DeHacker

Code Security Assessment

Singularity Dao Yield Farming

March10th, 2023

Contents

CONTENTS 1
SUMMARY 2
ISSUE CATEGORIES 3
OVERVIEW 4
PROJECT SUMMARY 4
VULNERABILITY SUMMARY 4
AUDIT SCOPE 5
FINDINGS 6
INFORMATIONAL 7
SDA-01 : Redundant Variable Initialization 7
DESCRIPTION 7
RECOMMENDATION 7
INFORMATIONAL 8
SDA-02 : Inexistent Input Sanitization 8
DESCRIPTION 8
RECOMMENDATION 8
MINOR 9
SDA-03 : Ambiguous Data Type 9
DESCRIPTION 9
RECOMMENDATION 9
MEDIUM 10
SDA-04 : Potential Re-Entrancy 10
DESCRIPTION 10
RECOMMENDATION 10
INFORMATIONAL 11
SDA-05 : Partial NatSpec Comments 11
DESCRIPTION 11
RECOMMENDATION 11
MEDIUM 12
SDA-06 : Potential Over-centralization of Functionality 12
DESCRIPTION 12
RECOMMENDATION 12
INFORMATIONAL 13
SDA-07 : Redundant Type Cast 13
DESCRIPTION 13
RECOMMENDATION 13
DISCLAIMER 14
APPENDIX 15
ABOUT 16

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	Singularity Dao Yield Farming
Platform	Ethereum
Website	https://www.singularitydao.ai/
Type	DAO
Language	Solidity

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
Major	0	0	0	0	0	0
Medium	2	0	0	1	0	1
Minor	1	0	0	1	0	0
Informational	4	0	0	0	0	4
Discussion	0	0	0	0	0	0

Audit scope

ID	File	SHA256 Checksum
SDA	SDAOTokenStaking.sol	b76c5df6aa2e5d22b1dfff66c8a63b5551b92cc87c9491cd1b6ab0eac7cf2d25

Findings

ID	Category	Severity	Status
SDA-01	Coding Style	Informational	Resolved
SDA-02	Logical Issue	Informational	Resolved
SDA-03	Volatile Code	Minor	Acknowledged
SDA-04	Volatile Code	Medium	Resolved
SDA-05	Coding Style, Inconsistency	Informational	Resolved
SDA-06	Centralization /Privilege	Medium	Acknowledged
SDA-07	Gas Optimization	Informational	Resolved

Informational

SDA-01 | Redundant Variable Initialization

Category	Severity	Location	Status
Coding Style	Informational	SDAOTokenStaking.sol (fade30b): 78	Resolved

Description

All variable types within Solidity are initialized to their default "empty" value, which is usually their zeroedout representation. Particularly:
uint / int : All uint and int variable types are initialized at 0

Recommendation

We advise that the linked initialization statements are removed from the codebase to increase legibility.

Informational

SDA-02 | Inexistent Input Sanitization

Category	Severity	Location	Status
Logical Issue	Informational	SDAOTokenStaking.sol (fade30b): 113~116	Resolved

Description

The linked function fails to check the values of the arguments, hence allowing for no pointsAllocator .

Recommendation

We advise to add a require statement, checking the input values against the zero address.

Minor

SDA-03 | Ambiguous Data Type

Category	Severity	Location	Status
Volatile Code	Minor	SDAOTokenStaking.sol (fade30b): 33	Acknowledged

Description

The rewardDebt member of the UserInfo struct is ambiguously declared as a int256 number.

Recommendation

We advise to either change the aforementioned data type, along with all the calculations it's involved in, touint256 , or add descriptive in-line documentation

Medium

SDA-04 | Potential Re-Entrancy

Category	Severity	Location	Status
Volatile Code	Medium	SDAOTokenStaking.sol (fade30b): 391, 366, 306, 278	Resolved

Description

The linked code segment updates the state of the contract after an external call.

Recommendation

We advise to execute the external call at the end of the function, hence following the Checks-Effects-Interactions pattern.

Informational

SDA-05 | Partial NatSpec Comments

Category	Severity	Location	Status
Coding Style, Inconsistency	Informational	SDAOTokenStaking.sol (fade30b): 136~139	Resolved

Description

The code segment does not contain NatSpec comments for all the parameters.

Recommendation

We advise to update the NatSpec comments of the linked code segment.

Medium

SDA-06 | Potential Over-centralization of Functionality

Category	Severity	Location	Status
Centralization / Privilege	Medium	SDAOTokenStaking.sol (fade30b): 396~401	Acknowledged

Description

The linked function is meant to be used in an edge-case situation whereby the owner can withdraw thecontracts native tokens and Ether.

Recommendation

We advise this functionality to be guarded by either a time delay to ensure that the normal course ofoperation of the contract has progressed.

Informational

SDA-07 | Redundant Type Cast

Category	Severity	Location	Status
Gas Optimization	Informational	SDAOTokenStaking.sol (fade30b): 165~168	Resolved

Description

The linked type cast to uint256 is redundant, as the data type of pool.tokenPerBlock is already of thesame type.

Recommendation

We advise to remove redundant code.

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

March 2023