HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: DAOVentures
Date: September 27th, 2021

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 DAOVentures.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	Vault, Strategy
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/daoventures/dao-layer2-vaults/blob/MVF/contracts/
Commit	cbdb449472b49c37552c4f83c8a248ea05ece05a - MVFVault.sol 60d975a4e9f5d39f86cda71b0b468904451f35db - MVFStrategy.sol
Technical Documentation	NO
JS tests	NO
Timeline	16 SEPTEMBER 2021 – 27 SEPTEMBER 2021
Changelog	27 SEPTEMBER 2021 - Initial Audit

Table of contents

Introduction

Hacken OÜ (Consultant) was contracted by DAOVentures (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 September 16th, 2021 - September 27th, 2021.

Scope

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

Repository:

https://github.com/daoventures/dao-layer2-vaults/blob/MVF/contracts/

Commit:

cbdb449472b49c37552c4f83c8a248ea05ece05a – MVFVault.sol

60d975a4e9f5d39f86cda71b0b468904451f35db – MVFStrategy.sol

Technical Documentation: No

JS tests: No

Contracts:

MVFVault.sol

MVFStrategy.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 performed an analysis of 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 **3** low severity issues.

Graph 1. The distribution of vulnerabilities after the audit.

Low
100,0%
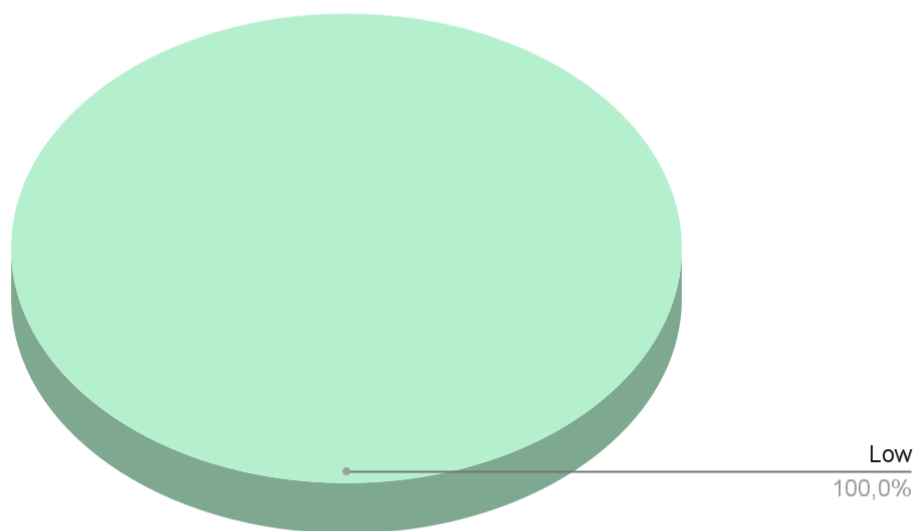
Severity Definitions

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 can't lead to assets loss or data manipulations.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that can't have a significant impact on execution

Audit overview

▪ ▪ ▪ ▪ Critical

No critical issues were found.

▪ ▪ ▪ High

No high severity issues were found.

▪ ▪ Medium

No medium severity issues were found.

▪ Low

1. Missing zero address validation

   _treasuryWallet, _communityWallet, _admin, _strategist, and _biconomy
   aren't validated as being zero-addresses in the initializer. That
   could lead to the contract's failure.

   Recommendation: Please consider adding zero-address validation.

Lines: MVFVault.sol#87

```
function initialize(
   string calldata name, string calldata ticker,
   address _treasuryWallet, address _communityWallet, address _admin,
address _strategist,
   address _biconomy, address _strategy
) external initializer {
   __ERC20_init(name, ticker);
   __Ownable_init();

   strategy = IStrategy(_strategy);

   treasuryWallet = _treasuryWallet;
   communityWallet = _communityWallet;
   admin = _admin;
   strategist = _strategist;
   trustedForwarder = _biconomy;
```

2. A public function that could be declared external

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

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

Lines: MVFStrategy.sol#399

```
function collectProfitAndUpdateWatermark() public onlyVault returns (uint
```

```
fee) {                          www.hacken.io
```

Lines: MVFVault.sol#175

```
function invest() public whenNotPaused {
```

3. State variables that could be declared constant

   Constant state variables should be declared constant to save gas.

   Recommendation: Add the constant attributes to state variables that never change or provide methods to change variables.

Lines: MVFStrategy.sol#146

```
address public treasuryWallet;
address public communityWallet;
address public strategist;
address public admin;
```

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 **3** low severity issues.

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed in accordance with the best industry practices at the date of this report, in relation to 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 as a sufficient assessment regarding the utility and safety of the code, bug-free status, or any other statements of the contract. 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 can't guarantee the explicit security of the audited smart contracts.