HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Brokkr
Date: 11 July, 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 Brokkr
Approved By	Noah Jelich | Lead Solidity SC Auditor at Hacken OU
Type	ERC20 token; Staking
Platform	EVM; Avalanche
Language	Solidity
Methodology	Link
Website	https://brokkr.finance
Changelog	15.05.2023 - Initial Review 15.06.2023 - 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 Brokkr (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

Brokkr is an investment protocol that provides sustainable and predictable returns through automated portfolios, and instruments that consist of several investment strategies that combine the usage of multiple dApps. It has the following contracts:

- *BroToken* – simple ERC-20 token that mints all initial supply to a specific address. Additional minting is not allowed. Owner can change name and symbol.
- *IndexAvalanche* – simple contract that extends *IndexStrategyUpgradeable*, with 3 ready implementations: equityValuation() and addSwapRoute() (2 different implementations) and is extended from other strategy contracts.
- *IndexStrategyUpgradeable* – base contract of the system. Implements the core features intended for all other trading strategy contracts and it is upgradeable.
- *IndexAvalancheDeFi* – simple contract that extends *IndexAvalanche* without any new implementations.
- *IndexAvalancheGamingNFT* – simple contract that extends *IndexAvalanche* without any new implementations.
- *OracleAvalanche* – an upgradeable contract that communicates with oracles in order to provide exchange rates for the tokens in the system.
- *IndexToken* – simple upgradeable ERC-20 token that mints all initial supply to a specific address. Additional minting is not allowed. Owner can change name and symbol.

Privileged roles

- The owner of the *IndexAvalanche* contract can add swap routes.
- The owner of the *IndexStrategyUpgradeable* contract can pause, unpause, rebalance system weights, manage components, manage swap routes, manage whitelisted tokens, set equity valuation limit and set oracle contract address.
- The owner of the *IndexToken* contract can mint, change the name and symbol of the token.
- Due to UUPS upgradeability, owners can authorize contracts implementations upgrades.

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 provided.
- Technical description is provided.

Code quality

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

- The code follows style guides and best practices.
- The development environment is configured.
- Instructions to set up the development environment are sufficient.

Test coverage

Code coverage of the project is **75.9%** (branch coverage).

- Deployment and basic user interactions are covered with tests.
- Not all possibilities (conditions, returns) are covered with tests.

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.

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
15 May 2023	1	2	0	0

15 June 2023	1	0	0	0

Risks

- Iterating over a dynamic array populated with custom tokenId can lead to Gas limit denial of service if the number of tokenId goes out of control.
- Front-running attacks can cause users to lose funds due to price manipulations. Smart contracts rely on data provided by the front-end in order to not suffer from front-running attacks that could cause higher slippage.

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	1.
Integer Overflow and Underflow	If unchecked math is used, all math operations should be safe from overflows and underflows.	Passed	
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.	Not Relevant	
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	

Race Conditions	Race Conditions and Transactions Order Dependency should not be possible.	Passed	
Authorization through tx.origin	tx.origin should not be used for authorization.	Passed	
Block values as a proxy for time	Block numbers should not be used for time calculations.	Passed	
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	
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.	Passed	
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	
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	
User Balances Manipulation	Contract owners or any other third party should not be able to access funds belonging to users.	Passed	
Data Consistency	Smart contract data should be consistent all over the data flow.	Passed	

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

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.	Passed	
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	
Requirements Compliance	The code should be compliant with the requirements provided by the Customer.	Passed	
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	

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. Front Running Attack

Impact	High
Likelihood	Medium

The functions swap tokens without setting proper slippage. This is because the function uses a user-supplied slippage value, which enables users to set *amountOutMin* as zero. This makes performing sandwich attacks to get profit on all users of the system possible.

This can lead to a loss of funds.

Paths: index/libraries/UniswapV2Library.sol : swapTokensForExactTokens(), swapExactTokensForTokens()

index/libraries/SwapAdapter.sol: swapTokensForExactTokens(), swapExactTokensForTokens()

Recommendation: Make sure that the *amountOutMin* parameter is greater than zero.

Found in: 66d0b81cb379bcf44874cf4b86622a7bab9d80f3t

Status: Mitigated. The related check has been implemented in the front-end. See the risks section for details.

Medium

M01. Contradiction - Unfinalized Code

Impact	Medium
Likelihood	Medium

The provided code should be implemented in the full logic of the project. Since any missing parts, TODOs, or drafts can change in time, the robustness of the audit cannot be guaranteed.

Path: index/oracles/OracleAvalanche.sol: _getAmmPrice()

Recommendation: Complete the code to meet all the requirements and delete the TODO comments.

Found in: 66d0b81cb379bcf44874cf4b86622a7bab9d80f3

Status: Fixed

Low

L01. Missing Zero Address Validation

Impact	Low
Likelihood	Medium

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

This can lead to unwanted external calls to 0x0.

Paths: index/oracles/OracleAvalanche.sol : setPriceFeed()

index/bases/IndexStrategyUpgradeable.sol : _addRouter(), _setDEX()

Recommendation: Implement zero address checks.

Found in: 66d0b81cb379bcf44874cf4b86622a7bab9d80f3

Status: Fixed (Revised Commit: c0706dbc067b549f06e46fcc7ec91bf0f1c5ff0f)

Informational

I01. Floating Pragma

The project uses floating pragmas 0.8.0

Paths: index/libraries/SwapAdapter.sol

index/token/IndexToken.sol

index/oracles/OracleAvalanche.sol

index/libraries/UniswapV2Library.sol

index/libraries/TraderJoeV2Library.sol

index/bases/IndexAvalanche.sol

index/indices/IndexAvalancheDeFi.sol

index/bases/IndexStrategyUpgradeable.sol

contracts/BroToken.sol

Recommendation: Consider locking the pragma version whenever possible and avoid using a floating pragma in the final deployment.

Found in: 66d0b81cb379bcf44874cf4b86622a7bab9d80f3, 75a01b6aeada35a1bf4644637f44f91cc7ce728f

Status: Fixed (Revised Commit: 20c0ae50e9eb9e92c362ed464e5d457bcbcbe882, 7b48e080d7b2fdc9fd92f32e6a5595dbae903411)

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

I02. Function That Can Be Declared External

"public" functions that are never called by the contract should be declared "external" to save Gas.

Notice: it is also applicable to "initialize" function in upgradable contracts. There is no magic in declaring them public if the contract is not inherited.

Path: index/token/IndexToken.sol : initialize()

Recommendation: Change function visibility to external.

Found in: 66d0b81cb379bcf44874cf4b86622a7bab9d80f3

Status: Fixed

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/BrokkrFinance/bro-strategies/tree/main/contracts/index https://github.com/BrokkrFinance/dao-core/blob/main/contracts/BroToken.sol
Commit	66d0b81cb379bcf44874cf4b86622a7bab9d80f3, 75a01b6aeada35a1bf4644637f44f91cc7ce728f
Whitepaper	Link
Requirements	Link
Technical Requirements	Link
Contracts	File: bases/IndexAvalanche.sol SHA3: cf7dcbcea02ab661f5fe369be864f76a970c7078a2cdff50df2dbbd4540b1903 File: bases/IndexStrategyUpgradeable.sol SHA3: 08bfd815467e2826852e735502e945669c366e915d502129e6759b339b1a46c8 File: dependencies/IChainlinkAggregatorV3.sol SHA3: 3b9286c8a7d0dc578b50f0abded6b63c6bb275c7080dc26bf1b037a9397d70ee File: dependencies/ITraderJoeV2Pair.sol SHA3: 1928fb17681f6bdbc9f9746b3116ad2ce8ea8d02ab26d303637faa0b7cbdbff1 File: dependencies/ITraderJoeV2Router.sol SHA3: 945b2109ff8a76c72006b7a696370c81d9a6503befd94ee653fb09e714c25133 File: dependencies/IUniswapV2Pair.sol SHA3: 105fa3dc465f990d5d31a59559cb237b3c5f8d345c5f590daf55020fcd33f7f8 File: dependencies/IUniswapV2Router.sol SHA3: e6029529e0d281168375b1cf745460b503b8080917c42911eafdfa1ae8ae97de File: indices/IndexAvalancheDeFi.sol SHA3: b1d8c15c05839290a38e46aa76694e1bf1fa8aece44f37df1e819d796f94d42a File: indices/IndexAvalancheGamingNFT.sol SHA3: 1f063f2cbc3a49a2af1ab1c3bcf0886055c68ea9509828280e9b4be09d013c09 File: interfaces/IIndexInit.sol SHA3: 3cdd632209b7d9379dcba7f134b24e1ebb1b0b83f496be7590369fa7e192e2fb File: interfaces/IIndexLimits.sol SHA3: 7a294eeb92d3ccca4d418dd5f94c8f19e1d677125ed94d479d6ee1c685e02df6 File: interfaces/IIndexOracle.sol SHA3: 4f0e77d51bd01c12426d023bd63424edaef3be2faa3fcd399685b5e2e7dd4794 File: interfaces/IIndexStrategy.sol SHA3: 29f57ebfc6b95eadf5e759ca4701de4a9c6018a13a1495a441eca7634fb5891c

	File: interfaces/IIndexToken.sol SHA3: cdd4032971ed0b4b46ba1910f41ada2a8822d1394f1b7380b3cc42d5c98b6cd7 File: libraries/Constants.sol SHA3: fb7f52389c316f488946e55d53ed12a87bf1c417996d0af8ca66e8a740b099fa File: libraries/Errors.sol SHA3: e63dcddf398b515f31146095af89797ca6e7aba9a574ba74acb67146678c3175 File: libraries/SwapAdapter.sol SHA3: e1b31a0dc282475a97065f81e2617eebb6fe9179ea4f1f433c9cbc3f96a95bf2 File: libraries/TraderJoeV2Library.sol SHA3: 8a5263e75c834530fef5f09dc19c2b30a5e42924116695af3668c03a1ec0b62f File: libraries/UniswapV2Library.sol SHA3: ff3ec2c5307c054967340ae451c820d8cbec22167af5a404965421694100d81d File: oracles/OracleAvalanche.sol SHA3: e8a3ae5e0a0810c802e3f7b33f05226e03a496fe5fd5b94b52609572c47c176a File: token/IndexToken.sol SHA3: 18ab555e789ef86fa774d6728e5b227738b7fc79348fb2ab2593458d8d85fa53 File: contracts/BroToken.sol SHA3: 5a2e201cad32be91089c6e221b53591b50e8478869a34463cfc29fddfe7f6fb8

Second review scope

Repository	https://github.com/BrokkrFinance/bro-strategies/tree/main/contracts/index https://github.com/BrokkrFinance/dao-core/blob/main/contracts/BroToken.sol
Commit	51946b313f91685129677a90d8545d13bdce8568, 75a01b6aeada35a1bf4644637f44f91cc7ce728f
Whitepaper	Link
Requirements	Link
Technical Requirements	Link
Contracts	File: bases/IndexAvalanche.sol SHA3: 1d1eec096057b106a53f8f65a67b6095b444e186b3698ebce99c0ae7e6275355 File: bases/IndexStrategyUpgradeable.sol SHA3: ea0f344b03528b721bf262010bdf9ed5359993a1acd501da393006754d9dc000 File: dependencies/IChainlinkAggregatorV3.sol SHA3: 3b9286c8a7d0dc578b50f0abded6b63c6bb275c7080dc26bf1b037a9397d70ee File: dependencies/ITraderJoeV2Pair.sol SHA3: 1928fb17681f6bdbc9f9746b3116ad2ce8ea8d02ab26d303637faa0b7cbdbff1 File: dependencies/ITraderJoeV2Router.sol SHA3: 945b2109ff8a76c72006b7a696370c81d9a6503befd94ee653fb09e714c25133 File: dependencies/IUniswapV2Pair.sol

	SHA3: 105fa3dc465f990d5d31a59559cb237b3c5f8d345c5f590daf55020fcd33f7f8 File: dependencies/IUniswapV2Router.sol SHA3: e6029529e0d281168375b1cf745460b503b8080917c42911eafdfa1ae8ae97de File: indices/IndexAvalancheDeFi.sol SHA3: b1d8c15c05839290a38e46aa76694e1bf1fa8aece44f37df1e819d796f94d42a File: indices/IndexAvalancheGamingNFT.sol SHA3: 1f063f2cbc3a49a2af1ab1c3bcf0886055c68ea9509828280e9b4be09d013c09 File: interfaces/IIndexInit.sol SHA3: 3cdd632209b7d9379dcba7f134b24e1ebb1b0b83f496be7590369fa7e192e2fb File: interfaces/IIndexLimits.sol SHA3: 7a294eeb92d3ccca4d418dd5f94c8f19e1d677125ed94d479d6ee1c685e02df6 File: interfaces/IIndexOracle.sol SHA3: 4f0e77d51bd01c12426d023bd63424edaef3be2faa3fcd399685b5e2e7dd4794 File: interfaces/IIndexStrategy.sol SHA3: 29f57ebfc6b95eadf5e759ca4701de4a9c6018a13a1495a441eca7634fb5891c File: interfaces/IIndexToken.sol SHA3: cdd4032971ed0b4b46ba1910f41ada2a8822d1394f1b7380b3cc42d5c98b6cd7 File: libraries/Constants.sol SHA3: fb7f52389c316f488946e55d53ed12a87bf1c417996d0af8ca66e8a740b099fa File: libraries/Errors.sol SHA3: c76f1e067470a3d2084fa0578ce3526a3938bee6c1a83c99e62bb5e4d6120bcf File: libraries/SwapAdapter.sol SHA3: aafc1b6fc3dbfad1c6e9a8dcb9b28e1c75bf38242abf18833cf248c40bbe902c File: libraries/TraderJoeV2Library.sol SHA3: 8a5263e75c834530fef5f09dc19c2b30a5e42924116695af3668c03a1ec0b62f File: libraries/UniswapV2Library.sol SHA3: ff3ec2c5307c054967340ae451c820d8cbec22167af5a404965421694100d81d File: oracles/OracleAvalanche.sol SHA3: a16b00bae4644b1b517ad74d8330ed94fc9eea8ac23fd0b12b15eb02263b9879 File: token/IndexToken.sol SHA3: cadb04554cd69f2c436ad277150f378c680d853591eb182ff46514d760a2a2de File: contracts/BroToken.sol SHA3: 5a2e201cad32be91089c6e221b53591b50e8478869a34463cfc29fddfe7f6fb8