HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Consensus Networks
Date: 14 September, 2023

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Document

Name	Smart Contract Code Review and Security Analysis Report for Consensus Networks
Approved By	Paul Fomichov | Lead Solidity SC Auditor at Hacken OU
Tags	Ethereum Staking;
Platform	EVM
Language	Solidity
Methodology	Link
Website	https://consensusnetworks.com/
Changelog	05.08.2023 - Initial Review 14.09.2023 - Second Review

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Table of contents

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Introduction

Hacken OÜ (Consultant) was contracted by Consensus Networks (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

Casimir is a decentralized Ethereum staking protocol. Casimir is a complete platform that allows users to monitor, move, and stake their assets while holding their own keys. It is designed to offer users the experience and security of solo-staking while pooling their assets.

The scope of this audit contains the following contracts :

- **v1/CasimirManager.sol** - Accepts and distributes deposits.
- **v1/CasimirPool.sol** - Accepts deposits and stakes a validator.
- **v1/CasimirRegistry.sol** - Manages operator registration.
- **v1/CasimirUpkeep.sol** - Automates and handles reports.
- **v1/interfaces/ICasimirManager.sol** - Interface for CasimirManager.sol.
- **v1/interfaces/ICasimirPool.sol** - Interface for CasimirPool.sol.
- **v1/interfaces/ICasimirRegistry.sol** - Interface for CasimirRegistry.sol.
- **v1/interfaces/ICasimirUpkeep.sol** - Interface for CasimirUpkeep.sol.
- **v1/libraries/Types.sol** - Defines internal types.
- **v1/vendor/interfaces/IDepositContract.sol** - Interface of the Beacon deposit contract.
- **v1/vendor/interfaces/ISSVNetwork.sol** - Imports the ISSVNetwork interface.
- **v1/vendor/interfaces/ISSVNetworkCore.sol** - Imports the ISSVNetworkCore interface.
- **v1/vendor/interfaces/ISSVNetworkViews.sol** - Imports the ISSVNetworkViews interface
- **v1/vendor/interfaces/IWETH9.sol** - Interface for the WETH token.
- **v1/vendor/interfaces/KeeperRegistrarInterface.sol** - Interface for the keeper registrar.

Privileged roles

- **CasimirManager.sol :**
  - Owner :
    - Can withdraw a given amount of reserved fees.
    - Can withdraw a given amount from the LINK balance.
    - Can withdraw a given amount from the SSV balance.
    - Can update the functions oracle address.

- Pool :
  - Can deposit exited balance from a given pool ID.
- Oracle :
  - Can deposit to a cluster balance.
  - Can initiate the next ready pool.
  - Can report pool forced (unrequested) exits.
  - Can report a completed exit.
  - Can report a reshare.
  - Can cancel upkeep and withdraw the upkeep balance.
- Oracle or Registry :
  - Can request reshares for an operator.
- Oracle or Upkeep :
  - Can deposit to the upkeep balance.
  - Can activate a given count of the next pending pools.
  - Can request reports a given count of forced exits.
  - Can request reports for a given count of completed exits.
- Registry :
  - Can deposit recovered balance for a given pool from an operator.
- Upkeep :
  - Can rebalance the rewards to stake ratio and redistribute swept rewards.
  - Can compound rewards given a list of pool IDs.
  - Can fulfill a given count of pending withdrawals.

- **CasimirPool.sol :**
  - Owner :
    - Can deposit rewards from a pool to the manager.
    - Can withdraw balance from a pool to the manager.
    - Can set the operator IDs.
    - Can set the reshare count.
    - Can set the pool status.
- **CasimirRegistry.sol :**
  - Owner :
    - Can add a pool to an operator.
  - Owner or Pool :
    - Can remove a pool from an operator.
- **CasimirUpkeep. sol :**
  - Owner :
    - Can set the bytes representing the CBOR-encoded Functions.Request.
    - Can update the functions oracle address.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

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**.

- Functional requirements are detailed:
  - Project overview is detailed
  - All roles in the system are described.
  - Use cases are described and detailed.
  - All interactions are described.

- Technical description is robust:
  - Run instructions are provided.
  - Technical specification is provided.
  - NatSpec is sufficient.

Code quality

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

- The code contains no quality violations.

Test coverage

Code coverage of the project is **75.22%** (branch coverage).

- Coverage is not complete.
- Some tests are failing.

Security score

As a result of the audit, the code contains **no** issues. The security score is **10** out of **10**.

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

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Summary

According to the assessment, the Customer's smart contract has the following score: **9.0**.The system users should acknowledge all the risks summed up in the risks section of the report.

Table. The distribution of issues during the audit

Review date	Low	Medium	High	Critical
05 August 2023	6	6	3	0
08 September 2023	0	0	0	0

Risks

- ISSVNetwork.sol, ISSVNetworkCore.sol and ISSVNetworkViews.sol do not contain code but imports external contracts that are **not in the scope of this audit**.
- The system interacts with external contracts through interfaces (IDepositContract.sol, KeeperRegistrarInterface.sol, ISSVNetwork.sol, ISSVNetworkCore.sol and ISSVNetworkViews.sol). The contracts that will implement these interfaces and interact with the system are **not in the scope of this audit**.
- The system uses an internal oracle that is not described and its code is **not in the scope of this audit**.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Checked Items

We have audited the Customers' smart contracts for commonly known and specific vulnerabilities. Here are some items considered:

Item	Description	Status	Related Issues
Default Visibility	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	Passed	I02
Integer Overflow and Underflow	If unchecked math is used, all math operations should be safe from overflows and underflows.	Not Relevant	
Outdated Compiler Version	It is recommended to use a recent version of the Solidity compiler.	Passed	
Floating Pragma	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	Passed	
Unchecked Call Return Value	The return value of a message call should be checked.	Passed	
Access Control & Authorization	Ownership takeover should not be possible. All crucial functions should be protected. Users could not affect data that belongs to other users.	Passed	
SELFDESTRUCT Instruction	The contract should not be self-destructible while it has funds belonging to users.	Not Relevant	
Check-Effect-Interaction	Check-Effect-Interaction pattern should be followed if the code performs ANY external call.	Passed	
Assert Violation	Properly functioning code should never reach a failing assert statement.	Passed	
Deprecated Solidity Functions	Deprecated built-in functions should never be used.	Passed	
Delegatecall to Untrusted Callee	Delegatecalls should only be allowed to trusted addresses.	Not Relevant	
DoS (Denial of Service)	Execution of the code should never be blocked by a specific contract state unless required.	Passed	

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Race Conditions	Race Conditions and Transactions Order Dependency should not be possible.	Passed	H03
Authorization through tx.origin	tx.origin should not be used for authorization.	Not Relevant	
Block values as a proxy for time	Block numbers should not be used for time calculations.	Not Relevant	
Signature Unique Id	Signed messages should always have a unique id. A transaction hash should not be used as a unique id. Chain identifiers should always be used. All parameters from the signature should be used in signer recovery. EIP-712 should be followed during a signer verification.	Not Relevant	
Shadowing State Variable	State variables should not be shadowed.	Passed	I05
Weak Sources of Randomness	Random values should never be generated from Chain Attributes or be predictable.	Not Relevant	
Incorrect Inheritance Order	When inheriting multiple contracts, especially if they have identical functions, a developer should carefully specify inheritance in the correct order.	Not Relevant	
Calls Only to Trusted Addresses	All external calls should be performed only to trusted addresses.	Passed	
Presence of Unused Variables	The code should not contain unused variables if this is not justified by design.	Passed	I07
EIP Standards Violation	EIP standards should not be violated.	Passed	
Assets Integrity	Funds are protected and cannot be withdrawn without proper permissions or be locked on the contract.	Passed	H01, M05
User Balances Manipulation	Contract owners or any other third party should not be able to access funds belonging to users.	Failed	H02
Data Consistency	Smart contract data should be consistent all over the data flow.	Passed	M02, M06

Flashloan Attack	When working with exchange rates, they should be received from a trusted source and not be vulnerable to short-term rate changes that can be achieved by using flash loans. Oracles should be used. Contracts shouldn't rely on values that can be changed in the same transaction.	Not Relevant	
Token Supply Manipulation	Tokens can be minted only according to rules specified in a whitepaper or any other documentation provided by the Customer.	Not Relevant	
Gas Limit and Loops	Transaction execution costs should not depend dramatically on the amount of data stored on the contract. There should not be any cases when execution fails due to the block Gas limit.	Passed	
Style Guide Violation	Style guides and best practices should be followed.	Passed	I01, I04
Requirements Compliance	The code should be compliant with the requirements provided by the Customer.	Failed	H02
Environment Consistency	The project should contain a configured development environment with a comprehensive description of how to compile, build and deploy the code.	Passed	
Secure Oracles Usage	The code should have the ability to pause specific data feeds that it relies on. This should be done to protect a contract from compromised oracles.	Passed	
Tests Coverage	The code should be covered with unit tests. Test coverage should be sufficient, with both negative and positive cases covered. Usage of contracts by multiple users should be tested.	Failed	
Stable Imports	The code should not reference draft contracts, which may be changed in the future.	Passed	M01

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Findings

■■■■ Critical

No critical severity issues were found.

■■■ High

H01. Funds Lock

Impact	High
Likelihood	Medium

An operator can request a withdrawal of his collateral through the function requestWithdrawal(). This function allows him to withdraw a specified amount of his collateral that must be higher or equal than the required collateral (4 ether). In some specific cases, some collateral may be locked.

Let's assume the following scenario :

- An operator deposits a collateral of 6 ETH.
- The operator requests a withdrawal of 4 ETH. The withdrawal is accepted.
- The operator has only 2 ETH left as collateral. He can not withdraw them.

Another scenario :

- An operator deposits a collateral of 4 ETH.
- The operator has to pay a blame amount of 0.5 ETH.
- The operator has only 3.5 ETH left as collateral. He can not withdraw it.

Path: ./contracts/ethereum/src/v1/CasimirRegistry.sol : requestWithdrawal()

Recommendation: allow operators to withdraw their full collateral if it is lower than the required collateral.

Found in: 4a4ab39

Status: Fixed (4f246eb)

H02. Undocumented Behavior - Unlimited Variables - Highly Permissive Role Access

Impact	High
Likelihood	Medium

It is specified in the documentation that *"Collateral is used to recover lost validator effective balance at the time of completing an exit. The loss blame is assigned to the four responsible operators based on performance over the duration of the validator's existence."* However, it is never specified what would be the amount of the blame amount nor it is limited in the implemented code. This is, therefore, possible for the owner to use any blame amount, which can lead to abuse.

Path: ./contracts/ethereum/src/v1/*

Recommendation: inform the users about the blame amount and implement limits in the code to guarantee this amount.

Found in: 4a4ab39

Status: New

H03. Front Running

Impact	High
Likelihood	Medium

The amount out value is set to 0 when interacting with Uniswap.

This allows attackers to perform front running attacks and the operations may result in an unexpected amount of tokens out.

Path: ./contracts/ethereum/src/v1/CasimirManager.sol : processFees()

Recommendation: specify the amount out values when interacting with Uniswap.

Found in: 4a4ab39

Status: Fixed (4f246eb)

Medium

M01. Non-Finalized Code

Impact	Low
Likelihood	High

Part of the code is used for testing or developer tools code insertions. When ready for production, none of the code used for testing should remain.

Path: ./contracts/ethereum/src/v1/CasimirUpkeep.sol : mockFulfillRequest()

./contracts/ethereum/src/v1/CasimirViews.sol : getValidatorPublicKeys()

Recommendation: remove the code that is not designed for production.

Found in: 4a4ab39

Status: Fixed (4f246eb)

M02. Data Consistency

Impact	Medium
Likelihood	Medium

When adding a pool to an operator, several verifications are done. One of these is to check that the operator has collateral, but instead of checking if the operator collateral is higher or equal than the required collateral, it is only checked that it is higher or equal than 0.

Path: ./contracts/ethereum/src/v1/CasimirRegistry.sol : addOperatorPool()

Recommendation: check that the operator collateral is higher or equal than the required collateral.

Found in: 4a4ab39

Status: Fixed (4f246eb)

M03. Missing Events on Critical State Updates

Impact	Low
Likelihood	High

Critical state changes should emit events for tracking things off-chain.

This can lead to inability for users to subscribe events and check what is going on with the project.

Paths: ./contracts/ethereum/src/v1/CasimirManager.sol : depositExitedBalance(), depositRecoveredBalance(), depositClusterBalance(), depositUpkeepBalance(), depositReservedFees(), withdrawReservedFees(), fulfillWithdrawal(), registerPool(), reportForcedExits(), withdrawUpkeepBalance(), withdrawLINKBalance(), withdrawSSVBalance(), setFunctionsAddress()

./contracts/ethereum/src/v1/CasimirPool.sol : setOperatorIds(), setReshares(), setStatus()

./contracts/ethereum/src/v1/CasimirRegistry.sol : depositCollateral(), requestWithdrawal(), requestDeregistration(), addOperatorPool(), removeOperatorPool()

./contracts/ethereum/src/v1/CasimirUpkeep.sol : generateRequest(), setRequest(), setOracleAddress()

Recommendation: emit events on critical state changes.

Found in: 4a4ab39

Status: Fixed (4f246eb)

M04. Admin Functions Not Restricted

Impact	Medium
Likelihood	Medium

depositRewards() and depositReservedFees() are admin functions, but they are accessible by any users.

Path: ./contracts/ethereum/src/v1/CasimirManager.sol : depositRewards(), depositReservedFees()

Recommendation: restrict the access to these functions.

Found in: 4a4ab39

Status: Fixed (4f246eb)

M05. Funds Lock

Impact	Medium
Likelihood	Medium

A user can request a withdrawal of his stake through the function requestWithdrawal(). This function allows him to withdraw a specified amount of his stake that must be higher or equal than the minimum stake (0.0001 ether). If his remaining stake is lower than this minimum it will remain locked.

Path: ./contracts/ethereum/src/v1/CasimirManager.sol : RequestWithdrawal()

Recommendation: allow stakers to withdraw their full collateral if it is lower than the minimum stake.

Found in: 4a4ab39

Status: Fixed (4f246eb)

M06. Data Consistency

Impact	High
Likelihood	Low

The function *registerOperator()* can be called already with a pre-existing operator id, and overwrite the collateral field.

This leads to the locking of funds in the smart contract. The operator loses his collateral, which leads to unpredictable behavior of the smart contract.

Path: ./contracts/ethereum/src/v1/CasimirRegistry.sol : registerOperator()

Recommendation: add a check for the existence of the operator.

Found in: 4a4ab39

Status: Fixed (4f246eb)

Low

L01. State Variables That Should Be Constant

Impact	Low
Likelihood	Medium

State variables that do not change their value should be declared constant to save Gas.

Paths: ./contracts/ethereum/src/v1/CasimirPool.sol : poolCapacity

./contracts/ethereum/src/v1/CasimirManager.sol : feePercent, upkeepRegistrationMinimum

./contracts/ethereum/src/v1/CasimirUpkeep.sol : minimumCollateralDeposit, requiredCollateral

Recommendation: declare the above-mentioned variables as constants.

Found in: 4a4ab39

Status: Fixed (4f246eb)

L02. Public Functions That Should Be External

Impact	Low
Likelihood	Medium

Functions that are only called from outside the contract should be defined as external. External functions are much more Gas efficient compared to public functions.

Paths: ./contracts/ethereum/src/v1/CasimirUpkeep.sol : generateRequest()

./contracts/ethereum/src/v1/CasimirManager.sol : getReservedFeeBalance()

./contracts/ethereum/src/v1/CasimirViews.sol : getSweptBalance()

Recommendation: make these functions external.

Found in: 4a4ab39

Status: Fixed (4f246eb)

L03. Best Practice Violation

Impact	Medium
Likelihood	Low

The functions do not use SafeERC20 library for checking the result of ERC20 token transfers.

Path: ./contracts/ethereum/src/v1/CasimirManager.sol : withdrawLINKBalance(), withdrawSSVBalance()

Recommendation: use SafeERC20 library to interact with tokens safely.

Found in: 4a4ab39

Status: Fixed (4f246eb)

L04. Environment Configuration

Impact	Low
Likelihood	Medium

The development environment is not configured to run tests and tools without the necessity of third-party tools or access rights.

Paths: ./*

Recommendation: configure the development environment properly.

Found in: 4a4ab39

Status: Fixed (4f246eb)

L05. State Variables That Should Be Immutable

Impact	Low
Likelihood	Medium

State variables that do not change their value after being initialized in the constructor should be declared immutable to save Gas.

Path: ./contracts/ethereum/src/v1/CasimirPool.sol : id

Recommendation: declare the above-mentioned variables as immutable.

Found in: 4a4ab39

Status: Fixed (4f246eb)

L06. Costly Operation Inside a Loop

Impact	Low
Likelihood	Medium

Allocating a new value to a state variable is costly. When updating a value multiple times inside a loop, it is a best practice to copy the value inside a memory variable and update the memory variable. At the end of the loop, the value should be reassigned to the state variable.

By assigning a new value to the state variable only once, some Gas will be saved.

Path: ./contracts/ethereum/src/v1/CasimirManager.sol : fulfillWithdrawals(), fulfillWithdrawal(), reportForcedExits()

Recommendation: create a variable in memory to be updated several times inside the loops and then assign the value to the state variable.

Found in: 4a4ab39

Status: Fixed (4f246eb)

Informational

I01. Style Guide Violation

Functions should be grouped according to their visibility and ordered:

1. constructor
2. receive function (if exists)

3. fallback function (if exists)
4. external
5. public
6. internal
7. private

Within a grouping, place the view and pure functions last.

Path: ./contracts/ethereum/src/v1/*

Recommendation: follow the official Solidity guidelines.

Found in: 4a4ab39

Status: Fixed (4f246eb)

I02. State Variables Default Visibility

Some variables' visibility is not specified. Specifying state variables visibility helps to catch incorrect assumptions about who can access the variable.

This makes the contract`s code quality and readability higher.

Paths: ./contracts/ethereum/src/v1/CasimirUpkeep.sol : reportPeriod, reportRemainingRequests, reportRequestBlock, reportCompoundablePoolIds, finalizableCompoundablePoolIds, fulfillGasLimit

./contracts/ethereum/src/v1/CasimirPool.sol : poolCapacity

./contracts/ethereum/src/v1/CasimirManager.sol : upkeepRegistrationMinimum

Recommendation: specify variables as public, internal, or private. Explicitly define visibility for all state variables.

Found in: 4a4ab39

Status: Fixed (4f246eb)

I03. Missing Zero Address Validation

Address parameters are being used without checking against the possibility of 0x0.

This can lead to unwanted external calls to 0x0.

Paths: ./contracts/ethereum/src/v1/CasimirManager.sol : constructor(), setFunctionsAddress()

./contracts/ethereum/src/v1/CasimirPool.sol : constructor()

./contracts/ethereum/src/v1/CasimirRegistry.sol : constructor()

./contracts/ethereum/src/v1/CasimirUpkeep.sol : constructor()

./contracts/ethereum/src/v1/CasimirViews.sol : constructor()

Recommendation: implement zero address checks.

Found in: 4a4ab39

Status: Fixed (4f246eb)

I04. Style Guide Violation

Constant state variables should be in UPPER_CASE_WITH_UNDERSCORES.

Path: ./contracts/ethereum/src/v1/CasimirManager.sol : actionPeriod, maxActionsPerPeriod, compoundMinimum, stakeMinimum, scaleFactor, uniswapFeeTier, poolCapacity

Recommendation: follow the official Solidity guidelines.

Found in: 4a4ab39

Status: Fixed (4f246eb)

I05. Variable Shadowing

CasimirPool.setStatus(ICasimirPool.PoolStatus)._status shadows:

- ReentrancyGuard._status

ICasimirPool.setReshares(uint256).reshares shadows:

- ICasimirPool.reshares()

ICasimirPool.setStatus(ICasimirPool.PoolStatus).status shadows:

- ICasimirPool.status()

Path: ./contracts/ethereum/src/v1/CasimirPool.sol : setStatus()

./contracts/ethereum/src/v1/interfaces/ICasimirPool.sol : setReshares(), setStatus()

Recommendation: rename the variables.

Found in: 4a4ab39

Status: Fixed (4f246eb)

I06. Unused Function

The private function *withdrawClusterBalance()* is never used.

Path: ./contracts/ethereum/src/v1/CasimirManager.sol : withdrawClusterBalance()

Recommendation: remove unused function.

Found in: 4a4ab39

Status: Fixed (4f246eb)

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

I07. Unused Variable

Impact	Low
Likelihood	Medium

The state variable *poolCapacity* is never used.

Path: ./contracts/ethereum/src/v1/CasimirUpkeep.sol : pool

Recommendation: remove unused variable.

Found in: 4a4ab39

Status: Fixed (4f246eb)

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

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.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

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.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

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.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Appendix 2. Scope

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

Initial review scope

Repository	https://github.com/consensusnetworks/casimir
Commit	4a4ab39b7d1faec2ffb6b3a26e633b084343f3f4
Whitepaper	-
Requirements	https://github.com/consensusnetworks/casimir/blob/audit/hacken/contracts/ethereum/README.md
Technical Requirements	https://github.com/consensusnetworks/casimir/blob/audit/hacken/contracts/ethereum/README.md
Contracts	File: ./ethereum/src/v1/CasimirManager.sol SHA3: ec24c274dc7491fab64852c1be45157785867219d88de12d95bd1fe333caeb28 File: ./ethereum/src/v1/CasimirPool.sol SHA3: 487f60bc8a5b2b71b152e508a97c690e205b8b69d05e617ee7fe06fe2f2d5cb4 File: ./ethereum/src/v1/CasimirRegistry.sol SHA3: 4e82b97b45cb9c938387e99a80e53aab8f8f9721feb3fc004d619b3b22fc5956 File: ./ethereum/src/v1/CasimirUpkeep.sol SHA3: 99aacb24cbaef21906ec516fd3572b8f920bb165ae062a540795eed2f732874f File: ./ethereum/src/v1/interfaces/ICasimirManager.sol SHA3: fba21afaca2396971c3883afde8dc14558945f0ed4ce68e19047c9c15396e6c8 File: ./ethereum/src/v1/interfaces/ICasimirPool.sol SHA3: a3eeebc904d1d5c1867a9e8a8de9e8ea48a82ae5e64fe6dde81d528c76140203 File: ./ethereum/src/v1/interfaces/ICasimirRegistry.sol SHA3: d25dd8933f70b08a20ad0d046bfe72454461f48f4e031d70151ef2de753c7e4b File: ./ethereum/src/v1/interfaces/ICasimirUpkeep.sol SHA3: a50bbd0319f4ac882ab4f0835fc57a9fd27bb61d05c5e41394faa409445224e5 File: ./ethereum/src/v1/libraries/Types.sol SHA3: c232e66716cdae0879c80803f6e2446ce607987eefdc49833d344f10a394e26a File: ./ethereum/src/v1/vendor/interfaces/IDepositContract.sol SHA3: 2c158885904349cbf85f61c320faad01f4ce2690d1bd7bdc6810519ec3062050 File: ./ethereum/src/v1/vendor/interfaces/ISSVNetwork.sol SHA3: 4128c258476be08aab12af3ad210f8ef1f409096c76c4ec50c8b22ef736f792e File: ./ethereum/src/v1/vendor/interfaces/ISSVNetworkCore.sol SHA3: 303c6f82655839d7b23585c1dbed01e944514552683956b4ebdf3dde3b1e92a9 File: ./ethereum/src/v1/vendor/interfaces/ISSVNetworkViews.sol SHA3: 980cc7f0e519095113d2e7eb044be6f0cd7ef80de281eab534c5d3f75b2ba66e File: ./ethereum/src/v1/vendor/interfaces/IWETH9.sol SHA3: eb85a4c21c2f8cdba2c63eb8f639c2f5bf7b9d8c81c97062234c94dbd525283e

	File: ./ethereum/src/v1/vendor/interfaces/KeeperRegistrarInterface.sol SHA3: 643c60d28ee8c61e9db7c16d06a5983f13150c4fee26de7a83fbb8bf230527aa