



VIRTUAL CLAIM

LIMITORDER

Smart Contract Review

Deliverable: Smart Contract Audit Report

Security Assessment
February 2025

Disclaimer

The information and views set out in this publication are those of the author(s) and do not necessarily reflect the official opinion of the Company. The content, conclusions, and recommendations set out in this publication are elaborated in the specific for only project.

Virtual Caim does not guarantee the authenticity of the project or organization or team of members that are connected/owner behind the project nor the accuracy of the data included in this study. All representations, warranties, undertakings, and guarantees relating to the report are excluded, particularly concerning – but not limited to – the qualities of the assessed projects and products. Neither the Company nor any person acting on the Company's behalf may be held responsible for the use that may be made of the information contained herein.

Virtual Caim retains the right to display audit reports and other content elements as examples of their work in their portfolio and as content features in other projects with protecting all security purposes of customers. 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.

©Virtual Caim Private Limited, 2025.

Smart Contract Audit

Report Summary

Title	LIMITORDER Smart Contract Audit		
Project Owner	Abelpay		
Classification	Public		
Reviewed by	Virtual Caim Private Limited	Review date	13/02/2025
Approved by	Virtual Caim Private Limited	Approval date	13/02/2025
		Nº Pages	25

Smart Contract Audit

Overview

Background

Abelpay's team requested Virtual Caim to perform an Extensive Smart Contract Audit of their 'LIMITORDER' Smart Contract.

Project Dates

The following is the project schedule for this review and report:

- **February 10:** Smart Contract Review Started (*Completed*)
- **February 13:** Initial Delivery of Audit Findings (*Completed*)
- **February 13:** Final Delivery of Audit Report (*Completed*)

Coverage

Target Specification and Revision

For this audit, we performed the project's basic research, investigation by discussing the details with the project owner and developer and then reviewed the smart contracts of Abelpay.

The following documentation & repositories were considered in -scope for the review:

LIMITORDER Smart Contract	 limit-order_updated.txt
----------------------------------	---

Smart Contract Audit

Introduction

Given the opportunity to review LIMITORDER's Contracts related smart contract source code, we in the report summary our methodical approach to evaluate all potential common security issues in the smart contract implementation, expose possible semantic irregularities between smart contract code and design document, and provide additional suggestions or recommendations for improvement. Our results show that the given version of smart contracts is ready to be used after resolving the mentioned issues and done functional testing by owner/developer themselves, as there might be issues related to business logic, security or performance which only can found/understand by them.

About Audit

Item	Description
Issuer	LIMITORDER
Website	pro-trader.app
Language	Solidity
Audit Test Method	Whitebox Testing
Latest Audit Report	February 13, 2025

Smart Contract Audit

Token Overview:

Item	Description
Fees	0-0%
Fee Privilege	Owner
Ownership	Owned
Minting	None
Max Tx	No

Smart Contract Audit

Information about Test Methods

Test Methods

Black-box testing

Conduct security tests from an attacker's perspective externally.

Grey-box testing

Conduct security testing on code modules through the scripting tool, observing the internal running status, mining weaknesses.

White-box testing

Based on the open-source code, non-open-source code, to detect whether there are vulnerabilities in programs such as nodes, SDK, etc.

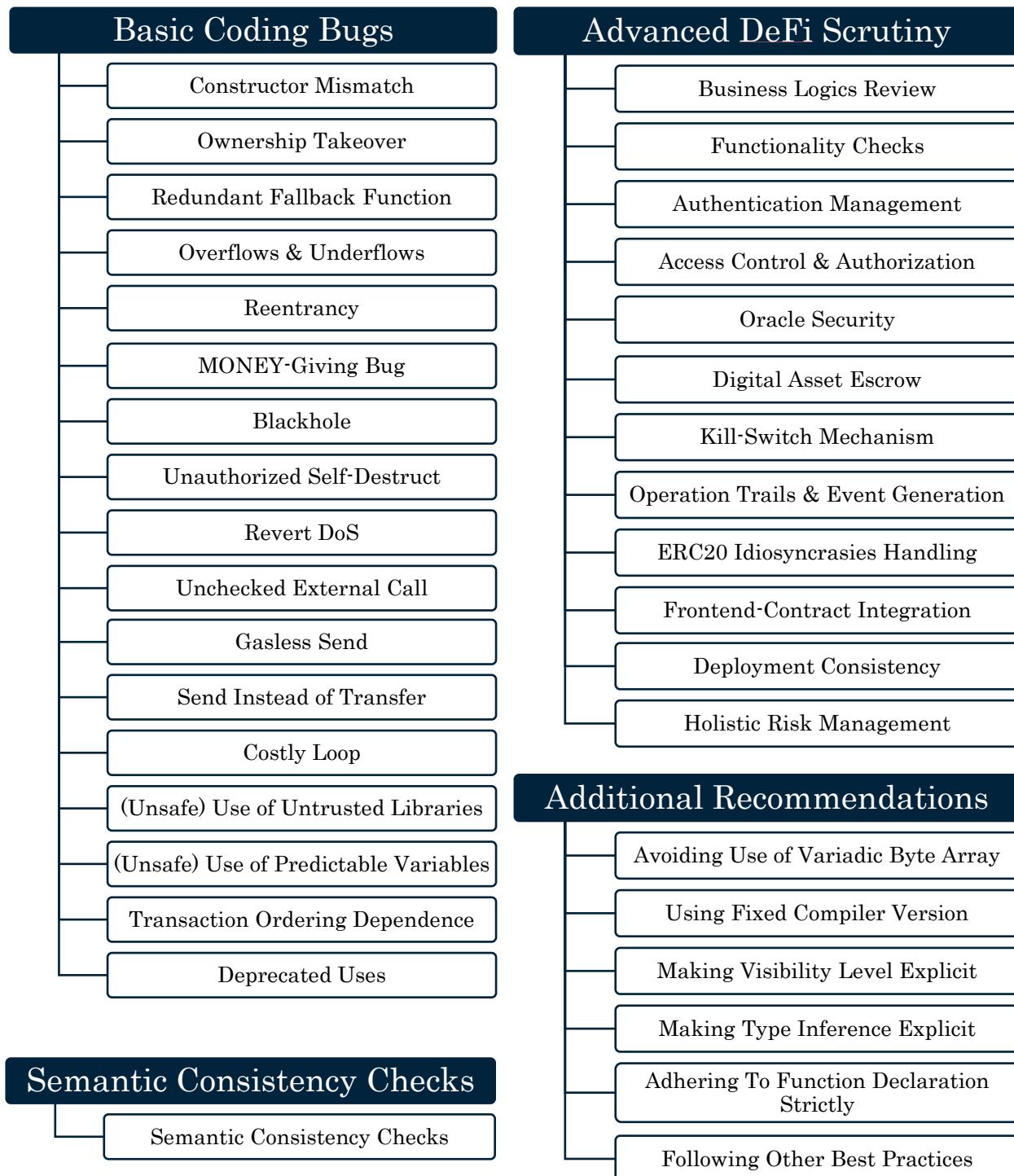
Smart Contract Audit

Vulnerability Severity Level Information

Level	Description
Critical	Critical severity vulnerabilities will have a significant effect on the security of the DeFi project, and it is strongly recommended to fix the critical vulnerabilities.
High	High severity vulnerabilities will affect the normal operation of the DeFi project. It is strongly recommended to fix high-risk vulnerabilities.
Medium	Medium severity vulnerability will affect the operation of the DeFi project. It is recommended to fix medium-risk vulnerabilities.
Low	Low severity vulnerabilities may affect the operation of the DeFi project in certain scenarios. It is suggested that the project party should evaluate and consider whether these vulnerabilities need to be fixed.
Weakness	There are safety risks theoretically, but it is extremely difficult to reproduce in engineering.

Smart Contract Audit

List of Check Items

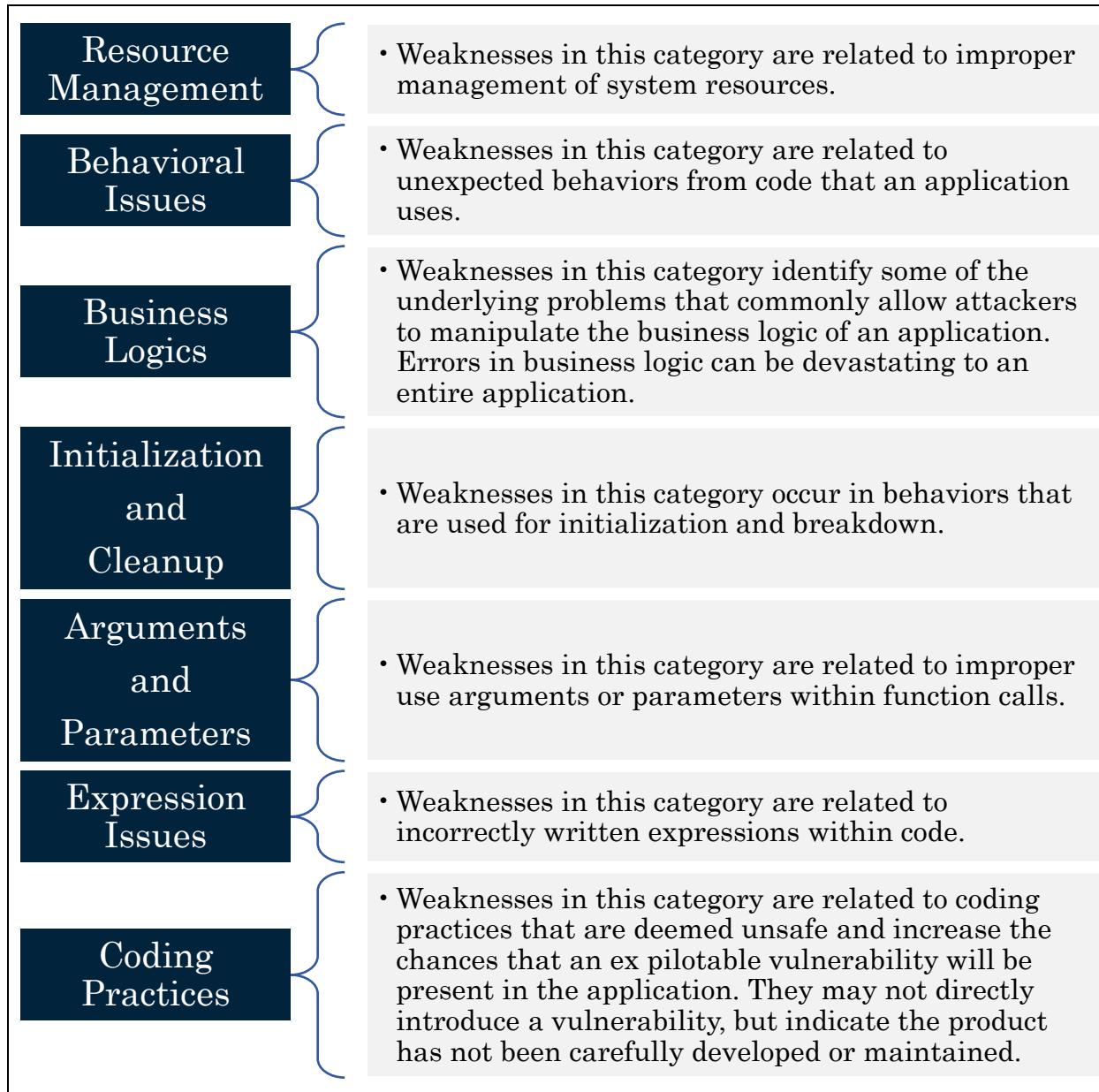


Smart Contract Audit

Common Weakness Enumeration (CWE) Classifications Used in this Audit.

Configuration	<ul style="list-style-type: none">Weaknesses in this category are typically introduced during the configuration of the software.
Data Processing Issues	<ul style="list-style-type: none">Weaknesses in this category are typically found in functionality that processes data.
Numeric Errors	<ul style="list-style-type: none">Weaknesses in this category are related to improper calculation or conversion of numbers.
Security Features	<ul style="list-style-type: none">Weaknesses in this category are concerned with topics like authentication, access control, confidentiality, cryptography, and privilege management. (Software security is not security software.)
Time and State	<ul style="list-style-type: none">Weaknesses in this category are related to the improper management of time and state in an environment that supports simultaneous or near-simultaneous computation by multiple systems, processes, or threads.
Error Conditions, Return Values, Status Codes	<ul style="list-style-type: none">Weaknesses in this category include weaknesses that occur if a function does not generate the correct return/status code, or if the application does not handle all possible return/status codes that could be generated by a function.

Smart Contract Audit



Smart Contract Audit

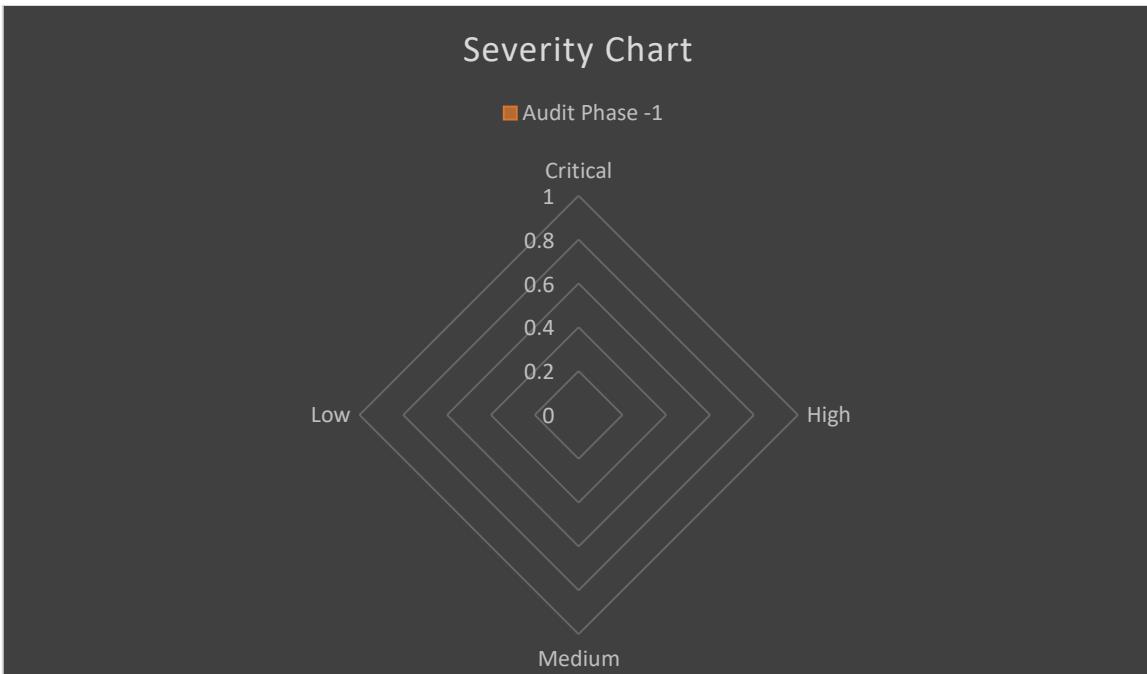
Findings

Summary

Here is a summary of our findings after scrutinizing the LIMITORDER Smart Contract Review. During the first phase of our audit, we studied the smart contract source code and ran our in-house static code analyzer through the Specific tools. The purpose here is to statically identify known coding bugs, and then manually verify (reject or confirm) issues reported by tools. We further manually review business logics, examine system operations, and place DeFi-related aspects under scrutiny to uncover possible pitfalls and/or bugs.

Severity	No. of Issues	Current Status
Critical	0	-
High	0	0
Medium	0	0
Low	2	0
Total	0 (Currently Open Issues)	

Smart Contract Audit



We have so far identified that there are potential issues with severity of **0 Critical**, **0 High**, **0 Medium**, and **0 Low**. Overall, these smart contracts are well-designed and engineered.

Smart Contract Audit

Functional Overview

<code>(\\$)</code> = payable function	[Pub] public
<code>#</code> = non-constant function	[Ext] external
	[Prv] private
	[Int] internal

+ LimitOrder (Ownable, ReentrancyGuard)

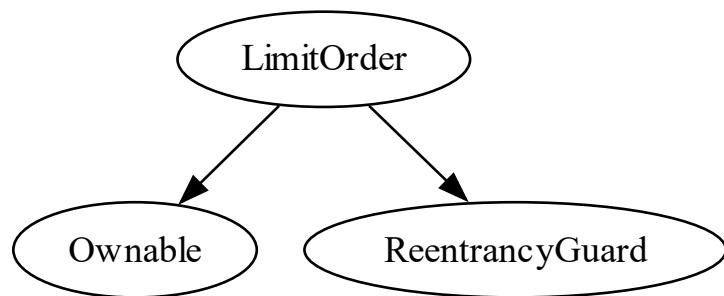
- [Pub] <Constructor> #
- [Pub] createOrder (`$`)
 - modifiers: nonReentrant
- [Pub] editOrder (`$`)
 - modifiers: nonReentrant
- [Pub] cancelOrder (`$`)
 - modifiers: nonReentrant
- [Pub] executeOrder #
 - modifiers: onlyOperator
- [Pub] executeMultipleOrder #
 - modifiers: onlyOperator
- [Pub] setOperator #
 - modifiers: onlyOwner
- [Pub] setOperatorFee #
 - modifiers: onlyOwner
- [Pub] setWhitelist #

Smart Contract Audit

- modifiers: onlyOwner
- [Pub] toggleEnableWhitelist #
 - modifiers: onlyOwner
- [Pub] getOrderLength
- [Pub] getOrders
- [Ext] <Receive Ether> (\$)

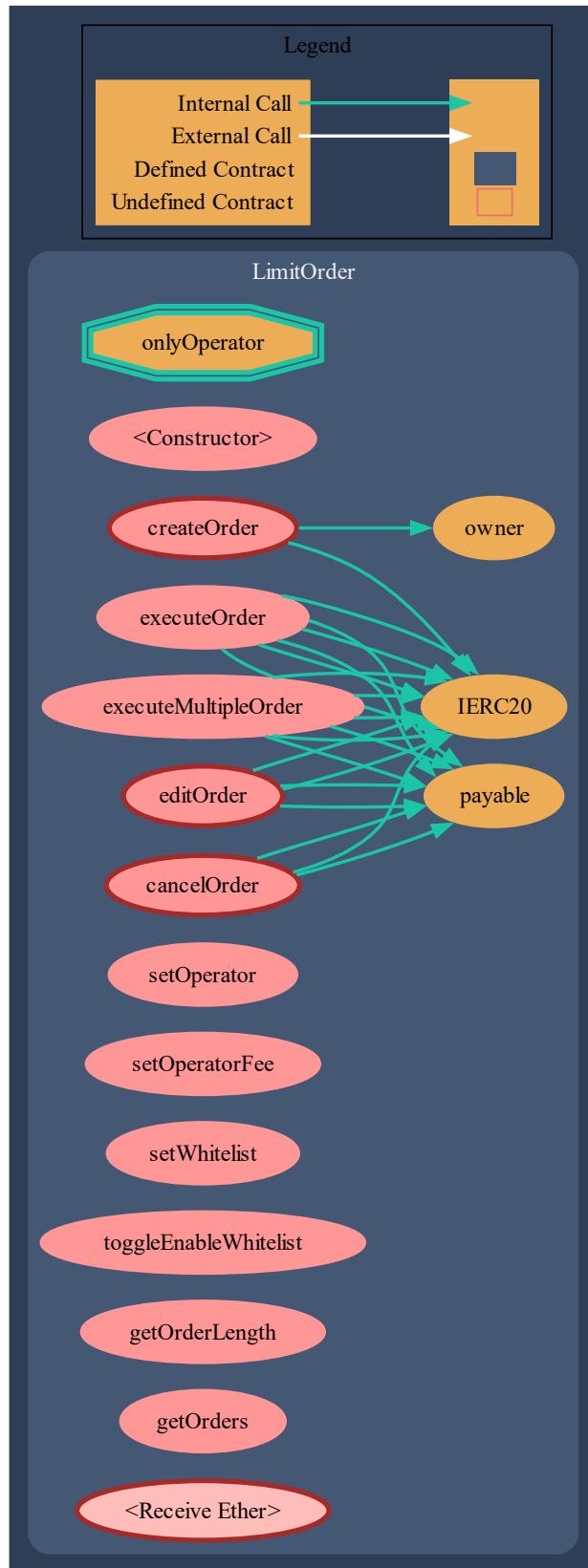
Smart Contract Audit

Inheritance Tree



Smart Contract Audit

Graph for LIMITORDER



Smart Contract Audit

Detailed Results

Ownership Privileges

- The owner can create/edit an order.
- The owner can execute an order.
- The owner can execute multiple orders.
- The owner set the operator's address.
- The owner sets the operator fees.
- The owner can whitelist addresses.
- The owner can toggle enable whitelist.

Smart Contract Audit

Issues Checking Status

1. Centralization – Floating pragma solidity

- Severity: Low
- Overview: Adding the constant version of solidity is recommended, as this prevents the unintentional deployment of a contract with an outdated compiler that contains unresolved bugs.
- Status: Close
- POC:

```
pragma solidity ^0.8.0;
```

- Issue Resolved: After the initial findings raised by the auditors, developers have resolved this issue.

```
pragma solidity 0.8.26;
```

Smart Contract Audit

2. Centralization – Missing events arithmetic

- Severity: Low
- Overview: Adding a constant version of solidity is recommended, as this prevents the unintentional deployment of a contract with an outdated compiler that contains unresolved bugs.
- Status: Open

```
function setOperator(address _new) public onlyOwner {
    operator = _new;
}

function setOperatorFee(uint256 _new) public onlyOwner {
    operatorFee = _new;
}
```

- Issue Resolved: After the initial findings raised by the auditors, developers have resolved this issue.

```
function setOperator(address _new) public onlyOwner {
    operator = _new;
    emit OperatorChanged(operator);
}

function setOperatorFee(uint256 _new) public onlyOwner {
    operatorFee = _new;
    emit OperatorFeeChanged(operatorFee);
}
```

Smart Contract Audit

Automated Tool Results

Slither: -

NA

Smart Contract Audit

Basic Coding Bugs

No.	Name	Description	Severity	Result
1.	Constructor Mismatch	Whether the contract name and its constructor are not identical to each other.	Critical	PASSED
2.	Ownership Takeover	Whether the set owner function is not protected.	Critical	PASSED
3.	Redundant Fallback Function	Whether the contract has a redundant fallback function.	Critical	PASSED
4.	Overflows & Underflows	Whether the contract has general overflow or underflow vulnerabilities	Critical	PASSED
5.	Reentrancy	Reentrancy is an issue when code can call back into your contract and change state, such as withdrawing ETHs	High	PASSED
6.	MONEY-Giving Bug	Whether the contract returns funds to an arbitrary address	High	PASSED
7.	Blackhole	Whether the contract locks ETH indefinitely: merely in without out	High	PASSED
8.	Unauthorized Self-Destruct	Whether the contract can be killed by any arbitrary address	Medium	PASSED
9.	Revert DoS	Whether the contract is vulnerable to DoS attack because	Medium	PASSED

Smart Contract Audit

		of unexpected revert		
10.	Unchecked External Call	Whether the contract has any external call without checking the return value	Medium	PASSED
11.	Gasless Send	Whether the contract is vulnerable to gasless send	Medium	PASSED
12.	Send Instead of Transfer	Whether the contract uses send instead of transfer	Medium	PASSED
13.	Costly Loop	Whether the contract has any costly loop which may lead to Out-Of-Gas exception	Medium	PASSED
14.	(Unsafe) Use of Untrusted Libraries	Whether the contract use any suspicious libraries	Medium	PASSED
15.	(Unsafe) Use of Predictable Variables	Whether the contract contains any randomness variable, but its value can be predicated	Medium	PASSED
16.	Transaction Ordering Dependence	Whether the final state of the contract depends on the order of the transactions	Medium	PASSED
17.	Deprecated Uses	Whether the contract use the deprecated tx.origin to perform the authorization	Medium	PASSED
18.	Semantic Consistency Checks	Whether the semantic of the white paper is different from the implementation of the contract	Critical	PASSED

Smart Contract Audit

Conclusion

In this audit, we thoroughly analyzed Abelpay's 'LIMITORDER' Smart Contract. The current code base is well organized, and low-level issues were resolved by the developer in this phase of testing of Smart Contract.

Meanwhile, we need to call attention to the fact that smart contracts are still in an early but exciting stage of development. To improve this report, we greatly appreciate any constructive feedback or suggestions on our methodology, audit findings, or potential gaps in scope/coverage.

Smart Contract Audit

About Virtual Caim

Just like our other parallel journey at eNebula Solution, we believe that people have a fundamental need for security and that the use of secure solutions enables every person to use the Internet and every other connected technology more freely. We aim to provide security consulting services to help others make their solutions more resistant to unauthorized access to data & inadvertent manipulation of the system. We support teams from the design phase through production to launch and surely after.

The Virtual Caim is specifically incorporated to handle all kind of Security related operations, our Highly Qualified and Certified security team has skills for reviewing coding languages like Solidity, Rust, Go, Python, Haskell, C, C++, and JavaScript for common security vulnerabilities & specific attack vectors. The team has been reviewing the implementations of cryptographic protocols and distributed system architecture, including in cryptocurrency, blockchains, payments, and smart contracts. Additionally, the team can utilize various tools to scan code & networks and build custom tools as necessary.

Although we are a small team, we surely believe that we can have a momentous impact on the world by being translucent & open about the work we do.

For more information about our other security services and consulting,
please visit -- <https://virtualcaim.com/>
& Mail us at – audit@virtualcaim.com