



HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT



Customer: Formless

Date: April 04th, 2022

This document 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 containing confidential information can be used internally by the Customer, or it can be disclosed publicly after all vulnerabilities are fixed – upon a decision of the Customer.

Document

Name	Smart Contract Code Review and Security Analysis Report for Formless.
Approved By	Evgeniy Bezuglyi SC Department Head at Hacken OU
Type of Contracts	Proxy; Exchange
Platform	EVM
Language	Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Website	https://www.formless.me
Timeline	10.03.2022 - 04.04.2022
Changelog	18.03.2022 - Initial Review 04.04.2022 - Second Review



Table of contents

Introduction	4
Scope	4
Executive Summary	5
Severity Definitions	7
Findings	8
Recommendations	9
Disclaimers	10

Introduction

Hacken OÜ (Consultant) was contracted by Formless (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 is smart contracts in the repository:

Repository:

<https://github.com/FormLess-Games/Formless-Free-Market>

Commit:

9b8c359b75a01e3654393cee5b62d67ba1af1fbf

Technical Documentation: Yes (<https://formless.gitbook.io/white-paper-eng/>)

JS tests: No

Contracts:

WyvernProxyRegistry
WyvernTokenTransferProxy
WyvernExchange
WyvernExchangeMultiple

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	<ul style="list-style-type: none">▪ Reentrancy▪ Ownership Takeover▪ Timestamp Dependence▪ Gas Limit and Loops▪ Transaction-Ordering Dependence▪ Style guide violation▪ EIP standards violation▪ Unchecked external call▪ Unchecked math▪ Unsafe type inference▪ Implicit visibility level▪ Deployment Consistency▪ Repository Consistency
Functional review	<ul style="list-style-type: none">▪ Business Logics Review▪ Functionality Checks▪ Access Control & Authorization▪ Escrow manipulation▪ Token Supply manipulation▪ Assets integrity▪ User Balances manipulation▪ Data Consistency▪ Kill-Switch Mechanism

Executive Summary

The score measurements details can be found in the corresponding section of the [methodology](#).

Documentation quality

The Customer provided superficial functional requirements and no technical requirements. The total Documentation Quality score is **6** out of **10**.

Code quality

The total CodeQuality score is **6** out of **10**. Long lines. Some missed NatSpec. No comments in the code. No tests.

Architecture quality

The architecture quality score is **10** out of **10**. As far as we can see in our scope, the architecture looks very good.

Security score

As a result of the audit, security engineers found **1** low severity issue. The security score is **10** out of **10**. All found issues are displayed in the "Issues overview" section.

Summary

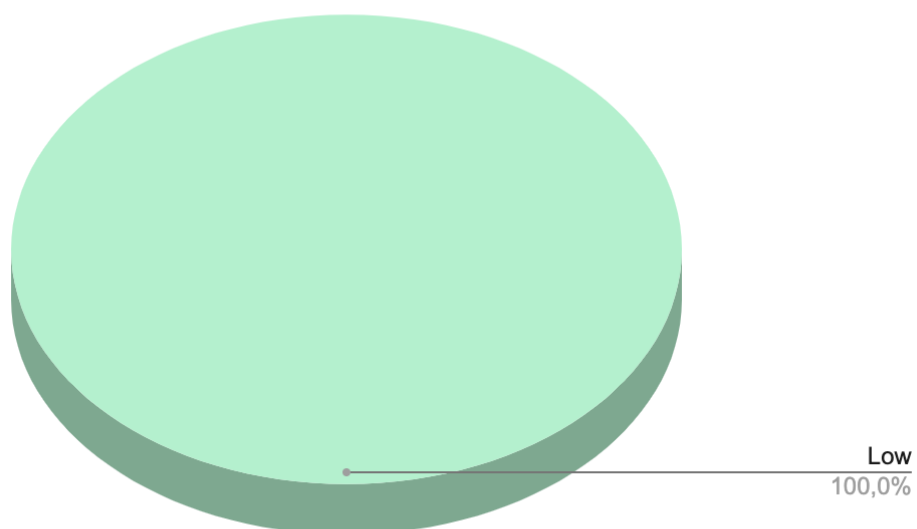
According to the assessment, the Customer's smart contract has the following score: **9.2**



Notices

1. The **entire** business logic was out of the current audit scope. There is a need for a deep analysis of the imported contracts.

Graph 1. The distribution of vulnerabilities after the audit.



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

Findings

■■■■ Critical

No critical severity issues were found.

■■■ High

No high severity issues were found.

■■ Medium

No medium severity issues were found.

■ Low

Conformance to Solidity naming conventions

Solidity defines a [naming convention](#) that should be followed

Contract: WyvernProxyRegistry.sol

Constant: name

Recommendation: follow the Solidity naming convention. Make constant name UPPERCASE (NAME)

Status: New

Recommendations

1. Please follow the [Solidity Style Guide](#) for naming, lines length, code layout, etc.
2. Make sure imported and extended contracts are also audited.
3. Write unit and integration tests and cover at least 95-100% of code branches.
4. Comment the code, especially for “magic numeric constants” like 14, 18, 5, etc.

Contract: WyvernExchangeMultiple.sol

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 it 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.