HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Libertify
Date: July 14, 2023

This report 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 can be disclosed publicly after prior consent by another Party. Any subsequent publication of this report shall be without mandatory consent.

Document

Name	Smart Contract Code Review and Security Analysis Report for Libertify
Approved By	Marcin Ugarenko | Lead Solidity SC Auditor at Hacken OU
Type	ERC4626
Platform	EVM
Language	Solidity
Methodology	Link
Website	https://www.libertify.com/
Changelog	01.03.2023 - Initial Review 09.03.2023 - Second Review 13.03.2023 - Third Review 22.03.2023 - Fourth Review 14.07.2023 - Fifth Review

Table of contents

Introduction

Hacken OÜ (Consultant) was contracted by Libertify (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:

Initial review scope

Repository	https://github.com/LibertyFi/libertify.protocol
Commit	03740ef3953595e268cb55b34166c58398f14a15
Whitepaper	Link
Functional Requirements	Attached for the audit
Technical Requirements	Attached for the audit
Contracts	File: ./contracts/interfaces/ILibertiPriceFeed.sol SHA3: d743650563f21375b23fa423aa7df789c5c3495bd89ebec4e4a57cde06b62c1b File: ./contracts/interfaces/ILibertiVault.sol SHA3: 1533efa055deef53c0384305213bef63b3a726280767a7e5bfc76014c60b7ca7 File: ./contracts/interfaces/ISanctionsList.sol SHA3: 8649ff7d3b46dbe55aad77651dbb3f7a0e7dbe9d7724bd1d939a6ee3be8ff1c8 File: ./contracts/LibertiFactory.sol SHA3: 0526e715d802174b4f8622b0320bf5f0beb432ed38ed2c9a8738adbf8b263e03 File: ./contracts/LibertiPriceFeed.sol SHA3: 22d88050f1fbd08b93f93ed6fee487dd1e3c411704875ba167c1ac870bd56d9b File: ./contracts/LibertiSwap.sol SHA3: effb06f54830d0f98829952c390a9b0c4b7a1cc6cb9145c9bce224ad83a7fe37 File: ./contracts/LibertiVault.sol SHA3: 8de6fb44771bc1ec54410ea97225c8d4194a7158dcf91ad664abcc679c3ed876

Second review scope

Repository	https://github.com/LibertyFi/libertify.protocol
Commit	2ec8b102e4b34e719c21a5c42358bcfcdc24f3b4
Whitepaper	Link

Functional Requirements	Attached for the audit
Technical Requirements	Attached for the audit
Contracts	File: ./contracts/interfaces/ILibertiPriceFeed.sol SHA3: 3f89c2c1a730197f3eee45702db651e5968f5d55e4c7d44d5ea0411af482e6e9 File: ./contracts/interfaces/ILibertiVault.sol SHA3: 3597b88e3412c7b6491b25151dc065c30d69216453fa80c3fac714e9d6ad0369 File: ./contracts/interfaces/ISanctionsList.sol SHA3: c3cf31fd3ac4401bcfa8c99851bc2e843f64ecfc35674cc92846df10da5471d2 File: ./contracts/LibertiAggregationRouterV4.sol SHA3: 797bb7e91627f175492ae91c10ec24d142de4e92440fb937a310dc403763bc2e File: ./contracts/LibertiFactory.sol SHA3: 3f9dc5f3a394d94820505d8b25177ff53e9d000924dd58e6cb05718feb3e4b9a File: ./contracts/LibertiFactoryBase.sol SHA3: 1e9e9479ad8782726472063ce444acd13f2d2c3afa4970691d1648d41b3f1974 File: ./contracts/LibertiPriceFeed.sol SHA3: 9598adb421e2d6ec07233d1cd6462276f3471be3f6f51050ea49233c4c80bab0 File: ./contracts/LibertiSwap.sol SHA3: effb06f54830d0f98829952c390a9b0c4b7a1cc6cb9145c9bce224ad83a7fe37 File: ./contracts/LibertiVault.sol SHA3: 9d07fe78562a7cbc8dc7cdc7922d7ab0045ff27da0388f327683731b32f88ca5

Third review scope

Repository	https://github.com/LibertyFi/libertify.protocol
Commit	595bf178d47cd79cbdb146bb08170f88555931f4
Whitepaper	Link
Functional Requirements	Attached for the audit
Technical Requirements	Attached for the audit
Contracts Addresses	https://polygonscan.com/address/0x3220de3865b30c641206fc9ff6de3a49960a92b9#code
Contracts	File: ./contracts/interfaces/ILibertiPriceFeed.sol SHA3: 3f89c2c1a730197f3eee45702db651e5968f5d55e4c7d44d5ea0411af482e6e9 File: ./contracts/interfaces/ILibertiVault.sol SHA3: 3597b88e3412c7b6491b25151dc065c30d69216453fa80c3fac714e9d6ad0369

	File: ./contracts/interfaces/ISanctionsList.sol SHA3: c3cf31fd3ac4401bcfa8c99851bc2e843f64ecfc35674cc92846df10da5471d2 File: ./contracts/LibertiAggregationRouterV4.sol SHA3: 797bb7e91627f175492ae91c10ec24d142de4e92440fb937a310dc403763bc2e File: ./contracts/LibertiFactory.sol SHA3: 3f9dc5f3a394d94820505d8b25177ff53e9d000924dd58e6cb05718feb3e4b9a File: ./contracts/LibertiFactoryBase.sol SHA3: 1e9e9479ad8782726472063ce444acd13f2d2c3afa4970691d1648d41b3f1974 File: ./contracts/LibertiPriceFeed.sol SHA3: 9598adb421e2d6ec07233d1cd6462276f3471be3f6f51050ea49233c4c80bab0 File: ./contracts/LibertiVault.sol SHA3: 713399186a0284b5f5661cd46378928a511d94b20a82f29797df007716360aae

Fourth review scope

Repository	https://github.com/LibertyFi/libertify.protocol
Commit	5ff2415db5b9a7281f1bc57c68b715a2997010b8
Whitepaper	Link
Functional Requirements	Attached for the audit
Technical Requirements	Attached for the audit
Contracts Addresses	Not provided.
Contracts	File: ./contracts/interfaces/ILibertiPriceFeed.sol SHA3: 3f89c2c1a730197f3eee45702db651e5968f5d55e4c7d44d5ea0411af482e6e9 File: ./contracts/interfaces/ILibertiVault.sol SHA3: 3597b88e3412c7b6491b25151dc065c30d69216453fa80c3fac714e9d6ad0369 File: ./contracts/interfaces/ISanctionsList.sol SHA3: c3cf31fd3ac4401bcfa8c99851bc2e843f64ecfc35674cc92846df10da5471d2 File: ./contracts/LibertiAggregationRouterV4.sol SHA3: 797bb7e91627f175492ae91c10ec24d142de4e92440fb937a310dc403763bc2e File: ./contracts/LibertiFactory.sol SHA3: 3f9dc5f3a394d94820505d8b25177ff53e9d000924dd58e6cb05718feb3e4b9a File: ./contracts/LibertiFactoryBase.sol SHA3: 1e9e9479ad8782726472063ce444acd13f2d2c3afa4970691d1648d41b3f1974 File: ./contracts/LibertiPriceFeed.sol SHA3: 9598adb421e2d6ec07233d1cd6462276f3471be3f6f51050ea49233c4c80bab0 File: ./contracts/LibertiVault.sol SHA3: a08a8027df4af2fc1d651c67596f993503b67c667dafe47a147e34326fca5967

Fifth review scope

Repository	https://github.com/LibertyFi/libertify.protocol
Commit	3b1a3ce2896c414c8bd1ba5503b25db48b33642a
Whitepaper	Link
Functional Requirements	Attached for the audit
Technical Requirements	Attached for the audit
Contracts Addresses	Not provided.
Contracts	File: ./contracts/interfaces/ILibertiPriceFeed.sol SHA3: 3f89c2c1a730197f3eee45702db651e5968f5d55e4c7d44d5ea0411af482e6e9 File: ./contracts/interfaces/ILibertiVault.sol SHA3: 3597b88e3412c7b6491b25151dc065c30d69216453fa80c3fac714e9d6ad0369 File: ./contracts/interfaces/ISanctionsList.sol SHA3: c3cf31fd3ac4401bcfa8c99851bc2e843f64ecfc35674cc92846df10da5471d2 File: ./contracts/LibertiAggregationRouterV4.sol SHA3: d7bf1dcc56016b90c5267781464df1fe0936797d2f3c9b60f0e9ddbf53132dc5 File: ./contracts/LibertiFactory.sol SHA3: 3f9dc5f3a394d94820505d8b25177ff53e9d000924dd58e6cb05718feb3e4b9a File: ./contracts/LibertiFactoryBase.sol SHA3: 1e9e9479ad8782726472063ce444acd13f2d2c3afa4970691d1648d41b3f1974 File: ./contracts/LibertiPriceFeed.sol SHA3: 9598adb421e2d6ec07233d1cd6462276f3471be3f6f51050ea49233c4c80bab0 File: ./contracts/LibertiVault.sol SHA3: 57b3c2fd14e6eb75ed6cb49650561e2d8c51e4c1c016da79530475267bc5c842

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to the loss of user funds or contract state manipulation by external or internal actors.
High	High vulnerabilities are usually harder to exploit, requiring specific conditions, or have a more limited scope, but can still lead to the loss of user funds or contract state manipulation by external or internal actors.
Medium	Medium vulnerabilities are usually limited to state manipulations but cannot lead to asset loss. Major deviations from best practices are also in this category.
Low	Low vulnerabilities are related to outdated and unused code or minor Gas optimization. These issues won't have a significant impact on code execution but affect code quality

Executive Summary

The score measurement details can be found in the corresponding section of the scoring methodology.

Documentation quality

The total Documentation Quality score is **9** out of **10**.

- Provided documentation well describes both technical and functional parts of the system.
- Missing NatSpecs for multiple functions.

Code quality

The total Code Quality score is **10** out of **10**.

- Code is well-written and designed.

Test coverage

Code coverage of the project is **98%** (branch coverage).

- Deployment and basic user interactions are covered with tests.
- Negative case coverage is present.

Security score

As a result of the audit, the code contains **1** low severity issue. The security score is **10** out of **10**.

All found issues are displayed in the “Findings” section.

Summary

According to the assessment, the Customer's smart contract has the following score: **9.8**.

1
2
3
4
5
6
7
8
9
10
The final score

Table. The distribution of issues during the audit

Review date	Low	Medium	High	Critical
01 March 2023	6	8	2	1
09 March 2023	1	2	1	0
13 March 2023	1	0	0	0
22 March 2023	1	0	0	0

14 July 2023	1	0	0	0

System Overview

Libertify is a decentralized investment platform that uses a concept of tokenized vaults with a fixed asset ratio. The main idea is to decrease risks, related to holding assets, using vault rebalancing.

The files in the scope:

- LibertiAggregationRouterV4.sol - contains a logic to execute a swap using 1Inch Aggregation Router V4 protocol.
- LibertiFactory.sol - an instance of LibertiVault factory.
- LibertiFactoryBase.sol - a factory that uses a minimal proxy pattern for deploying new instances of LibertiVault.
- LibertiPriceFeed.sol - interacts with Chainlink Aggregator V3 interface to fetch data about the current token price.
- LibertiVault.sol - a core contract of the system. ERC4626-like vault.

Privileged roles

- Owner:
  - LibertiPriceFeed.sol - can add price feed.
  - LibertiVault.sol - can rebalance the vault, set minimal and maximal deposit values, entry and exit fee.

Risks

- Off-chain trading logic is not verified in the scope of the audit.

Checked Items

We have audited the Customers' smart contracts for commonly known and specific vulnerabilities. Here are some items considered:

Item	Type	Description	Status
Default Visibility	SWC-100 SWC-108	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	Passed
Integer Overflow and Underflow	SWC-101	If unchecked math is used, all math operations should be safe from overflows and underflows.	Not Relevant
Outdated Compiler Version	SWC-102	It is recommended to use a recent version of the Solidity compiler.	Passed
Floating Pragma	SWC-103	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	Passed
Unchecked Call Return Value	SWC-104	The return value of a message call should be checked.	Passed
Access Control & Authorization	CWE-284	Ownership takeover should not be possible. All crucial functions should be protected. Users could not affect data that belongs to other users.	Passed
SELFDESTRUCT Instruction	SWC-106	The contract should not be self-destructible while it has funds belonging to users.	Not Relevant
Check-Effect-Interaction	SWC-107	Check-Effect-Interaction pattern should be followed if the code performs ANY external call.	Passed
Assert Violation	SWC-110	Properly functioning code should never reach a failing assert statement.	Passed
Deprecated Solidity Functions	SWC-111	Deprecated built-in functions should never be used.	Passed
Delegatecall to Untrusted Callee	SWC-112	Delegatecalls should only be allowed to trusted addresses.	Not Relevant
DoS (Denial of Service)	SWC-113 SWC-128	Execution of the code should never be blocked by a specific contract state unless required.	Passed

Race Conditions	SWC-114	Race Conditions and Transactions Order Dependency should not be possible.	Passed
Authorization through tx.origin	SWC-115	tx.origin should not be used for authorization.	Not Relevant
Block values as a proxy for time	SWC-116	Block numbers should not be used for time calculations.	Not Relevant
Signature Unique Id	SWC-117 SWC-121 SWC-122 EIP-155 EIP-712	Signed messages should always have a unique id. A transaction hash should not be used as a unique id. Chain identifiers should always be used. All parameters from the signature should be used in signer recovery. EIP-712 should be followed during a signer verification.	Not Relevant
Shadowing State Variable	SWC-119	State variables should not be shadowed.	Passed
Weak Sources of Randomness	SWC-120	Random values should never be generated from Chain Attributes or be predictable.	Not Relevant
Incorrect Inheritance Order	SWC-125	When inheriting multiple contracts, especially if they have identical functions, a developer should carefully specify inheritance in the correct order.	Passed
Calls Only to Trusted Addresses	EEA-Level-2 SWC-126	All external calls should be performed only to trusted addresses.	Passed
Presence of Unused Variables	SWC-131	The code should not contain unused variables if this is not justified by design.	Passed
EIP Standards Violation	EIP	EIP standards should not be violated.	Passed
Assets Integrity	**Custom**	Funds are protected and cannot be withdrawn without proper permissions or be locked on the contract.	Passed
User Balances Manipulation	**Custom**	Contract owners or any other third party should not be able to access funds belonging to users.	Passed
Data Consistency	**Custom**	Smart contract data should be consistent all over the data flow.	Passed

Flashloan Attack	**Custom**	When working with exchange rates, they should be received from a trusted source and not be vulnerable to short-term rate changes that can be achieved by using flash loans. Oracles should be used.	Passed
Token Supply Manipulation	**Custom**	Tokens can be minted only according to rules specified in a whitepaper or any other documentation provided by the Customer.	Passed
Gas Limit and Loops	**Custom**	Transaction execution costs should not depend dramatically on the amount of data stored on the contract. There should not be any cases when execution fails due to the block Gas limit.	Passed
Style Guide Violation	**Custom**	Style guides and best practices should be followed.	Passed
Requirements Compliance	**Custom**	The code should be compliant with the requirements provided by the Customer.	Passed
Environment Consistency	**Custom**	The project should contain a configured development environment with a comprehensive description of how to compile, build and deploy the code.	Passed
Secure Oracles Usage	**Custom**	The code should have the ability to pause specific data feeds that it relies on. This should be done to protect a contract from compromised oracles.	Passed
Tests Coverage	**Custom**	The code should be covered with unit tests. Test coverage should be sufficient, with both negative and positive cases covered. Usage of contracts by multiple users should be tested.	Passed
Stable Imports	**Custom**	The code should not reference draft contracts, which may be changed in the future.	Passed

Findings

■■■■ Critical

C01. Data Consistency

It is recommended to perform a validity check on the *SwapDescription.srcToken* and *SwapDescription.dstToken* values within both the *_deposit* and *_redeem* functions. This precautionary measure is necessary because the swap data may be subject to manipulation, which could result in the depletion of all funds from the contract.

For instance, in the WBTC/USDT vault, an attacker could potentially exploit the redemption process by reversing the order of the swap for 100 USDT to drain 100 WBTC through the swap operation. Therefore, it is crucial to ensure that the swap path is properly validated to prevent any potential security breaches.

Path:
./contracts/LibertiVault.sol : _deposit(), _redeem(), rebalance()

Recommendation: Add validations that *srcToken* and *destToken* are expected tokens in all functions that use the swap() function. For example, in *_redeem()* function check if *srcToken == other* and *destTokne == asset*.

Found in: 03740ef3953595e268cb55b34166c58398f14a15

Status: Fixed (Revised commit: 2ec8b10)

■■■ High

H01. Funds Lock

Native coins and tokens should have mechanisms of their withdrawal from the contract if they are accepted by the contract.

Path:
./contracts/LibertiVault.sol : receive()

Recommendation: Add mechanism of native tokens withdrawal available for owner.

Found in: 03740ef3953595e268cb55b34166c58398f14a15

Status: Fixed (Revised commit: 2ec8b10)

H02. Non-Finalized Code: FIXME Comments

The code should not contain FIXME comments. Otherwise, it means that the code is not finalized and additional changes will be introduced in the future.

Path:
./contracts/LibertiVault.sol : rebalance(), deposit(), depositETH()

Recommendation: Finalize the code and remove the comments.

Found in: 03740ef3953595e268cb55b34166c58398f14a15

Status: Fixed (Revised commit: 2ec8b10)

H03. Invalid Calculations

sharesToToken1 function uses *shares = MathUpgradeable.max(shares, supply);* logic to handle cases when *shares* parameter is bigger than *supply*. This logic is incorrect, as it always uses bigger value, so for any input parameter less than *totalSupply* result would be the same.

Path:
./contracts/LibertiVault.sol : sharesToToken1()

Recommendation: Replace *MathUpgradeable.max* with *MathUpgradeable.min*.

Found in: 2ec8b102e4b34e719c21a5c42358bcfcdc24f3b4

Status: Fixed (Revised commit: 595bf17)

Medium

M01. Requirements Violation

According to existing comments in the code - it is impossible to send tokens from sanctioned addresses, but LibertiVault tokens could be sent from sanctioned addresses to not sanctioned addresses.
The sanctioned address is able to *redeem* their funds from the vault.

During the second review, the issue is still reproducible in the scenario: A user deposited his funds into the vault, and after his address is added to sanctions list, but user can still send LP tokens to any other address and withdraw them.

Path:
./contracts/LibertiVault.sol : _beforeTokenTransfer()

Recommendation: Add *from* address sanctions check in _beforeTokenTransfer() or update the documentation with proper information.

Found in: 03740ef3953595e268cb55b34166c58398f14a15

Status: Mitigated (Documentation is updated: sanctioned address allowed to redeem LP tokens, but not allowed to deposit.)

M02. Best Practice Violation: Failing Assert Statement

The code uses *assert(false)* in a case when the chain is different from ETH, Polygon or BNB. Unlike *revert* approach, this way of throwing errors uses more Gas and provides no readable error description.

Path:
./contracts/LibertiAggregationRouterV4.sol : swap()

Recommendation: Replace to *revert Error()*.

Found in: 03740ef3953595e268cb55b34166c58398f14a15

Status: Fixed (Revised commit: 2ec8b10)

M03. Inefficient Gas Model: Redundant Interactions

Function swap checks equality of *swapAmount* and *desc.amount*. When the function is called from the *rebalance()* method, *swapAmount* value is decoded from struct and equals the *swapAmount* itself.

Path:
./contracts/LibertiVault.sol : rebalance()

Recommendation: In *rebalance()* method, remove the decode of the desc struct and pass *swapAmount* as a function argument.

Found in: 03740ef3953595e268cb55b34166c58398f14a15

Status: Fixed (Revised commit: 2ec8b10)

M04. Missing Event For Critical Value Updation

Critical state changes should emit events for tracking things off-chain.

Paths:
./contracts/LibertiPriceFeed.sol : addPriceFeed()
./contracts/LibertiVault.sol : setMinDeposit(), setMaxDeposit(), setEntryFee(), setExitFee()

Recommendation: Emit events on critical state changes.

Found in: 03740ef3953595e268cb55b34166c58398f14a15

Status: Fixed (Revised commit: 2ec8b10)

M05. Name Contradiction

The state variable *invariant* contradicts its name.

The logic of contract indicates that invariant should be a constant for the Basic Point numerator.

There should be a state variable called "proportion/ratio" used as a way to track the allocation coefficient.

Path:
./contracts/LibertiVault.sol : invariant

Recommendation: There should be no magic number in code 10_000.

Found in: 03740ef3953595e268cb55b34166c58398f14a15

Status: Fixed (Revised commit: 2ec8b10)

M06. Missing Validation

There is no *entryFee* and *exitFee* validation in the initialize function, so any value could be set during the initialization.

Path:
./contracts/LibertiVault.sol : initialize()

Recommendation: Validate fees during contract initialization.

Found in: 03740ef3953595e268cb55b34166c58398f14a15

Status: Fixed (Revised commit: 2ec8b10)

M07. Missing Validation

sharesToToken1 function should validate if value from the input parameter *shares* is not bigger than *totalSupply()*.

During the second review, H04 issue was added related to this problem.

Path:
./contracts/LibertiVault.sol : sharesToToken1()

Recommendation: Add proper validation.

Found in: 03740ef3953595e268cb55b34166c58398f14a15

Status: Fixed (Revised commit: 595bf17)

M08. Documentation Mismatch

Contract contains functions that are not covered by the documentation: *depositEth()* and *redeemEth()* are only valid and designed for native token vaults. Moreover, it will always revert in vaults like WBTC/USDT.

Path:
./contracts/LibertiVault.sol : depositEth(), redeemEth()

Recommendation: Describe this functionality in documentation or remove it from the contract.

Found in: 03740ef3953595e268cb55b34166c58398f14a15

Status: Fixed (Revised commit: 2ec8b10)

Low

L01. Floating Pragma

Locking the pragma helps ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.

Paths:
./contracts/interfaces/ILibertiPriceFeed.sol
./contracts/interfaces/ILibertiVault.sol
./contracts/interfaces/ISanctionsList.sol
./contracts/interfaces/IWeth9.sol
./contracts/LibertiAggregationRouterV4.sol
./contracts/LibertiFactory.sol
./contracts/LibertiFactoryBase.sol
./contracts/LibertiPriceFeed.sol
./contracts/LibertiVault.sol

Recommendation: Consider locking the pragma version whenever possible and avoid using a floating pragma in the final deployment.

Found in: 03740ef3953595e268cb55b34166c58398f14a15

Status: Fixed (Revised commit: 2ec8b10)

L02. Missing Zero Address Validation

`_asset` and `_other` addresses should be checked if they are not zero to prevent contract initialization with zero addresses.

Path: ./contracts/LibertiVault.sol : initialize();

Recommendation: Check if token addresses are not zero.

Found in: 03740ef3953595e268cb55b34166c58398f14a15

Status: Fixed (Revised commit: 2ec8b10)

L03. Functions That Can Be Declared External

The “public” functions that are never called by the contract should be declared “external” to save Gas.

Path:
./contracts/LibertiVault.sol: convertToAssets(), convertToShares(), initialize()

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

Found in: 03740ef3953595e268cb55b34166c58398f14a15

Status: Fixed (Revised commit: 2ec8b10)

L04. Style Guide Violation

The project should follow the official code style guidelines. Inside each contract, library, or interface, use the following order:

- Type declarations
- State variables
- Events
- Modifiers
- Functions

Functions should be grouped according to their visibility and ordered:

- constructor
- receive function (if exists)
- fallback function (if exists)
- external
- public
- internal
- private

Within a grouping, place the view and pure functions at the end.

Path:
./contracts/LibertiVault.sol

Recommendation: The official Solidity style guidelines should be followed.

Found in: 03740ef3953595e268cb55b34166c58398f14a15

Status: Fixed (Revised commit: 2ec8b10)

L05. Missing/Inconsistent NatSpec

NatSpec are inconsistent in *LibertiVault.sol* and missing in the other contracts.

Paths:
./contracts/LibertiAggregationRouterV4.sol
./contracts/LibertiFactory.sol
./contracts/LibertiFactoryBase.sol
./contracts/LibertiPriceFeed.sol
./contracts/LibertiVault.sol

Recommendation: Correct NatSpecs.

Found in: 03740ef3953595e268cb55b34166c58398f14a15

Status: Reported

L06. Typo in Comments

Some comments have typos that should be corrected.

“inclusing” -> including
“thei” -> their
“transfering” -> transferring

Path:
./contracts/LibertiVault.sol

Recommendation: Correct typos in comments.

Found in: 03740ef3953595e268cb55b34166c58398f14a15

Status: Fixed (Revised commit: 2ec8b10)

L07. Redundant Virtual Modifier

The virtual keyword in top-level functions is redundant and can be removed.

Path:
./contracts/LibertiVault.sol : _beforeTokenTransfer()

Recommendation: Remove virtual from function.

Found in: 03740ef3953595e268cb55b34166c58398f14a15

Status: Fixed (Revised commit: 2ec8b10)

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed based on best industry practices at the time of the writing 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 report contains no statements or warranties on the identification of all vulnerabilities and security of the code. The report covers the code submitted and reviewed, so it may not be relevant after any modifications. Do not consider this report as a final and 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.

English is the original language of the report. The Consultant is not responsible for the correctness of the translated versions.

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