HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Dexalot
Date: 22 May, 2023

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

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

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

Document

Name	Smart Contract Code Review and Security Analysis Report for Dexalot
Approved By	Noah Jelich | Lead Solidity SC Auditor at Hacken OU
Type	Request For Quote
Platform	EVM
Language	Solidity
Methodology	Link
Website	https://dexalot.com/
Changelog	25.04.2023 - Initial Review 16.05.2023 - Second Review 22.05.2023 - Third 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 Dexalot (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.

Scope

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

Initial review scope

Repository	https://github.com/Dexalot/contracts
Commit	f8881f901e3680cdf281de7ef8e2812e4a89ec8d
Whitepaper	Link
Functional Requirements	Link
Technical Requirements	Link
Contracts	File: contracts/MainnetRFQ.sol SHA3: 334e4563a80a14c1707118924c89971eb32b9d407d94be8778597b06202d4ad8

Second review scope

Repository	https://github.com/Dexalot/contracts
Commit	4d650f9152b5c90a63a25f13c2a0176c2632526d
Whitepaper	Link
Requirements	Link
Technical Requirements	Link
Contracts	File: contracts/MainnetRFQ.sol SHA3: 36be1f2e5698e8e9b9e9c0aa7efc002d60f48d2d5eaaf83c179356b307e3c12b

Third review scope

Repository	https://github.com/Dexalot/contracts
Commit	e2cfd502dd25949661675f5f905f8506ae112477
Whitepaper	Link

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

Requirements	Link
Technical Requirements	Link
Contracts	File: contracts/MainnetRFQ.sol SHA3: 94c7dc33ae76ba2502a07fd48760687ff4b1aa11799aad1186c2d9b7011b0a1b

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

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to the loss of user funds or contract state manipulation by external or internal actors.
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 by external or internal actors.
Medium	Medium vulnerabilities are usually limited to state manipulations but cannot lead to asset loss. Major deviations from best practices are also in this category.
Low	Low vulnerabilities are related to outdated and unused code or minor Gas optimization. These issues won't have a significant impact on code execution but affect code quality

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 **7** out of **10**.

- Functional requirements are missing. Only the Litepaper is provided, but contract specific description is limited to technical description.
- Technical specifications, including NatSpec are provided and very detailed.
- Description of the development environment is sufficient.

Code quality

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

- The development environment is configured.
- Style guides are not followed perfectly, but the function organization makes sense.

Test coverage

Code coverage of the project is **100%** (branch coverage).

- Deployment and basic user interactions are covered with tests.
- Negative cases are covered.
- Interactions by several users are not tested thoroughly.

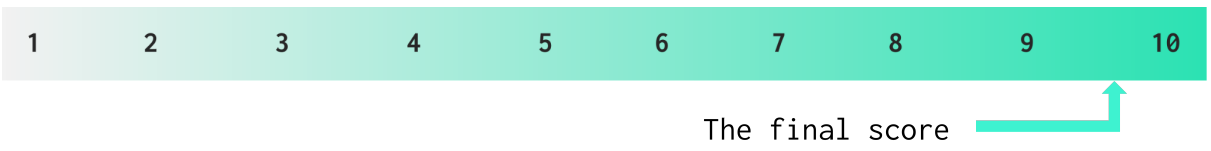
Security score

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: **9.7**. The system users should acknowledge all the risks summed up in the risks section of the report.



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

Table. The distribution of issues during the audit

Review date	Low	Medium	High	Critical
25 April 2023	5	0	1	0
16 May 2023	1	0	0	0
22 May 2023	0	0	0	0

Risks

- The **off-chain REST API used to get a signed quote** that also **determines the swap rate** of the **assets** is **out of this audit scope** and its **security can not be guaranteed**.

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

System Overview

The scope of this audit consists of an upgradeable contract that handles swapping of any two assets based on a signed quote that is generated through an off-chain REST API. The swapping details, such as the amounts and receivers, are determined by the quote generated by the REST API.

The files in the scope:

- **MainnetRFQ.sol** - The contract that handles the signature verified swapping.

Privileged roles

- swapSigner: creates signature.
- rebalancer: rebalances inventory of the smart contract, updates quote expiry and quote maker amount.
- default admin: manages swapSigner and rebalancer addresses. Sets trusted contracts, changes the admin, and can pause/unpause the contract, set slippage tolerance.
- trusted contracts: can initiate a swap manually, without the need for user interaction.

Recommendations

- In the `batchClaimBalance()` function, read the `rebalancer` variable to memory and use that instead of reading from storage in every iteration.

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	Type	Description	Status
Default Visibility	SWC-100 SWC-108	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	Passed
Integer Overflow and Underflow	SWC-101	If unchecked math is used, all math operations should be safe from overflows and underflows.	Passed
Outdated Compiler Version	SWC-102	It is recommended to use a recent version of the Solidity compiler.	Passed
Floating Pragma	SWC-103	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	Passed
Unchecked Call Return Value	SWC-104	The return value of a message call should be checked.	Passed
Access Control & Authorization	CWE-284	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	SWC-106	The contract should not be self-destructible while it has funds belonging to users.	Not Relevant
Check-Effect-Interaction	SWC-107	Check-Effect-Interaction pattern should be followed if the code performs ANY external call.	Passed
Assert Violation	SWC-110	Properly functioning code should never reach a failing assert statement.	Passed
Deprecated Solidity Functions	SWC-111	Deprecated built-in functions should never be used.	Passed
Delegatecall to Untrusted Callee	SWC-112	Delegatecalls should only be allowed to trusted addresses.	Not Relevant
DoS (Denial of Service)	SWC-113 SWC-128	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	SWC-114	Race Conditions and Transactions Order Dependency should not be possible.	Passed
Authorization through tx.origin	SWC-115	tx.origin should not be used for authorization.	Not Relevant
Block values as a proxy for time	SWC-116	Block numbers should not be used for time calculations.	Not Relevant
Signature Unique Id	SWC-117 SWC-121 SWC-122 EIP-155 EIP-712	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.	Passed
Shadowing State Variable	SWC-119	State variables should not be shadowed.	Passed
Weak Sources of Randomness	SWC-120	Random values should never be generated from Chain Attributes or be predictable.	Not Relevant
Incorrect Inheritance Order	SWC-125	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	EEA-Level-2 SWC-126	All external calls should be performed only to trusted addresses.	Passed
Presence of Unused Variables	SWC-131	The code should not contain unused variables if this is not justified by design.	Passed
EIP Standards Violation	EIP	EIP standards should not be violated.	Passed
Assets Integrity	**Custom**	Funds are protected and cannot be withdrawn without proper permissions or be locked on the contract.	Passed
User Balances Manipulation	**Custom**	Contract owners or any other third party should not be able to access funds belonging to users.	Passed
Data Consistency	**Custom**	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	**Custom**	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	**Custom**	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	**Custom**	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	**Custom**	Style guides and best practices should be followed.	Passed
Requirements Compliance	**Custom**	The code should be compliant with the requirements provided by the Customer.	Passed
Environment Consistency	**Custom**	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	**Custom**	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.	Not Relevant
Tests Coverage	**Custom**	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.	Passed
Stable Imports	**Custom**	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. Upgradeability Issues

The contract is upgradable but does not follow the upgradability best practices by not adding a gap in the contract storage.

This may lead to contract storage layout corruption during an upgrade.

The contract inherits EIP712Upgradeable that contains a __gap variable, but it is a best practice to create a new __gap variable that will be more accessible due to variables order.

Path: ./contracts/MainnetRFQ.sol

Recommendation: add a gap to the contract storage to allow future upgradability.

Found in: f8881f901e3680cdf281de7ef8e2812e4a89ec8d

Status: Fixed

(Revised commit: 4d650f9152b5c90a63a25f13c2a0176c2632526d) (__gap variable is added.)

Medium

No medium severity issues were found.

Low

L01. Inefficient Gas Model - Loop of Storage Interactions

In the batchClaimBalance() function, the variable *rebalancer* is read from storage in every loop iteration.

Accessing storage variables multiple times is not very Gas efficient.

Path: ./contracts/MainnetRFQ.sol : batchClaimBalance()

Recommendation: read *rebalancer* variable to memory and use the memory variable inside the while loop.

Found in: f8881f901e3680cdf281de7ef8e2812e4a89ec8d

Status: Fixed

(Revised commit: 4d650f9152b5c90a63a25f13c2a0176c2632526d) (rebalancer variable is now msg.sender and there is an access control modifier)

L02. 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.

Path: ./contracts/MainnetRFQ.sol : initialize(), addAdmin(), addTrustedContract()

Recommendation: implement zero address checks.

Found in: f8881f901e3680cdf281de7ef8e2812e4a89ec8d

Status: Fixed

(Revised commit: 4d650f9152b5c90a63a25f13c2a0176c2632526d) (Zero address checks are added)

L03. Functions that Can Be Declared External

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

Path: ./contracts/MainnetRFQ.sol : intialize()

Recommendation: use the external attribute for functions never called from the contract.

Found in: f8881f901e3680cdf281de7ef8e2812e4a89ec8d

Status: Fixed

(Revised commit: 4d650f9152b5c90a63a25f13c2a0176c2632526d) (initializer is declared external)

L04. Boolean Equality

Boolean constants can be used directly and do not need to be compared to true or false.

Path: ./contracts/MainnetRFQ.sol : simpleSwap(), claimBalance(), batchClaimBalance()

Recommendation: remove boolean equality.

Found in: f8881f901e3680cdf281de7ef8e2812e4a89ec8d

Status: Fixed

(Revised commit: e2cfd502dd25949661675f5f905f8506ae112477)

L05. Duplicate Code

The check if the caller is the rebalancer is repeated several times instead of being used in a modifier.

```
require(msg.sender == rebalancer, "RF-OCR-01");
```

Repeating require statements throughout the contract code can lead to unnecessary code duplication. This can make the codebase harder to maintain and more prone to errors.

Path: ./contracts/MainnetRFQ.sol : claimBalance(), batchClaimBalance(), receive()

Recommendation: use a modifier instead of repeating require statements. It will make code more maintainable, consistent and readable, while potentially improving Gas efficiency.

Found in: f8881f901e3680cdf281de7ef8e2812e4a89ec8d

Status: Fixed

(Revised commit: 4d650f9152b5c90a63a25f13c2a0176c2632526d) (access control is used and rebalancer is now the REBALANCER_ADMIN_ROLE role)

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.