SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Prometeus Labs Ventures
Date: April 5th, 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 Prometeus Labs Ventures.
Approved By	Evgeniy Bezuglyi | SC Department Head at Hacken OU
Type of Contracts	ERC20 token; Staking
Platform	EVM
Language	Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Website	https://prometeus.io
Timeline	09.03.2022 - 05.04.2022
Changelog	15.03.2022 - Initial Review 04.04.2022 - Revising 05.03.2022 - Revising

Table of contents

Introduction

Hacken OÜ (Consultant) was contracted by Prometeus Labs Ventures (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/Prometeus-Network/takeus-contracts
Commit:
bc190ee239733b819d5a7976e875a4babf2a82f1
Technical Documentation: Yes:
https://doc.clickup.com/d/h/kj9eb-188/e305b75eb0fe384
JS tests: Yes:
https://github.com/Prometeus-Network/takeus-contracts/tree/main/test
Contracts:
SafeVault/accessors/SimulateTxAccessor.sol
SafeVault/base/Executor.sol
SafeVault/base/FallbackManager.sol
SafeVault/base/OwnerManager.sol
SafeVault/common/EtherPaymentFallback.sol
SafeVault/common/Enum.sol
SafeVault/common/SecuredTokenTransfer.sol
SafeVault/common/SelfAuthorized.sol
SafeVault/common/SignatureDecoder.sol
SafeVault/common/Singleton.sol
SafeVault/common/StorageAccessible.sol
SafeVault/external/GnosisSafeMath.sol
SafeVault/handler/CompatibilityFallbackHandler.sol
SafeVault/handler/DefaultCallbackHandler.sol
SafeVault/handler/HandlerContext.sol
SafeVault/proxies/GnosisSafeProxy.sol
SafeVault/proxies/IProxyCreationCallback.sol
SafeVault/proxies/SafeVaultProxyFactory.sol
SafeVault/libraries/CreateCall.sol
SafeVault/libraries/GnosisSafeStorage.sol
SafeVault/libraries/SignMessage.sol
SafeVault/libraries/MultiSend.sol
SafeVault/libraries/MultiSendCallOnly.sol
SafeVault/SafeVault.sol
VaultManager.sol
TakeUsMarketplace.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 ▪ 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	▪ 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 some functional requirements and a few technical requirements. However, the project is based on well-documented contracts. The total Documentation Quality score is **7** out of **10**.

Code quality

The total CodeQuality score is **7** out of **10**. Code duplications. Unit tests provided. The code is dirty. Hardcodes in the code.

Architecture quality

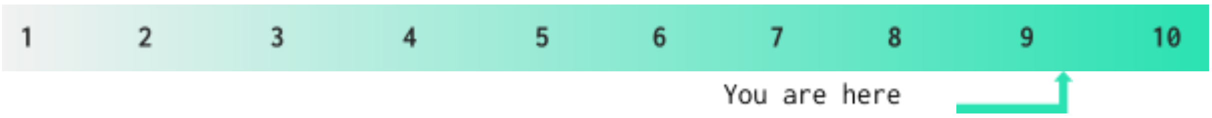
The architecture quality score is **8** out of **10**. The logic is split correctly into corresponding files. There is a repeating in the functionality of functions.

Security score

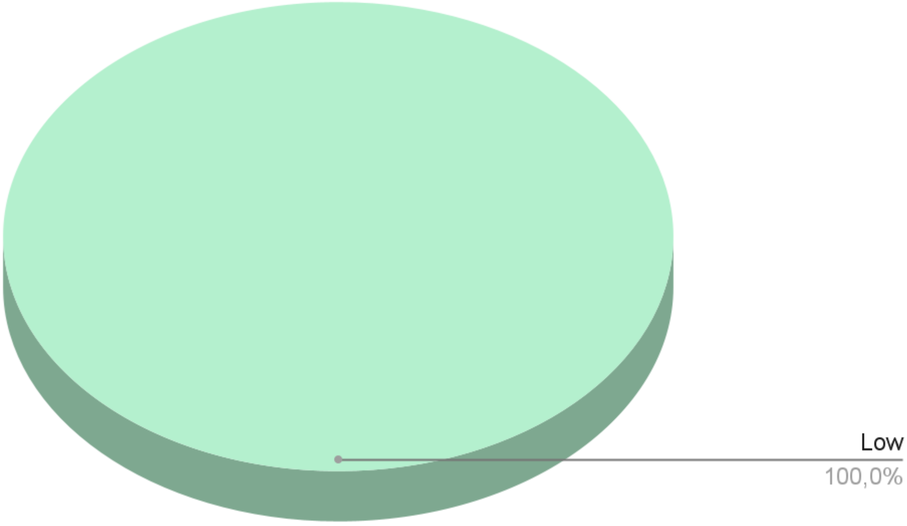
As a result of the audit, security engineers found **1** low severity issue. The security score is **10** out of **10**. All found issues are displayed in the "Issues overview" section.

Summary

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



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

1. **Incorrect balance checking**

   [redacted]

   [redacted]

   [redacted]

   Contract: [redacted]

   Functions: [redacted]

   Recommendation: [redacted]

   Status: Fixed (Revised Commit: bc190ee)

2. **An incorrect value used for a lender address**

   [redacted]

   [redacted]

   Contract: [redacted]

   Function: [redacted]

   Recommendation: [redacted]

   Status: Fixed (Revised Commit: 8aa4510)

🟨🟨🟨 High

1. **Tests failing**

   [redacted]

   Scope: testing

   Recommendation: [redacted]

   Status: Tests are successful when running one by one (Revised Commit: 8aa4510)

2. **Possible logic inconsistency**

[redacted]

Contract: [redacted]

Functions: [redacted]

Recommendation: [redacted]

Status: Acknowledged. The customer says it should be that way. (Revised Commit: bc190ee)

🟩🟩 Medium

1. Contracts that lock Ether

[redacted]

Contract: [redacted]

Functions: [redacted]

Recommendation: [redacted]

Status: Added a withdrawal function (Revised Commit: 8aa4510)

🟦 Low

1. No events emitted

[redacted]

Contract: [redacted]

Functions: [redacted]

Recommendation: [redacted]

Status: Fixed (Revised Commit: 8aa4510)

2. Using of time unit suffixes

[redacted]

Contract: [redacted]

Functions: [redacted]

Recommendation: [redacted]

Status: Fixed (Revised Commit: 8aa4510)

3. Duplicated logic

[redacted]

Contract: [redacted]

Functions: [redacted]

Recommendation: [redacted]

Status: Partly Fixed (Revised Commit: bc190ee)

4. Implicit visibility declaration

[redacted]

Contract: [redacted]

Constants: [redacted]

Recommendation: [redacted]

Status: Fixed (Revised Commit: bc190ee)

5. Hardcoded address declaration

[redacted]

Contract: [redacted]

Constant: [redacted]

Recommendation: [redacted]

Status: Will not Fix (Revised Commit: bc190ee)

6. Duplicated code

[redacted]

Contract: [redacted]

Functions: [redacted]

Recommendation: optimize the code to remove duplications.

Status: Partly fixed (Revised Commit: bc190ee)

7. Duplicated code

[redacted]

Contract: [redacted]

Functions: [redacted]

Recommendation: [redacted]

Status: Partly fixed (Revised Commit: bc190ee)

Recommendations

1. Revise the logic of the TakeUsMarketplace.
2. The logic of the SafeVault.checkIfNFTisLocked could be cleared.
3. Cover code by unit and integration tests.

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.