SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Dexalot
Date: April 13th, 2022

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 Dexalot.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	Exchange; Portfolio; Fee; OrderBooks; TradePairs
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Git repository	https://github.com/Dexalot/contracts
Commit	42f91db4989dd31146854170bf64f04f20989b4d
Technical Documentation	YES
JS tests	YES
Website	https://dexalot.com/
Timeline	26 AUGUST 2021 - 07 APRIL 2022
Changelog	03 SEPTEMBER 2021 - Initial Audit 15 SEPTEMBER 2021 - Second Review 15 NOVEMBER 2021 - Third Review 18 FEBRUARY 2022 - Fourth Review 22 FEBRUARY 2022 - Fifth Review 16 MARCH 2022 - Sixth Review 23 MARCH 2022 - Seventh Review 04 APRIL 2022 - Eighths Review 07 APRIL 2022 - Ninth Review 10 APRIL 2022 - Tenth Review

Table of contents

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 contract and its code review conducted between August 25th, 2021 - September 3rd, 2021.

The second code review was conducted on September 15th, 2021.

The third code review was conducted on November 15th, 2021.

The fourth code review was conducted on February 18th, 2022.

The fifth review was conducted on February 22nd, 2022.

The sixth review was conducted on March 16th, 2022.

The seventh review was conducted on March 23rd, 2022.

The eighths review was conducted on April 4th, 2022.

The ninth review was conducted on April 7th, 2022.

The tenth review was conducted on April 13th, 2022.

Scope

The scope of the project is smart contracts in the repository:

Git repository:
    https://github.com/Dexalot/contracts
md5 hash:
    42f91db4989dd31146854170bf64f04f20989b4d
Technical Documentation: Yes
JS tests: Yes
Contracts:
    interfaces/IPortfolio.sol
    interfaces/ITradePairs.sol
    library/Bytes32Library.sol
    library/Bytes32LinkedListLibrary.sol
    library/RBTLibrary.sol
    library/StringLibrary.sol
    token/Airdrop.sol
    token/AirdropV1.sol
    token/DexalotToken.sol
    token/MockToken.sol
    token/Staking.sol
    token/TokenVesting.sol
    Exchange.sol
    OrderBooks.sol
    Portfolio.sol
    TradePairs.sol

We have scanned this smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that are considered:

Category	Check Item
Code review	▪ Reentrancy ▪ Ownership Takeover ▪ Timestamp Dependence ▪ Gas Limit and Loops ▪ DoS with (Unexpected) Throw ▪ DoS with Block Gas Limit ▪ Transaction-Ordering Dependence ▪ Style guide violation ▪ Costly Loop ▪ ERC20 API violation ▪ Unchecked external call ▪ Unchecked math ▪ Unsafe type inference ▪ Implicit visibility level ▪ Deployment Consistency ▪ Repository Consistency ▪ Data Consistency
Functional review	▪ Business Logics Review ▪ Functionality Checks ▪ Access Control & Authorization ▪ Escrow manipulation ▪ Token Supply manipulation ▪ Assets integrity ▪ User Balances manipulation ▪ Data Consistency manipulation ▪ Kill-Switch Mechanism ▪ Operation Trails & Event Generation

Executive Summary

According to the assessment, the Customer's smart contracts are well-secured.

Insecure
Poor secured
Secured
Well-secured
You are here

Our team analyzed code functionality, manual audit, and automated checks with Mythril and Slither. All issues found during automated analysis were manually reviewed, and important vulnerabilities are presented in the Audit overview section. All found issues can be found in the Audit overview section.

As a result of the audit, security engineers found **5** low severity issues.

After the second review, security engineers found that all issues were **resolved**.

After the third review, security engineers found that SafeERC20 and IERC20Metadata imports were replaced by their upgradeable versions, and the Fee contract was removed. No new issues were found.

After the fourth review, security engineers found that the scope was slightly updated with the token and vesting contracts and the Airdrop. However, only **1** medium and **1** low severity issues were found.

After the fifth review, security engineers found **1** medium severity issue.

After the sixth review, security engineers found that vesting was added to an Airdrop contract. **1** medium and **1** low severity issues was identified.

After the seventh review, security engineers found some changes, however, there is still **1** medium severity issue.

After the eighths review, security engineers found some changes, however, there is still **1** medium severity issue.

After the ninth review, security engineers found that the portfolio functionality was completely removed from the Airdrop contract, therefore **no security issues** were found.

After the tenth review, security engineers found that the staking contract was slightly updated, re-stake functionality was added as well as the emergency withdrawal function now leaves user-staked tokens on the contract, therefore **no security issues** were found.

NOTICE:

The Staking contract owner can withdraw all reward tokens at any time but always leaves the current staked token amount on the contract.

NOTICE 2:

The Airdrop contract owner can withdraw all tokens at any time.

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to assets loss or data manipulations.
High	High-level vulnerabilities are difficult to exploit; however, they also have a significant impact on smart contract execution, e.g., public access to crucial functions
Medium	Medium-level vulnerabilities are important to fix; however, they cannot lead to assets loss or data manipulations.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that cannot have a significant impact on execution

Audit overview

🟥🟥🟥🟥 Critical

No critical issues were found.

🟨🟨🟨 High

No high severity issues were found.

🟩🟩 Medium

1. Test coverage is low

   Test coverage is about 63% for statements and only 42% for branches which is really low.

   Scope: Tests

   Recommendation: Please make sure tests cover at least 95% of statements and up to 100% for code branches.

   Status: Fixed

2. Possible unexpected behavior

   In the case of calling “Airdrop.claim” function with passing “isFundingPortfolio” as **true**, it is unclear that a caller should give the allowance for the “Portfolio” contract to spend their tokens in the “unreleased” amount, which is unknown until “claim” execution.

   This part of logic is not straightforward and looks like a contract expecting that the sender will approve the “Portfolio” to spend the “uint256.MAX” amount of tokens.

   Contract: Airdrop.sol

   Function: claim

   Recommendation: We would recommend either adding a new function to the Portfolio contract allowing to pass the receiver, but transferring tokens from the sender, or at least checking the allowance in case of “isFundingPortfolio” at the beginning of the “claim” function to eliminate excess gas usages and improve the logic readability.

   Status: Fixed

🟦 Low

1. No events on setPortfolio function

   The function setPortfolio updates a critical contract value. Therefore, it should emit an event for better tracking off-chain.

   Recommendation: Please emit an event when changing the portfolio value.

   Status: Fixed

2. No events on setTradePairs function

   The function setTradePairs updates a critical contract value. Therefore, it should emit an event for better tracking off-chain.

   Recommendation: Please emit an event when changing the tradePairs value.

   Status: Fixed

3. Implicit state variable visibility

   When visibility is not explicitly declared, it is assumed to be internal. However, it could be unclear to reviewers.

   Recommendation: Please add an explicit visibility declaration.

   Status: Fixed

4. Reading state variable in the loop

   Calling length() method of the EnumerableSetUpgradeable for the state variable is burning gas.

   Recommendation: Please store the result of the length() call to the local variable and use it in the loop.

   Status: Fixed

5. Multiple access for the state variable

   Accessing the state variable in the function multiple times burns the gas.

   Recommendation: Please store the value of the state variable in the local variable.

   Status: Fixed

6. Too many digits

   Literals with many digits are difficult to read and review

   Contract: DexalotToken

   Function: constructor

   Recommendation: Please consider using scientific notation and ether unit suffix (i.e. *100e6* ***ether***).

   Status: The contract is already deployed.

7. Using time unit suffixes

   Solidity provides the time unit suffixes that make it easier to calculate time units.

   Contract: Staking

   Constant: SECONDSINYEAR

   Recommendation: Try using **365 days** instead of 31536000.

Status: Fixed

Conclusion

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools.

The audit report contains all found security vulnerabilities and other issues in the reviewed code.

As a result of the audit, security engineers found **5** low severity issues.

After the second review, security engineers found that all issues were **resolved**.

After the third review, security engineers found that SafeERC20 and IERC20Metadata imports were replaced by their upgradeable versions, and the Fee contract was removed. No new issues were found.

After the fourth review, security engineers found that the scope was slightly updated with the token and vesting contracts and the Airdrop. However, only **1** medium and **1** low severity issues were found.

After the fifth review, security engineers found **1** medium severity issue.

After the sixth review, security engineers found that vesting was added to an Airdrop contract. **1** medium and **1** low severity issues was identified.

After the seventh review, security engineers found some changes, however, there is still **1** medium severity issue.

After the eighths review, security engineers found some changes, however, there is still **1** medium severity issue.

After the ninth review, security engineers found that the portfolio functionality was completely removed from the Airdrop contract, therefore **no security issues** were found.

After the tenth review, security engineers found that the staking contract was slightly updated, re-stake functionality was added as well as the emergency withdrawal function now leaves user-staked tokens on the contract, therefore **no security issues** were found.

NOTICE:

The Staking contract owner can withdraw all reward tokens at any time but always leaves the current staked token amount on the contract.

NOTICE 2:

The Airdrop contract owner can withdraw all tokens at any time.

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed by the best industry practices at the date 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 audit makes no statements or warranties on the security of the code. It also cannot be considered a 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.

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 audit cannot guarantee the explicit security of the audited smart contracts.