// HALBORN

Razor Network

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **April 11th, 2022 - June 12th, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	04/19/2022	István Böhm
0.2	Document Updates	05/16/2022	István Böhm
0.3	Draft Updates	06/12/2022	Gokberk Gulgun
0.4	Draft Review	06/13/2022	Gabi Urrutia
1.0	Remediation Plan	07/21/2022	Gokberk Gulgun
1.1	Remediation Plan Review	07/22/2022	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
István Böhm	Halborn	Istvan.Bohm@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Razor Network engaged Halborn to conduct a security audit on their smart contracts beginning on April 11th, 2022 and ending on June 12th, 2022. The security assessment was scoped to the smart contracts provided in the contracts GitHub repository razor-network/contracts.

1.2 AUDIT SUMMARY

The team at Halborn was provided eight weeks for the engagement and assigned two full-time security engineers to audit the security of the smart contract. The security engineers are blockchain and smart-contract security experts with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The purpose of this audit is to:

- Ensure that smart contract functions operate as intended.
- Identify potential security issues with the smart contracts.

In summary, Halborn identified few security risks that were mostly addressed by the Razor Network 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 code and can quickly identify items that do not follow the security best practices. The following phases and associated tools were used during 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:

- Core/parameters/interfaces/IRandomNoManagerParams.sol
- randomNumber/IRandomNoProvider.sol
- randomNumber/RandomNoStorage.sol
- tokenization/IStakedTokenFactory.sol
- Core/parameters/ACL.sol
- tokenization/StakedTokenFactory.sol
- Pause.sol
- Core/parameters/child/RandomNoManagerParams.sol
- Core/parameters/interfaces/ICollectionManagerParams.sol
- lib/Random.sol
- tokenization/RAZOR.sol
- mocks/InitializableMock.sol
- Core/storage/BlockStorage.sol
- Core/parameters/interfaces/IVoteManagerParams.sol
- Core/parameters/child/CollectionManagerParams.sol
- mocks/MerklePosAwareTest.sol
- Core/parameters/interfaces/IBlockManagerParams.sol
- randomNumber/IRandomNoClient.sol
- Core/interface/IBlockManager.sol
- Core/interface/IRewardManager.sol
- Core/parameters/child/VoteManagerParams.sol
- Core/storage/VoteStorage.sol
- IDelegator.sol
- Core/parameters/child/BlockManagerParams.sol
- Core/storage/CollectionStorage.sol
- Delegator.sol
- tokenization/IStakedToken.sol
- Core/parameters/interfaces/IRewardManagerParams.sol
- Initializable.sol
- Core/parameters/child/RewardManagerParams.sol
- Core/StateManager.sol
- lib/MerklePosAware.sol

- randomNumber/RandomNoManager.sol
- Core/storage/Constants.sol
- lib/Structs.sol
- Core/interface/ICollectionManager.sol
- Core/interface/IVoteManager.sol
- tokenization/StakedToken.sol
- Core/parameters/interfaces/IStakeManagerParams.sol
- Core/storage/StakeStorage.sol
- Core/interface/IStakeManager.sol
- Core/parameters/child/StakeManagerParams.sol
- Core/RewardManager.sol
- Core/parameters/Governance.sol
- Core/VoteManager.sol
- Core/CollectionManager.sol
- Core/BlockManager.sol
- Core/StakeManager.sol

Commit ID: 90070b129551e43cb454d60b04629d8f5e1fe096

FIX Commit TREE/ID :

Commit ID: : 6c4ba25d4223703fe179561fd4cbba0bbcbe8cdb.

Code Location : Commit ID

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	3	5	13

EXECUTIVE OVERVIEW

LIKELIHOOD

IMPACT

	(HAL-03)			
(HAL-04) (HAL-05) (HAL-06) (HAL-07) (HAL-08)		(HAL-01) (HAL-02)		
(HAL-09) (HAL-10) (HAL-11) (HAL-12) (HAL-13) (HAL-14) (HAL-15) (HAL-16) (HAL-17) (HAL-18) (HAL-19) (HAL-20) (HAL-21)				

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL01 - ANYONE CAN UNSTAKE OTHER STAKER IF HOLDS ENOUGH AMOUNT	Medium	SOLVED - 07/21/2022
HAL02 - MISSING INITIALIZED MODIFIER ON THE GIVEBLOCKREWARD FUNCTION	Medium	SOLVED - 07/21/2022
HAL03 - ROLES ARE NOT ASSIGNED IN THE INITIALIZERS	Medium	RISK ACCEPTED
HAL04 - REWARD MANAGER DOES NOT CHECK STAKER STATUS	Low	SOLVED - 07/21/2022
HAL05 - UNSTAKE FUNCTION DOES NOT CHECK WITHDRAW LOCK	Low	SOLVED - 07/21/2022
HAL06 - ESCAPE HATCH STATUS DEFAULT IS ENABLED AS TRUE	Low	RISK ACCEPTED
HAL07 - EXISTENCE OF JOB IDS IS NOT CHECKED	Low	SOLVED - 07/21/2022
HAL08 - MISSING REQUIRE STATEMENT IF THE STAKER IS ALREADY SLASHED	Low	SOLVED - 07/21/2022
HAL09 - ADD CONSTRUCTOR INITIALIZERS	Informational	ACKNOWLEDGED
HAL10 - EVENTS ARE NOT INDEXED	Informational	SOLVED - 07/21/2022
HAL11 - REVERT STRING SIZE OPTIMIZATION	Informational	SOLVED - 07/21/2022
HAL12 - USE SAFETRANSFER SAFETRANSFERFROM CONSISTENTLY INSTEAD OF TRANSFER TRANSFERFROM	Informational	ACKNOWLEDGED
HAL13 - MISTAKENLY SENT ERC20 TOKENS CAN NOT BE RESCUED IN THE CONTRACTS	Informational	ACKNOWLEDGED
HAL14 - UPGRADE PRAGMA TO AT LEAST 0.8.4	Informational	SOLVED - 07/21/2022
HAL15 - GOVERNANCE PARAMETERS DO NOT HAVE ANY UPPER/LOWER BOUND	Informational	ACKNOWLEDGED
HAL16 - BURN ADDRESS IS DEFINED AS EVM NATIVE TOKEN ADDRESS	Informational	SOLVED - 07/21/2022

HAL17 - USING POSTFIX OPERATORS IN LOOPS	Informational	ACKNOWLEDGED
HAL18 - UNNEEDED INITIALIZATION OF UNSIGNED INTEGER VARIABLES TO 0	Informational	ACKNOWLEDGED
HAL19 - ARRAY.LENGTH USED IN LOOP CONDITIONS	Informational	ACKNOWLEDGED
HAL20 - USING != 0 CONSUMES LESS GAS THAN > 0 IN UNSIGNED INTEGER VALIDATION	Informational	ACKNOWLEDGED

FINDINGS & TECH DETAILS

3.1 (HAL-01) ANYONE CAN UNSTAKE OTHER STAKER IF HOLDS ENOUGH AMOUNT - MEDIUM

Description:

Any staker can call unstake with function with another staker ID. Even if the staker must have enough to start the **unstake** operation, the staker's address should be compared to **msg.sender**.

Code Location:

Reference

Listing 1

```
function unstake(uint32 stakerId, uint256 sAmount) external initialized whenNotPaused {
    require(sAmount > 0, "Non-Positive Amount");
    require(stakerId != 0, "staker.id = 0");
    require(stakers[stakerId].stake > 0, "Nonpositive stake");
    // slither-disable-next-line timestamp
    require(locks[msg.sender][stakers[stakerId].tokenAddress][LockType.Unstake].amount == 0, "Existing Unstake Lock");
    uint32 epoch = getEpoch();
    Structs.Staker storage staker = stakers[stakerId];
    IStakedToken sToken = IStakedToken(staker.tokenAddress);

    require(sToken.balanceOf(msg.sender) >= sAmount, "Invalid Amount");

    locks[msg.sender][staker.tokenAddress][LockType.Unstake] = Structs.Lock(sAmount, epoch + unstakeLockPeriod);
    emit Unstaked(msg.sender, epoch, stakerId, sAmount, staker.stake, block.timestamp);
    // Ignoring below line for testing as this is standard erc20 function
    require(sToken.transferFrom(msg.sender, address(this), sAmount), "sToken transfer failed");
}
```

Recommendation:

In the stake manager, consider to check **stakers[stakerId]._address == msg.sender**.

Remediation Plan:

SOLVED: The Razor Team states that the **sToken** will be minted to a specific account. Using the **transferFrom** function, it is impossible to unstake another account.

3.2 (HAL-02) MISSING INITIALIZED MODIFIER ON THE GIVEBLOCKREWARD FUNCTION - MEDIUM

Description:

claimBlockReward() designed to be called by the selected **staker** whose proposed block has the lowest iteration. If the necessary constraints are met, **claimBlockReward()**, will call rewardManager to give a block reward to the staker. However, in the **RewardManager** contract, the **giveBlockReward** function does not have the **initialized** modifier. If **REWARD_MODIFIER_-ROLE** is not assigned to the **BlockManager** contract, the **giveBlockReward** function will revert.

Code Location:

Reference

Listing 2

```
    function claimBlockReward() external initialized checkState(State.Confirm, buffer) {
        uint32 epoch = getEpoch();
        uint32 stakerId = stakeManager.getStakerId(msg.sender);
        require(stakerId > 0, "Structs.Staker does not exist");
        // slither-disable-next-line timestamp
        require(blocks[epoch].proposerId == 0, "Block already confirmed");

        if (sortedProposedBlockIds[epoch].length != 0 && blockIndexToBeConfirmed != -1) {
            uint32 proposerId = proposedBlocks[epoch][sortedProposedBlockIds[epoch][uint8(blockIndexToBeConfirmed)]].proposerId;
            require(proposerId == stakerId, "Block Proposer mismatches");
            _confirmBlock(epoch, proposerId);
        }
```

```
        uint32 updateRegistryEpoch = collectionManager.getUpdateRegistryEpoch();
        // slither-disable-next-line incorrect-equality, timestamp
        if (updateRegistryEpoch <= epoch) {
            collectionManager.updateDelayedRegistry();
        }
    }
```

Recommendation:

Consider adding the **initialized** modifier to functions.

Remediation Plan:

SOLVED: The Razor Team fixed the issue in the following commit.

3.3 (HAL-03) ROLES ARE NOT ASSIGNED IN THE INITIALIZERS - MEDIUM

Description:

No role is assigned to the initializers. All roles are managed by deployment scripts. For instance, in the **CollectionManager** contract, many functions are managed by the **COLLECTION_MODIFIER** role. However, the contract role is not assigned in the initializer. As another example, **REWARD_MODIFIER_ROLE** is not reassigned in initializers. For that reason, during the deployment scripts, if the contract role is not assigned to the contract, the functionality of the contract will break.

Code Location:

Reference

Listing 3

```
function initialize(address voteManagerAddress, address blockManagerAddress) external initializer onlyRole(DEFAULT_ADMIN_ROLE) {
    voteManager = IVoteManager(voteManagerAddress);
    blockManager = IBlockManager(blockManagerAddress);
}
```

Recommendation:

Initialize all contract roles in the initializer or double-check the deployment scripts to prevent unintended behavior.

Remediation Plan:

RISK ACCEPTED: The Razor Team accepted the risk of this finding. The Razor Team claims that the deployment script will make sure that the contracts are not broken.

3.4 (HAL-04) REWARD MANAGER DOES NOT CHECK STAKER STATUS - LOW

Description:

The block reward is distributed to the staker during the **giveBlockReward** function. However, no check is implemented whether the staker is active or not. If the staker is not active, they can still claim the block reward.

Code Location:

Reference

Listing 4

```
    function giveBlockReward(uint32 stakerId, uint32 epoch) external override onlyRole(REWARD_MODIFIER_ROLE) {
        Structs.Staker memory staker = stakeManager.getStaker(stakerId);
        if (!staker.acceptDelegation) {
            stakeManager.setStakerStake(epoch, stakerId, StakeChanged.BlockReward, staker.stake, staker.stake + blockReward);
            return;
        }
        IStakedToken sToken = IStakedToken(staker.tokenAddress);
        uint256 totalSupply = sToken.totalSupply();
        uint256 stakerSRZR = sToken.balanceOf(staker._address);
        uint256 delegatorShare = blockReward - ((blockReward * stakerSRZR) / totalSupply);
        uint8 commissionApplicable = staker.commission < maxCommission ? staker.commission : maxCommission;
        uint256 stakerReward = (delegatorShare * commissionApplicable) / 100;
        stakeManager.setStakerStake(epoch, stakerId, StakeChanged.BlockReward, staker.stake, staker.stake + (blockReward - stakerReward));
        stakeManager.setStakerReward(
            epoch,
```

```
            stakerId,
            StakerRewardChanged.StakerRewardAdded,
            staker.stakerReward,
            staker.stakerReward + stakerReward
        );
    }

```

Recommendation:

In the rewards' manager, consider checking the **_isStakerActive** statement.

Remediation Plan:

SOLVED: A staker can only claim rewards if they have been active the previous states. When inactive stakers start voting in the commit state, they also receive inactivity penalties before registering their commit.

3.5 (HAL-05) UNSTAKE FUNCTION DOES NOT CHECK WITHDRAW LOCK - LOW

Description:

In the unstake function, it has been observed that the current withdrawal locks are not checked.

Code Location:

Reference

Listing 5

```
function unstake(uint32 stakerId, uint256 sAmount) external initialized whenNotPaused {
    require(sAmount > 0, "Non-Positive Amount");
    require(stakerId != 0, "staker.id = 0");
    require(stakers[stakerId].stake > 0, "Nonpositive stake");
    // slither-disable-next-line timestamp
    require(locks[msg.sender][stakers[stakerId].tokenAddress][LockType.Unstake].amount == 0, "Existing Unstake Lock");
    uint32 epoch = getEpoch();
    Structs.Staker storage staker = stakers[stakerId];
    IStakedToken sToken = IStakedToken(staker.tokenAddress);

    require(sToken.balanceOf(msg.sender) >= sAmount, "Invalid Amount");

    locks[msg.sender][staker.tokenAddress][LockType.Unstake] = Structs.Lock(sAmount, epoch + unstakeLockPeriod);
    emit Unstaked(msg.sender, epoch, stakerId, sAmount, staker.stake, block.timestamp);
    // Ignoring below line for testing as this is standard erc20 function
    require(sToken.transferFrom(msg.sender, address(this), sAmount), "sToken transfer failed");
}
```

Recommendation:

At the stake manager, consider checking:

Listing 6

```
require(locks[msg.sender][stakers[stakerId].tokenAddress][LockType
↳ .Withdraw].unlockAfter == 0, "Withdraw Lock exists");
```

Remediation Plan:

SOLVED: The Razor Team fixed the issue in the following commit.

3.6 (HAL-06) ESCAPE HATCH STATUS DEFAULT IS ENABLED AS TRUE - LOW

Description:

escapeHatchEnabled defaults to true, the default admin role can remove all funds in an emergency. Only the governor can set this variable to false.

Instead of setting the value to true, during the emergency, the governor should change it to true/false.

Code Location:

Reference

Recommendation:

It is recommended to set the initial state of **escapeHatchEnabled** to false, **escapeHatchEnabled** should be set to true/false from the governance mechanism.

- Timelock with reasonable latency, e.g. 48 hours, for knowledge of privileged operations;
- Governance must enable hatching progress with the function. The parameter should not be enabled by default.

Remediation Plan:

RISK ACCEPTED: The Razor Team accepted the risk of this finding.

3.7 (HAL-07) EXISTENCE OF JOB IDS IS NOT CHECKED - LOW

Description:

The **createCollection** function creates a collection on the network. The **jobIDs** parameter is an array containing which jobs stakers should query to report on the collection. However, the existence of **jobIDS** is not checked when the collection is created.

Code Location:

Reference

Listing 7

```
    function createCollection(
        uint32 tolerance,
        int8 power,
        uint32 aggregationMethod,
        uint16[] memory jobIDs,
        string calldata name
    ) external onlyRole(COLLECTION_MODIFIER_ROLE) checkState(State.Confirm, buffer) {
        require(jobIDs.length > 0, "no jobs added");
        require(tolerance <= maxTolerance, "Invalid tolerance value");

        uint32 epoch = getEpoch();

        // slither-disable-next-line incorrect-equality,timestamp
        if (updateRegistryEpoch <= epoch) {
            _updateDelayedRegistry();
        }

        numCollections = numCollections + 1;

        collections[numCollections] = Structs.Collection(true, numCollections, power, tolerance, aggregationMethod, jobIDs, name);
```

```

        numActiveCollections = numActiveCollections + 1;

        updateRegistryEpoch = epoch + 1;
        _updateRegistry();

        emit CollectionCreated(numCollections, block.timestamp);

        _setIDName(name, numCollections);
        voteManager.storeDepth(_getDepth()); // TODO : Create method called as createCollectionBatch and update storeDepth only once
    }
```

Recommendation:

Make sure the collections are created with the existing **jobIDS**.

Remediation Plan:

SOLVED: The Razor Team fixed the issue in the following commit.

3.8 (HAL-08) MISSING REQUIRE STATEMENT IF THE STAKER IS ALREADY SLASHED - LOW

Description:

In the protocol, if a bounty hunter reveals a secret in the commit state, they can claim the bounty from the system. When the **staker** is correctly ratted out, their stake is slashed and the bounty hunter receives a reward. However, the codebase is missing the check if the staker is already slashed.

Code Location:

Reference

Listing 8

```
    function slash(
        uint32 epoch,
        uint32 stakerId,
        address bountyHunter
    ) external override onlyRole(STAKE_MODIFIER_ROLE) {
        uint256 _stake = stakers[stakerId].stake;

        uint256 bounty;
        uint256 amountToBeBurned;
        uint256 amountToBeKept;

        // Block Scoping
        // Done for stack too deep issue
        // https://soliditydeveloper.com/stacktoodeep
        {
            (uint32 bountyNum, uint32 burnSlashNum, uint32 keepSlashNum) = (slashNums.bounty, slashNums.burn, slashNums.keep);
            bounty = (_stake * bountyNum) / BASE_DENOMINATOR;
            amountToBeBurned = (_stake * burnSlashNum) / BASE_DENOMINATOR;
```

```
            amountToBeKept = (_stake * keepSlashNum) / BASE_DENOMINATOR;
        }

        uint256 slashPenaltyAmount = bounty + amountToBeBurned + amountToBeKept;
        _stake = _stake - slashPenaltyAmount;
        stakers[stakerId].isSlashed = true;
        _setStakerStake(epoch, stakerId, StakeChanged.Slashed, _stake + slashPenaltyAmount, _stake);

        if (bounty == 0) return;
        bountyCounter = bountyCounter + 1;
        bountyLocks[bountyCounter] = Structs.BountyLock(epoch + withdrawLockPeriod, bountyHunter, bounty);

        emit Slashed(bountyCounter, bountyHunter);
        //please note that since slashing is a critical part of consensus algorithm,
        //the following transfers are not required, even if the transfers fail, the slashing
        //tx should complete.
        // Ignoring below line for testing as this is standard erc20 function
        require(razor.transfer(BURN_ADDRESS, amountToBeBurned), "couldn't burn");
    }
```

Recommendation:

Consider adding the following check at the beginning of the function.

Listing 9

```
require(!stakers[stakerId].isSlashed,"Already slashed");
```

Remediation Plan:

SOLVED: Once a staker is slashed, they can no longer participate in the Razor Network. Hence, they cannot be slashed again.

3.9 (HAL-09) ADD CONSTRUCTOR INITIALIZERS - INFORMATIONAL

Description:

According to OpenZeppelin recommendation, the guidelines now are to make it impossible for someone to initialize on an implementation contract, by adding an empty constructor with the initializer modifier. From that reason, the implementation contract is automatically initialized at deployment time.

The implementation can be viewed from the Openzeppelin Wizard by selecting Transparent or UUPS in the upgradeability section.

Front-running can be prevented by adding the necessary arguments defined by OpenZeppelin.

Code Location:

Reference

Recommendation:

Implement the following constructor in the upgradable contracts.

Listing 10

```
    constructor() {
        _disableInitializers();
    }
```

Remediation Plan:

ACKNOWLEDGED: The Razor Team acknowledged this issue.

3.10 (HAL-10) ACCEPT DELEGATE IS NOT SET IN THE STAKE FUNCTION - INFORMATIONAL

Description:

Accept delegate is not set during the stake function. If the staker wants to enable delegation, they must call the setDelegationAcception function.

Code Location:

Reference

Listing 11

```
    function setDelegationAcceptance(bool status) external {
        uint32 stakerId = stakerIds[msg.sender];
        require(stakerId != 0, "staker id = 0");
        require(stakers[stakerId].commission != 0, "comission not set");
        stakers[stakerId].acceptDelegation = status;
        emit DelegationAcceptanceChanged(status, msg.sender, stakerId);
    }
```

Recommendation:

The variable can be set with the stake function as an argument.

Remediation Plan:

ACKNOWLEDGED: The Razor Team acknowledged this issue.

3.11 (HAL-11) EVENTS ARE NOT INDEXED - INFORMATIONAL

Description:

Emitted events are not indexed, making it difficult for off-chain scripts, such as dApp front-ends, to filter events efficiently.

Recommendation:

Add the indexed keyword in each event.

Remediation Plan:

SOLVED: The Razor Team fixed the issue in the following commit.

3.12 (HAL-12) REVERT STRING SIZE OPTIMIZATION - INFORMATIONAL

Description:

Shortening the revert strings to fit within 32 bytes will decrease deployment time gas and reduce runtime gas when the revert condition is met.

Revert strings that are longer than 32 bytes require at least one additional mstore, along with additional overhead to calculate memory offset, etc.

Recommendation:

Shorten the revert strings to fit within 32 bytes. That will affect gas optimization.

Remediation Plan:

SOLVED: The Razor Team does not have any revert strings longer than 32 bytes in the entire codebase.

3.13 (HAL-13) USE SAFETRANSFER SAFETRANSFERFROM CONSISTENTLY INSTEAD OF TRANSFER TRANSFERFROM - INFORMATIONAL

Description:

Contracts use the require statement instead of OpenZeppelin's **safeTransfer/safeTransferFrom**, unless one is sure that the given token reverts on failure. Failure to do so will result in silent transfer failures and affect token accounting in the contract.

Recommendation:

Consider using **safeTransfer/safeTransferFrom** instead of **transfer/transferFrom**.

Remediation Plan:

ACKNOWLEDGED: The Razor Team acknowledged this issue.

3.14 (HAL-14) MISTAKENLY SENT ERC20 TOKENS CAN NOT BE RESCUED IN THE CONTRACTS - INFORMATIONAL

Description:

Contracts are missing accidental sweep/rescue ERC-20 transfers. Accidentally, sent ERC20s will be locked in contracts.

Recommendation:

Consider adding a function to sweep accidental ERC-20 transfers to contracts.

Remediation Plan:

ACKNOWLEDGED: The Razor Team acknowledged this issue.

3.15 (HAL-15) UPGRADE PRAGMA TO AT LEAST 0.8.4 - INFORMATIONAL

Description:

Using newer compiler versions and the optimizer gives gas optimizations and additional safety checks are available free.

The advantages of versions 0.8.* over <0.8.0 are:

- Safemath by default from 0.8.0 (can be more gas efficient than library based safemath.)
- Low level inliner : from 0.8.2, leads to cheaper runtime gas. Especially relevant when the contract has small functions. For example, OpenZeppelin libraries typically have a lot of small helper functions and if they are not inlined, they cost an additional 20 to 40 gas because of 2 extra jump instructions and additional stack operations needed for function calls.
- Optimizer improvements in packed structs: Before 0.8.3, storing packed structs, in some cases, used an additional storage read operation. After EIP-2929, if the slot was already cold, this means unnecessary stack operations and extra deploy time costs. However, if the slot was already warm, this means additional cost of 100 gas alongside the same unnecessary stack operations and extra deploy time costs.
- Custom errors from 0.8.4, leads to cheaper deploy time cost and run time cost. Note: the run time cost is only relevant when the revert condition is met. In short, replace revert strings by custom errors.

Recommendation:

Consider upgrading pragma to at least 0.8.4.

Remediation Plan:

SOLVED: The Razor Team now compiles to 0.8.4 on hardhat.

3.16 (HAL-16) GOVERNANCE PARAMETERS DO NOT HAVE ANY UPPER/LOWER BOUND - INFORMATIONAL

Description:

In the governance parameters, no upper/lower bound has been defined. Even if parameters are controlled by the governance, the values may be limited by min/max values. As an example, an attacker took advantage of a malicious Proposal Hack before. For instance, minSafeRazor parameter has been used in **StakeManager**. The **stake()** function can be called with the zero amount.

Code Location:

Functions

Listing 12

```
    function setMinSafeRazor(uint256 _minSafeRazor) external initialized onlyRole(GOVERNER_ROLE) {
        emit ParameterChanged(msg.sender, "minSafeRazor", _minSafeRazor, block.timestamp);
        stakeManagerParams.setMinSafeRazor(_minSafeRazor);
    }
```

Recommendation:

It is recommended to define an upper/lower limit in the governance parameters.

Remediation Plan:

ACKNOWLEDGED: The Razor Team acknowledged this issue.

3.17 (HAL-17) BURN ADDRESS IS DEFINED AS EVM NATIVE TOKEN ADDRESS - INFORMATIONAL

Description:

0xEeeeeEeeeEeEeeEeEeEeeEEEeeeeEeeeeeeeEEeE is an address used to identify the ETH coin as an ERC20 token. Many protocols use ERC20 tokens so when it comes to the main currency in Ethereum, ETH itself, they have to create some kind of representation to use it as a token. However, in the Razor Network contracts, it has been defined as a **BURN** address.

Code Location:

Constants

Listing 13

```
BURN_ADDRESS = 0xEeeeeEeeeEeEeeEeEeEeeEEEeeeeEeeeeeeeEEeE;
```

Recommendation:

Consider changing **BURN_ADDRESS** with **0x000000000000000000000000000000000000dEaD**.

Remediation Plan:

SOLVED: The Razor Team fixed the issue in the following commit.

3.18 (HAL-18) USING POSTFIX OPERATORS IN LOOPS - INFORMATIONAL

Description:

In the loops below, postfix (e.g. i++) operators are used to increment or decrement the values of variables. It is known that, in loops, using prefix operators (e.g. ++i) costs less gas per iteration than using postfix operators.

Code Location:

Core/BlockManager.sol
- Line 148 for (uint32 i = 0; i < sortedValues.length; i++){
- Line 418 for (uint256 i = 0; i < _block.ids.length; i++){
- Line 457 for (uint8 i = 0; i < sortedProposedBlockslength; i++){
- Line 464 for (uint8 i = 0; i < sortedProposedBlockslength; i++){
- Line 471 for (uint256 j = sortedProposedBlockslength - 1; j > i; j--){
- Line 510 for (uint8 i = blockIndex + 1; i < sortedProposedBlocksLength ; i++){

Core/CollectionManager.sol
- Line 345 for (uint16 i = 1; i <= numCollections; i++){
- Line 364 for (uint16 i = 1; i <= numCollections; i++){
- Line 377 for (uint16 i = 1; i <= numCollections; i++){

Core/RewardManager.sol
- Line 133 for (uint16 i = 0; i < idsRevealedLastEpoch.length; i++){

Core/VoteManager.sol
- Line 139 for (uint16 i = 0; i < tree.values.length; i++){

lib/MerklePosAware.sol
- Line 17 for (uint256 i = 0; i < proofs.length; i++){
- Line 46 j--;
- Line 58 i++;

- Line 69 `for (uint8 i = 0; i < depth; i++){`

Proof of Concept:

For example, based on the following test contract:

Listing 14: Test.sol

```
//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 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` and `--j` instead of `i++` and `j--` to increment or decrement the values of `uint` variables within loops. This does not just apply to iterator variables. It also applies to increments and decrements done within the loop code block.

Remediation Plan:

ACKNOWLEDGED: The Razor Team acknowledged this issue.

3.19 (HAL-19) UNNEEDED INITIALIZATION OF UNSIGNED INTEGER VARIABLES TO 0 - INFORMATIONAL

Description:

Since the following variables are unsigned integers, they are already initialized to 0. Reassigning the same value to the variables wastes gas.

Code Location:

Core/BlockManager.sol

- Line 148 `for (uint32 i = 0; i < sortedValues.length; i++){`
- Line 291 `uint256 lower = 0;`
- Line 418 `for (uint256 i = 0; i < _block.ids.length; i++){`
- Line 457 `for (uint8 i = 0; i < sortedProposedBlockslength; i++){`
- Line 464 `for (uint8 i = 0; i < sortedProposedBlockslength; i++){`

Core/CollectionManager.sol

- Line 344 `uint16 j = 0;`
- Line 363 `uint16 j = 0;`
- Line 376 `uint16 j = 0;`
- Line 451 `for (n = 0; x > 1; x >>= 1){`

Core/RewardManager.sol

- Line 132 `uint64 penalty = 0;`
- Line 133 `for (uint16 i = 0; i < idsRevealedLastEpoch.length; i++){`

Core/VoteManager.sol

- Line 139 `for (uint16 i = 0; i < tree.values.length; i++){`

lib/MerklePosAware.sol

- Line 17 `for (uint256 i = 0; i < proofs.length; i++){`
- Line 43 `uint256 i = 0;`
- Line 69 `for (uint8 i = 0; i < depth; i++){`

Core/StakeManager.sol
- Line 230 `uint256 totalSupply = 0;`

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended not to initialize unsigned integer variables to 0 to save some gas. For example, use instead:
`for (uint8 i; i < sortedProposedBlockslength; i++){`.

Remediation Plan:

ACKNOWLEDGED: The Razor Team acknowledged this issue.

3.20 (HAL-20) ARRAY.LENGTH USED IN LOOP CONDITIONS - INFORMATIONAL

Description:

In the loops below, unnecessary reading the lengths of arrays at each iteration wastes gas.

Code Location:

Core/BlockManager.sol
- Line 148 `for (uint32 i = 0; i < sortedValues.length; i++){`
- Line 418 `for (uint256 i = 0; i < _block.ids.length; i++){`

Core/RewardManager.sol
- Line 133 `for (uint16 i = 0; i < idsRevealedLastEpoch.length; i++){`

Core/VoteManager.sol
- Line 139 `for (uint16 i = 0; i < tree.values.length; i++){`

lib/MerklePosAware.sol
- Line 17 `for (uint256 i = 0; i < proofs.length; i++){`

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to cache array lengths outside of loops as long as the size does not change during the loop:
`uint sortedValuesLength = sortedValues.length; for (uint32 i; i < sortedValuesLength; ++i){ ... }`

Remediation Plan:

ACKNOWLEDGED: The Razor Team acknowledged this issue.

3.21 (HAL-21) USING != 0 CONSUMES LESS GAS THAN > 0 IN UNSIGNED INTEGER VALIDATION - INFORMATIONAL

Description:

In the require statements below, > 0 was used to check whether unsigned integer parameters are greater than 0. Using != 0 is known to costs less gas than > 0.

Code Location:

Core/BlockManager.sol
- Line 180 require(stakerId > 0, "Structs.Staker does not exist");

Core/CollectionManager.sol
- Line 196 require(jobIDs.length > 0, "no jobs added");

Core/StakeManager.sol
- Line 318 require(stakers[stakerId].stake > 0, "Nonpositive stake");

Core/VoteManager.sol
- Line 84 require(stakerId > 0, "Staker does not exist");
- Line 121 require(stakerId > 0, "Staker does not exist");
- Line 195 require(thisStakerId > 0, "Staker does not exist");

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to use != 0 instead of > 0 to validate unsigned integer parameters. For example:

```
require(stakerId != 0, "Staker does not exist");
```

Remediation Plan:

ACKNOWLEDGED: The Razor Team acknowledged this issue.

AUTOMATED TESTING

4.1 STATIC ANALYSIS REPORT

Description:

Halborn used automated testing techniques to enhance the coverage of certain areas of the smart contracts in scope. Among the tools used was Slither, a Solidity static analysis framework. After Halborn verified the smart contracts in the repository and was able to compile them correctly into their abis and binary format, Slither was run against the 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:

StateManager.sol

```
StateManager._getEpoch() (contracts/Core/StateManager.sol#53-55) is never used and should be removed
StateManager._getState(uint8) (contracts/Core/StateManager.sol#57-68) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
```

- No major issues found by Slither.

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 the smart contracts and sent the compiled results to the analyzers in order to locate any vulnerabilities.

MythX results:

Report for contracts/Core/parameters/ACL.sol
https://dashboard.mythx.io/#/console/analyses/f02fdc37-1a7c-46ca-a413-c97614d0e945

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

Report for contracts/Core/storage/Constants.sol
https://dashboard.mythx.io/#/console/analyses/b2f0658d-9be6-47c6-9d7c-f0722ef3c506

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

Report for contracts/Delegator.sol
https://dashboard.mythx.io/#/console/analyses/6f37bb56-2deb-4c76-9e52-304cd813709e

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.
25	(SWC-123) Requirement Violation	Low	Requirement violation.
30	(SWC-123) Requirement Violation	Low	Requirement violation.
40	(SWC-123) Requirement Violation	Low	Requirement violation.

Report for contracts/Pause.sol
https://dashboard.mythx.io/#/console/analyses/dcadc59d-85db-4391-a42b-a134f239cfba

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

Report for contracts/Core/BlockManager.sol
https://dashboard.mythx.io/#/console/analyses/cd0586b3-1aca-4e09-8782-10b877a74e93

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

Report for contracts/lib/Random.sol
https://dashboard.mythx.io/#/console/analyses/1f27cbe4-0758-41c7-86a4-35a2576df22f

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

Report for contracts/lib/Structs.sol
https://dashboard.mythx.io/#/console/analyses/683abd13-4f66-4a97-b1cf-a1f176cb96bc

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

Report for contracts/Core/StakeManager.sol
https://dashboard.mythx.io/#/console/analyses/506f0b7c-941e-4c00-8a89-23b478d571d3

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

Report for contracts/Core/parameters/child/StakeManagerParams.sol
https://dashboard.mythx.io/#/console/analyses/2be588c4-c981-47dc-ac5f-1fe968c74198

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

Report for contracts/Core/RewardManager.sol
https://dashboard.mythx.io/#/console/analyses/df2d5205-9e8e-4905-82a0-e78b14c12b58

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

Report for StateManager.sol
https://dashboard.mythx.io/#/console/analyses/0a5d1e19-310e-4ca4-a66d-4e9dc97f8b7b

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

Report for contracts/Core/CollectionManager.sol
https://dashboard.mythx.io/#/console/analyses/dfd43e56-9b58-4118-9bfb-852234856c12

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

Report for contracts/Core/parameters/child/CollectionManagerParams.sol
https://dashboard.mythx.io/#/console/analyses/0617d94e-726d-4f34-aa09-f35fbc0399a4

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

Report for contracts/Core/VoteManager.sol
https://dashboard.mythx.io/#/console/analyses/ca352b79-23be-4584-9200-55eb3d2407cc

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

Report for contracts/Core/parameters/child/VoteManagerParams.sol
https://dashboard.mythx.io/#/console/analyses/4cbb893a-f994-4228-88c4-48430e1f3d82

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

Report for contracts/lib/MerklePosAware.sol
https://dashboard.mythx.io/#/console/analyses/9c44ed4a-adad-4aa4-b90a-0b75c2ca94f1

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

Report for contracts/Core/parameters/Governance.sol
https://dashboard.mythx.io/#/console/analyses/54e3c77d-34c9-426b-9da9-076f550d0f23

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

Report for contracts/Core/parameters/child/RandomNoManagerParams.sol
https://dashboard.mythx.io/#/console/analyses/9e979570-1e9a-4a54-bc6f-1a70caad39aa

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

Report for Core/storage/CollectionStorage.sol
https://dashboard.mythx.io/#/console/analyses/59b95c01-1bc5-4471-89f2-b5c4d61a2bf2

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

Report for Core/storage/VoteStorage.sol
https://dashboard.mythx.io/#/console/analyses/edbf00de-8440-45e3-a232-c21b8ce72813

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

Report for Core/storage/BlockStorage.sol
https://dashboard.mythx.io/#/console/analyses/a2d58ac8-399c-4417-ae8e-3394e6c429bc

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

Report for Core/storage/StakeStorage.sol
https://dashboard.mythx.io/#/console/analyses/08f3e488-87a7-4b68-b837-52cf11be56ed

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

Report for Core/interface/IStakeManager.sol
https://dashboard.mythx.io/#/console/analyses/ff90d90b-7992-4538-a05c-da3bcc626a3e

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

Report for Core/interface/IRewardManager.sol
https://dashboard.mythx.io/#/console/analyses/6a98baf1-95b9-4428-9272-f94802776fad

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

Report for Core/interface/IBlockManager.sol
https://dashboard.mythx.io/#/console/analyses/5b6eb788-5ae8-41e8-ba68-89e6ddaae71f

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

Report for Core/interface/IVoteManager.sol
https://dashboard.mythx.io/#/console/analyses/22a2a46c-8b49-4168-9852-d416c5b61087

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

Report for mocks/InitializableMock.sol
https://dashboard.mythx.io/#/console/analyses/9fc6c129-6707-41d1-b77a-0bb5113f8861

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

Report for mocks/MerklePosAwareTest.sol
https://dashboard.mythx.io/#/console/analyses/04238b86-7218-4a6a-a3e6-b50d7ba5806c

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

Report for contracts/randomNumber/RandomNoManager.sol
https://dashboard.mythx.io/#/console/analyses/e12b02d8-d69b-4f2f-992c-869fef7a1406

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

Report for contracts/randomNumber/RandomNoStorage.sol
https://dashboard.mythx.io/#/console/analyses/71a14db2-b7ce-46c2-b763-2cea8ca34d44

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

Report for node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol
https://dashboard.mythx.io/#/console/analyses/2afb871b-0c60-49f3-a20e-596064ab7069

Line	SWC Title	Severity	Short Description
183	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
206	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
239	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
241	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
262	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
263	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
288	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
290	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered
339	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered

Report for contracts/tokenization/RAZOR.sol
https://dashboard.mythx.io/#/console/analyses/48398c74-bd02-4032-944c-127496033231

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

Report for contracts/tokenization/StakedToken.sol
https://dashboard.mythx.io/#/console/analyses/2afb871b-0c60-49f3-a20e-596064ab7069

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.
44	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
58	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
58	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
82	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
85	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
86	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered

Report for contracts/tokenization/StakedTokenFactory.sol
https://dashboard.mythx.io/#/console/analyses/4575fc2e-aaff-4885-a6f4-53e6b46fa68f

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

- No major issues found by MythX.
- Integer Overflows and Underflows flagged by MythX are false positives, as all the contracts are using Solidity ^0.8.0 version.

After the Solidity version 0.8.0 Arithmetic operations revert to underflow and overflow by default.

THANK YOU FOR CHOOSING

// HALBORN