider using the `latestRoundData` function or any other non deprecated one.

<https://docs.chain.link/data-feeds/api-reference>

If `latestRoundData` is used consider following best practices to control stale prices.

**Found in:** 3d1b247ce4223ef2bc8ec51a2f893a18040b8b40

**Status:** **Fixed** (Revised commit: 876a5a463c42)

**Remediation:** The code uses the current working Chainlink function and the associated best practices as well.

### M03. `unstake` and `extendLock` May Revert Due to Access Control Modifier

Impact	Medium
Likelihood	Medium

The functions `unstake` and `extendLock` from the `RewardsBooster.sol` contract, whenever dealing with a staked position of uniswap, call the public function `buybackAndBurnAsx` of the same contract. The function `buybackAndBurnAsx` is protected with an `onlyOwner` modifier. Which means, users are not able to unstake/extend their positions on the contract.

This has serious effects by locking user funds on the contract.

**Path:** `./contracts/rewards-booster/RewardsBooster.sol` : `unstake()`  
`externdLock()`

**Recommendation:** Consider creating a private version of the external one that collects the fees from uniswap v3 positions and then call the internal `_buybackAndBurn`. Alternatively, create an internal function to collect the fees, then call the internal `_buybackAndBurn` function.

**Found in:** 3d1b247ce4223ef2bc8ec51a2f893a18040b8b40

**Status:** **Fixed** (Revised commit: 876a5a463c42)

**Remediation:** The access control modifier was removed.

### M04. Slippage Control Can Revert Due to Inaccurate Calculation

Impact	Medium
Likelihood	Medium

The `_buybackAndBurn` function uses the price obtained through the `ASXPriceFeed.sol` contract to compute the `amountOut` of the swap and thus, the corresponding slippage. The oracle returns the arithmetic average value of the balancer and uniswap pools ASX price in ETH.

The discrepancy between the pool prices can result in a higher price than what the price of Uniswap v3 actually is. So, even subtracting the slippage can result in a higher price than what uniswap may return.

```
function _buybackAndBurn(
    uint256 _wethAmount,
    uint256 _asxAmount,
    bool _useNative
) private returns (uint256 _asxPriceInWeth) {
    IOracle _asxOracle = asxOracle;

    _asxPriceInWeth = (uint256(_asxOracle.latestAnswer()) * 1e18) /
    _asxOracle.decimals();

    uint256 _amountOut = _wethAmount / _asxPriceInWeth;
    uint256 _amountOutMin = _amountOut - (( _amountOut * slippageTolerance) /
    ONE_HUNDRED_PERCENTS);
    address _asx = address(rewardToken);
    uint256 _swappedAsxAmount = uniswapWrapper.swapSingle{ value: _useNative ?
    msg.value : 0 }(
```

This can result in blocking many workflows of the protocol.

**Path:** `./contracts/core/prize-pool/PrizePoolV2.sol : _buybackAndBurn()`  
`./contracts/vesting/Buyback.sol : _buybackAndBurn()`

**Recommendation:** Consider using Uniswap methods to obtain the real amount that can be swapped from the pool. To avoid price fluctuations, compare that price with the obtained from the oracle to add an extra layer of security.

**Found in:** `3d1b247ce4223ef2bc8ec51a2f893a18040b8b40`

**Status:** **Fixed.**

**Remediation:** Although there is no change on the code base, the Asymetrix development team decided not to use the Balancer oracle for this operation.

■ Low

L01. *createVesting* Position on Claim Transaction Can Revert

Impact	Low
Likelihood	Low

The `_claimEsAsxAndVest` internal function from the `PrizePoolV2.sol` contract is triggered every time the external `claim` function is called. When a user has enough ESASX rewards, the pool creates a vesting position with those rewards calling the `ESASXVesting` contract.

```

if (_esAsxReward >= _esAsxVesting.getMinVestingAmount()) {
    _esAsx.approve(address(_esAsxVesting), _esAsxReward);
    _esAsxVesting.createVestingPosition(_user, _esAsxReward);
} else {
    userInfo.esAsxBoostlessReward += _esAsxReward;
}

```

If the amount is bigger than the min amount, the pool adds those rewards to the `esAsxBoostlessReward`. However, the `createVestingPosition` function has other constraint. If the amount is bigger than the withdrawable amount the transaction will revert, preventing the user from claiming the rewards.

```

function createVestingPosition(address _user, uint256 _amount) external {
    if (_amount < minVestingAmount) revert EsAsxErrors.WrongVestingAmount();
    if (getWithdrawableASXAmount() < _amount) revert
EsAsxErrors.InvalidEsASXAmount();

    esASX.safeTransferFrom(msg.sender, address(this), _amount);
    _createVestingPosition(_user, _amount);
}

```

**Path:** `./contracts/core/prize-pool/PrizePoolV2.sol` : `_claimEsAsxAndVest()`

**Recommendation:** Control the withdrawable amount before calling the create vesting function.

**Found in:** 3d1b247ce4223ef2bc8ec51a2f893a18040b8b40

**Status:** Fixed (Revised commit: 876a5a463c42)

**Remediation:** All the possible scenarios are now controlled before the function call.

## Informational

### I01. Use Custom Errors

Custom errors from Solidity 0.8.4 are cheaper than revert strings (cheaper deployment cost and runtime cost when the revert condition is met). Source Custom Errors in Solidity: Starting from Solidity v0.8.4, there is a convenient and gas-efficient way to explain to users why an operation failed through the use of custom errors. Until now, you could already use strings to give more information about failures (e.g., `revert("Insufficient funds.");`), but they are rather expensive, especially when it comes to deployment cost, and it is difficult to use dynamic information in them.

**Path:** `./contracts/core/prize-pool/PrizePoolV2.sol`

`./contracts/core/prize-pool/StakePrizePoolV2.sol`

**Recommendation:** Consider replacing strings for custom errors as done in the rest of the protocol implementation.

**Found in:** `3d1b247ce4223ef2bc8ec51a2f893a18040b8b40`

**Status:** **Fixed** (Revised commit: `876a5a463c42`)

**Remediation:** Custom errors were implemented and used.

### I02. Initialized Variable to Default Value

Initializing variables to default value executes an extra order that is not required.

**Path:** `./contracts/rewards-booster/oracles/ASXPriceFeed.sol`:  
`constructor, latestAnswer`

`./contracts/core/prize-pool/StakePrizePoolV2.sol`: `awardExternalERC721`

`./contracts/vesting/ESASXVesting.sol`: `release, releaseWithPenalty`

`./contracts/rewards-booster/Rewardsbooster.sol`:  
`_setLockDurationSettings, _getAdditionalLockDurationBoost`

**Recommendation:** Consider avoiding initializing variables to default value.

**Found in:** `3d1b247ce4223ef2bc8ec51a2f893a18040b8b40`

**Status:** **Fixed** (Revised commit: `876a5a463c42`)

**Remediation:** Initialized variables to default values were removed.

### I03. Shift Instead of Divide to Save Gas

While the DIV opcode uses 5 Gas, the SHR opcode only uses 3 Gas. Furthermore, Solidity's division operation also includes a division-by-0 prevention, which is bypassed using shifting.

**Path:** ./contracts/core/prize-pool/StakePrizePoolV2.sol: \_liquidate()  
./contracts/vesting/ESASXVesting.sol: buyEsAsxWithDiscount  
./contracts/rewards-booster/Rewardsbooster.sol: \_getBoost,  
\_getLockDurationBoostCoefficient

**Recommendation:** Consider using shift operator instead of dividing by a constant to save Gas.

**Found in:** 3d1b247ce4223ef2bc8ec51a2f893a18040b8b40

**Status:** **Fixed** (Revised commit: 876a5a463c42)

**Remediation:** All constant divisions of simple multiples were changed for shifting.

### I04. Using Bools for Storage Incurs Overhead

Use uint256(1) and uint256(2) for true/false to avoid a Gwarmaccess (100 Gas), and to avoid Gsset (20000 Gas) when changing from 'false' to 'true', after having been 'true' in the past.

See reference:

[openzeppelin-contracts/contracts/security/ReentrancyGuard.sol at 58f635312aa21f947cae5f8578638a85aa2519f5 · OpenZeppelin/contracts · GitHub](https://github.com/OpenZeppelin/contracts/blob/master/contracts/security/ReentrancyGuard.sol#L58f635312aa21f947cae5f8578638a85aa2519f5)

**Path:** ./contracts/ESASX.sol  
./contracts/rewards-booster/RewardsBooster.sol

**Recommendation:** Consider avoiding the usage of boolean types for storage variables.

**Found in:** 3d1b247ce4223ef2bc8ec51a2f893a18040b8b40

**Status:** **Fixed** (Revised commit: 876a5a463c42)

**Remediation:** Boolean values were removed from storage for unsigned integers.

### I05. validateStake Function Return Value Is Never Used

The validators' contracts are used to validate if a position is valid either on Balancer or Uniswap v3. They implement a function called

*validateStake* and return a boolean value if the position exists and revert if not. The returned boolean value is never used.

**Recommendation:** Consider not returning any value or return false if the position does not exist instead of reverting to be consistent with the code.

**Found in:** 3d1b247ce4223ef2bc8ec51a2f893a18040b8b40

**Status:** **Fixed** (Revised commit: 876a5a463c42)

**Remediation:** The return value was removed and the function reverts if the position is not valid.

## I06. Misleading NatSpec Documentation

The NatSpec documentation does not seem to be accurate in all comments with the implemented code.

Ex: On the stake function of RewardsBooster.sol contract the notice states “Accepts deposits from users and stakes them in the specified staking pool”. However, the function takes the position of an authorized pool and stakes them in the contract to boost rewards.

**Recommendation:** Consider reading carefully the documentation to match the code behavior.

**Found in:** 3d1b247ce4223ef2bc8ec51a2f893a18040b8b40

**Status:** **Fixed** (Revised commit: 876a5a463c42)

**Remediation:** The documentation was improved with even more level of detail and accuracy.

## I07. Centralization Risk

The protocol is heavily centralized. This is not a risk by itself. However, it is important to notice that any unauthorized access to the owner accounts can jeopardize the protocol stability.

**Recommendation:** Use multisignature wallet for privileged accounts.

**Found in:** 3d1b247ce4223ef2bc8ec51a2f893a18040b8b40

**Status:** **Mitigated**

**Remediation:** The contract owner is a multisignature wallet with a  $\frac{2}{3}$  scheme. The Asymetrix team does not implement any changes before announcing it with the community. On future releases, a DAO and a Timelock contract will be deployed to provide a more decentralized environment.



### I08. Owner Can Renounce Itself

The owner can renounce itself creating a problem to operate with high privileged functions.

**Recommendation:** Consider erasing the renounce owner function.

**Found in:** 3d1b247ce4223ef2bc8ec51a2f893a18040b8b40

**Status:** Mitigated

**Remediation:** Given the multisignature wallet, it is highly unlikely to arrive in this scenario.

### I09. Redeem Function Does Nothing

The redeem function used on the PrizePoolV2.sol and implemented on the StakePrizePoolV2.sol returns the same amount introduced as parameter.

**Path:** ./contracts/core/prize-pool/StakePrizePoolV2.sol : \_redeem()

**Recommendation:** Consider simplifying code if there are unnecessary sections.

**Found in:** 3d1b247ce4223ef2bc8ec51a2f893a18040b8b40

**Status:** Fixed (Revised commit: 876a5a463c42)

**Remediation:** The unused functions were removed.

## Disclaimers

### **Hacken Disclaimer**

The smart contracts given for audit have been analyzed based on best industry practices at the time of the writing of this report, with cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The report contains no statements or warranties on the identification of all vulnerabilities and security of the code. The report covers the code submitted and reviewed, so it may not be relevant after any modifications. Do not consider this report as a final and sufficient assessment regarding the utility and safety of the code, bug-free status, or any other contract statements.

While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only – we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

English is the original language of the report. The Consultant is not responsible for the correctness of the translated versions.

### **Technical Disclaimer**

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks. Thus, the Consultant cannot guarantee the explicit security of the audited smart contracts.

## Appendix 1. Severity Definitions

When auditing smart contracts Hacken is using a risk-based approach that considers the potential impact of any vulnerabilities and the likelihood of them being exploited. The matrix of impact and likelihood is a commonly used tool in risk management to help assess and prioritize risks.

The impact of a vulnerability refers to the potential harm that could result if it were to be exploited. For smart contracts, this could include the loss of funds or assets, unauthorized access or control, or reputational damage.

The likelihood of a vulnerability being exploited is determined by considering the likelihood of an attack occurring, the level of skill or resources required to exploit the vulnerability, and the presence of any mitigating controls that could reduce the likelihood of exploitation.

Risk Level	High Impact	Medium Impact	Low Impact
High Likelihood	Critical	High	Medium
Medium Likelihood	High	Medium	Low
Low Likelihood	Medium	Low	Low

### Risk Levels

**Critical:** Critical vulnerabilities are usually straightforward to exploit and can lead to the loss of user funds or contract state manipulation.

**High:** High vulnerabilities are usually harder to exploit, requiring specific conditions, or have a more limited scope, but can still lead to the loss of user funds or contract state manipulation.

**Medium:** Medium vulnerabilities are usually limited to state manipulations and, in most cases, cannot lead to asset loss. Contradictions and requirements violations. Major deviations from best practices are also in this category.

**Low:** Major deviations from best practices or major Gas inefficiency. These issues won't have a significant impact on code execution, don't affect security score but can affect code quality score.

## Impact Levels

**High Impact:** Risks that have a high impact are associated with financial losses, reputational damage, or major alterations to contract state. High impact issues typically involve invalid calculations, denial of service, token supply manipulation, and data consistency, but are not limited to those categories.

**Medium Impact:** Risks that have a medium impact could result in financial losses, reputational damage, or minor contract state manipulation. These risks can also be associated with undocumented behavior or violations of requirements.

**Low Impact:** Risks that have a low impact cannot lead to financial losses or state manipulation. These risks are typically related to unscalable functionality, contradictions, inconsistent data, or major violations of best practices.

## Likelihood Levels

**High Likelihood:** Risks that have a high likelihood are those that are expected to occur frequently or are very likely to occur. These risks could be the result of known vulnerabilities or weaknesses in the contract, or could be the result of external factors such as attacks or exploits targeting similar contracts.

**Medium Likelihood:** Risks that have a medium likelihood are those that are possible but not as likely to occur as those in the high likelihood category. These risks could be the result of less severe vulnerabilities or weaknesses in the contract, or could be the result of less targeted attacks or exploits.

**Low Likelihood:** Risks that have a low likelihood are those that are unlikely to occur, but still possible. These risks could be the result of very specific or complex vulnerabilities or weaknesses in the contract, or could be the result of highly targeted attacks or exploits.

## Informational

Informational issues are mostly connected to violations of best practices, typos in code, violations of code style, and dead or redundant code.

Informational issues are not affecting the score, but addressing them will be beneficial for the project.

## Appendix 2. Scope

The scope of the project includes the following smart contracts from the provided repository:

### Initial review scope

<b>Repository</b>	<a href="https://bitbucket.ideasoft.io/projects/PB0N/repos/solidity/commits?until=refs%2Fheads%2Ffeature%2FesASXVesting">https://bitbucket.ideasoft.io/projects/PB0N/repos/solidity/commits?until=refs%2Fheads%2Ffeature%2FesASXVesting</a>
<b>Commit</b>	3d1b247ce4223ef2bc8ec51a2f893a18040b8b40
<b>Whitepaper</b>	N/A
<b>Requirements</b>	N/A
<b>Technical Requirements</b>	N/A
<b>Contracts</b>	<p>File: contracts/rewards-booster/RewardsBooster.sol          SHA3:          ade74f5f369183abfbfe6bd6e1c73fb1052567538d847cb7b60c815017f0049b</p> <p>File: contracts/vesting/UniswapWrapper.sol          SHA3          bf484b6f59020cd3db7e398a314f1ac5090e87a6e6d6a160024a238894ae254e</p> <p>File: contracts/core/prize-pool/PrizePoolV2.sol          SHA3: 3e72af37891aec162cceca4d88f858373562ab69a0785e33b65159a2c2267b18</p> <p>File: contracts/core/prize-pool/StakePrizePoolV2.sol          SHA3          aca409b72cfcf8a680a9df56684f52b627491cb2b1992517b73ed0f0858dc1d0</p> <p>File: contracts/vesting/ESASXVesting.sol          SHA3: de2033ef1302fa47e5fca3a0479f9ce51b9ba0189e0584ce91f7c3aafc2c8de</p> <p>File: contracts/rewards-booster/valuers/ValuerUniswapV3.sol          SHA3          10830db7eedb715acef5a2ff260741244e24a70e194ad26739b9585a3836888b</p> <p>File:          contracts/rewards-booster/valuers/ValuerBalancerWeighted.sol          SHA3: 2e214d339f60f0cfbdd8b3b5e497104bd82c500d82d2ce7ec85c670cf8483cd0</p> <p>File:          contracts/rewards-booster/validators/ValidatorUniswapV3.sol          SHA3: 8e5d92de1c7bbb54ef353f3773f2f45ee4beb517d9e027860028217249f60573</p>

	<p>File:  contracts/rewards-booster/validators/ValidatorBalancerWeighted.sol  SHA3: f5cdbae415f6c0157609a7c66db7f6b955c126b86de1a65d2d10c19d9ca94ac2</p> <p>File: contracts/rewards-booster/oracles/OracleUniswapV3.sol  SHA3: 12a50339fcc14d73e2c1ae9d5a488134715a6d918f9dad99ffc97da48ed897ea</p> <p>File:  contracts/rewards-booster/oracles/OracleBalancerWeighted.sol  SHA3: da6add0471d4e472f4b45a7746ceb1fe9022feec8e050df047ee5ebdf210b994</p> <p>File: contracts/rewards-booster/oracles/ASXPriceFeed.sol  SHA3: 67646ea077d0ca9f6e98250fc3a120e1c711ca81ec44c5839a6e9dd6d38f57b2</p>
--	--

## Second review scope

<b>Repository</b>	<a href="https://bitbucket.ideasoft.io/projects/PB0N/repos/solidity/commits/876a5a463c42c43613b49bcff313a0bceec68fe5e">https://bitbucket.ideasoft.io/projects/PB0N/repos/solidity/commits/876a5a463c42c43613b49bcff313a0bceec68fe5e</a>
<b>Commit</b>	876a5a463c42c43613b49bcff313a0bceec68fe5e
<b>Whitepaper</b>	N/A
<b>Requirements</b>	N/A
<b>Technical Requirements</b>	N/A
<b>Contracts</b>	<p>File: contracts/rewards-booster/RewardsBooster.sol  SHA3:  748daf7b5dea0dc9c464017051fe60c3b2cca3b16e54459a59320511053a4cb  d</p> <p>File: contracts/vesting/UniswapWrapper.sol  SHA3  ee8786bc2172a199a579c754a0e832e846bc6203f481c23a7c0289f9b62ba96  d</p> <p>File: contracts/core/prize-pool/PrizePoolV2.sol  SHA3: 12d6a40af8138fa42126e2eeb9f55775743bf94d53728a002828176aab  514cce</p> <p>File: contracts/core/prize-pool/StakePrizePoolV2.sol  SHA3  45d6096296319ce9fd0b41ad4c68329b56a253acc1b6509b42fc4dfe4d0bfff4  d</p> <p>File: contracts/vesting/ESASXVesting.sol  SHA3: 29fc4b47c095a9a5deebf4cba59a8faacc83fd537d0ad0d46ce8b1cc86  8ebc22</p>

<p>File: contracts/rewards-booster/valuers/ValuerUniswapV3.sol          SHA3: 09e1c9ea8c903491e2b603af7c394cbee0d1ec4c7e925b2f920be6cbd8d95378</p> <p>File:          contracts/rewards-booster/valuers/ValuerBalancerWeighted.sol          SHA3: cf6e9bb6b180ac6dea7f7c6dc3a48834dc9d9d1e52fbfbf09c45b6413591e3b8</p> <p>File:          contracts/rewards-booster/validators/ValidatorUniswapV3.sol          SHA3: 4db1290338560c8de827b36a9bc7c5a41321b04642b69dae4d70e42296422b79</p> <p>File:          contracts/rewards-booster/validators/ValidatorBalancerWeighted.sol          SHA3: acb797565a49ece95b70e82b9502359a60e5d08221f8169bb9f6b6b30998bf31</p> <p>File: contracts/rewards-booster/oracles/OracleUniswapV3.sol          SHA3: d69fb03d156dd24279ec7a253f7355715b0fc5e977cad9c938915865abc44f921</p> <p>File:          contracts/rewards-booster/oracles/OracleBalancerWeighted.sol          SHA3: d684c8c7d817f7c688701f0be9e490700b63a554b0d910156b2596b35f63b3d1</p> <p>File: contracts/rewards-booster/oracles/ASXPriceFeed.sol          SHA3: 77dd8de36be60c03f83b886d49fda6b705a51fc53970040e5bca289db72d12d0</p>
---

### Third review scope

<b>Repository</b>	<a href="https://bitbucket.ideasoft.io/projects/PBON/repos/solidity/commits/876a5a463c42c43613b49bcff313a0bceec68fe5e">https://bitbucket.ideasoft.io/projects/PBON/repos/solidity/commits/876a5a463c42c43613b49bcff313a0bceec68fe5e</a>
<b>Commit</b>	6a79c024e1c2a2b21fa8fc07b2960e1aadfa97cc
<b>Whitepaper</b>	N/A
<b>Requirements</b>	N/A
<b>Technical Requirements</b>	N/A
<b>Contracts</b>	File: contracts/rewards-booster/RewardsBooster.sol SHA3: 16ea0b9e2766f42deb06ecf77802bed49efda62e39314d91af4bdd1bc8491934

<p>File: contracts/vesting/UniswapWrapper.sol SHA3 ee8786bc2172a199a579c754a0e832e846bc6203f481c23a7c0289f9b62ba96d</p> <p>File: contracts/core/prize-pool/PrizePoolV2.sol SHA3:c4382e3f08d4d4090578e87266c4f78faa603b2e41cf3ca8543b20610d354d2f</p> <p>File: contracts/core/prize-pool/StakePrizePoolV2.sol SHA3 45d6096296319ce9fd0b41ad4c68329b56a253acc1b6509b42fc4dfe4d0bff4d</p> <p>File: contracts/vesting/ESASXVesting.sol SHA3:42f7fe32bd2e9c97d6fc097413f76c9bc5c2cf7f1a6ddebe2a9e76c7520e1252</p> <p>File: contracts/rewards-boosters/valuers/ValuerUniswapV3.sol SHA3 09e1c9ea8c903491e2b603af7c394cbee0d1ec4c7e925b2f920be6cbd8d95378</p> <p>File: contracts/rewards-boosters/valuers/ValuerBalancerWeighted.sol SHA3:cf6e9bb6b180ac6dea7f7c6dc3a48834dc9d9d1e52fbfbf09c45b6413591e3b8</p> <p>File: contracts/rewards-boosters/validators/ValidatorUniswapV3.sol SHA3:4db1290338560c8de827b36a9bc7c5a41321b04642b69dae4d70e42296422b79</p> <p>File: contracts/rewards-boosters/validators/ValidatorBalancerWeighted.sol SHA3:acb797565a49ece95b70e82b9502359a60e5d08221f8169bb9f6b6b30998bf31</p> <p>File: contracts/rewards-boosters/oracles/OracleUniswapV3.sol SHA3:d6e3ac6be20f12ca073ca7817dcca701c4c83707cc26409abab1e354ac86a7df</p> <p>File: contracts/rewards-boosters/oracles/OracleBalancerWeighted.sol SHA3:d684c8c7d817f7c688701f0be9e490700b63a554b0d910156b2596b35f63b3d1</p> <p>File: contracts/rewards-boosters/oracles/ASXPriceFeed.sol SHA3:77dd8de36be60c03f83b886d49fda6b705a51fc53970040e5bca289db72d12d0</p>
--