DeHacker

Code Security Assessment

CoinFLEX

Mar 8th, 2024

Contents

Summary

DeHacker's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.
Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow/underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service/logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Issue Categories

Every issue in this report was assigned a severity level from the following:

Critical severity issues

A vulnerability that can disrupt the contract functioning in a number of scenarios or creates a risk that the contract may be broken.

Major severity issues

A vulnerability that affects the desired outcome when using a contract or provides the opportunity to use a contract in an unintended way.

Medium severity issues

A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.

Minor severity issues

A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.

Informational

A vulnerability that has informational character but is not affecting any of the code.

Overview

Project Summary

Project Name	CoinFLEX
Platform	Ethereum
Website	https://CoinFLEX.com/
Type	Dex
Language	Solidity

Vulnerability Summary

Vulnerability Level	**Total**	**Pending**	**Declined**	**Acknowledged**	**Partially Resolved**	**Resolved**
Critical	0	0	0	0	0	0
Major	1	0	0	1	0	0
Medium	1	0	0	1	0	0
Minor	1	0	0	1	0	0
Informational	1	0	0	1	0	0
Discussion	0	0	0	0	0	0

Audit scope

ID	File	SHA256 Checksum
BBS	CoinFLEX TOKEN.sol	bbfd85001c20a493630a181edff28d07dd6bd9dac1bc95b98384 b8e74aa6f604
BSC	strategies/BaseStrategy.s ol	762e19905e2777a91aca11df94731c71c10638d2e8df1b3fb334f7 ba930d0a99
FLE	strategies/FLEXStakingStrategy. sol	7c1bd0a584075586a513ff645a9baa6cc1966e1986732d5216ebf 9e031240ba6
DPC	Controller.sol	2344be4d1ef56a570588844ae0fba23bb8637cc302640a7991000 5ab396fe98a
QPC	QuarterlyPayout.sol	6aba1d7bd822780a844f296c0f14dcf2d37aca3dca110c9 45891585d18e30214
TCK	Timelock.sol	ce9042971ff1a5e04088993dac17a724f9165c7276069988a647ec 3cc9aa0a27

Findings

ID	Category	Severity	Status
GLOBAL-01	Centralization Risk	Major	Acknowledged
BSC-01	Potential Privilege Leakage	Minor	Acknowledged
CCK-01	Converter Not Initialized	Medium	Acknowledged
CCK-02	Unused Function Execute	Informational	Acknowledged

MAJOR

GLOBAL-01 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	Major	Global	Acknowledged

Description

In the contract Controller , the role governance has the authority over the following functions andupdates the significant states of the contract:setTreasury() : update the treasury address to any arbitrary address.setGovernance() : update the governance address to any arbitrary address.setVault() : update the vault contract address to any arbitrary address.revokeStrategy() : revoke the candidate strategy.setStrategy() : appoint the candidate strategy.withdrawAll() : withdraw all the funds within the strategy contracts.inCaseTokensGetStuck() : withdraw all the tokens within the contract.inCaseStrategyTokenGetStuck() : withdraw funds from strategy contracts

Recommendation

It is advised to carefully manage the governance , controller , vaults , and owner accounts' private keysto avoid any potential risks of being hacked. In addition, the timelock address should be correctlyinitialized in the constructor.In general, we strongly recommend centralized privileges or roles in the protocol to be improved via adecentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g.,Multisignature wallets

MINOR

BSC-01 | Control Flow

Category	Severity	Location	Status
Control Flow	Minor	strategies/BaseStrategy.sol: 75	Acknowledged

Description

According to the implementation of whitelistHarvesters() function, the harvester role can update theharvesters array and designate arbitrary address(es) as the harvester role. If the harvester is a maliciousaddress, it can designate numerous malicious addresses as the harvester. This might violate the principleof least privilege (PoLP) in which a user should be given the minimum levels of access/permissions neededto perform his/her job functions.In addition, the harvesters and the governance address can add harvesters, yet only governance canrevoke harvesters. This could be risky if governance mistakenly adds a malicious harvester, the malicioush arvester can add more malicious harvesters and trigger harvest() at any time. In the meantime, if thegovernance address does not take immediate action on this situation, it might cause unexpected loss

Recommendation

It is recommended to only allow the governance address to add harvesters.

Medium

CCK-01 | Converter Not Initialized

Category	Severity	Location	Status
Logical Issue	Medium	Controller.sol:122	Acknowledged

Description

The coverter of the contract is supposed to convert tokens to the wanted token.122122 addressaddress converterconverter == convertersconverters[[_token_token]][[_want_want]];;123123 IERC20IERC20((_token_token))..safeTransfersafeTransfer((converterconverter,, _amount_amount));;124124 _amount_amount == IConverterIConverter((converterconverter))..convertconvert((_strategy_strategy));;However, the array converters is not initialized within the contract, in this case, the converter will be theaddress(0) by default.

Recommendation

It is recommended to revisit the logic of convertor and add neccessary logic to update the convertersarray.

INFORMATIONAL

CCK-02 | Unused Function

Category	Severity	Location	Status
Gas Optimization	Informational	Contraller.sol:129-130	Acknowledged

Description

The internal facing function _execute in L129 is designed to execute low level calls. However, it is notused within the contract. If the function is not intended to be used anywhere, it can be safely omitted.

Recommendation

Remove the unused function from the contract.

Disclaimer

This report is based on the scope of materials and documentation provided for a limited review at the time provided. Results may not be complete nor inclusive of all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Blockchain technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. A report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate.

FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Coding Style

Coding Style findings usually do not affect the generated bytecode but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block. timestamp works.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux “sha256sum" command against the target file.

About

DeHacker is a team of auditors and white hat hackers who perform security audits and assessments. With decades of experience in security and distributed systems, our experts focus on the ins and outs of system security. Our services follow clear and prudent industry standards. Whether it's reviewing the smallest modifications or a new platform, we'll provide an in-depth security survey at every stage of your company's project. We provide comprehensive vulnerability reports and identify structural inefficiencies in smart contract code, combining high-end security research with a real-world attacker mindset to reduce risk and harden code.

BLOCKCHAIINS

Ethereum	Cosmos
Eos	Substrate

TECH STACK

Python	Solidity
Rust	c++

CONTACTS

https://dehacker.io

https://twitter.com/dehackerio

https://github.com/dehacker/audits_public

https://t.me/dehackerio

https://blog.dehacker.io/

DeHacker

Mar 2024