DeHacker

Code Security Assessment

DAO Maker - Vesting

Aug 1st, 2023

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	DAO Maker - Vesting
Platform	Ethereum
Website	https://daomaker.com/
Type	Defi
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	0	0	0	0	0	0
Minor	1	0	0	0	0	1
Informational	2	0	0	2	0	0
Discussion	0	0	0	0	0	0

Audit scope

ID	File	SHA256 Checksum
BVB	contracts/helpers/BurnValley.sol	fc5d630539f7cfd1a279a7bbd35ccf48035492d5cca4d9eac35ca2959f26c74b
SHO	contracts/SHO.sol	6fabd49ab29dd9e5809b043dbc8b88dcdde3ec51d94bb8f3d92cc3fb85a76b8f

Findings

ID	Category	Severity	Status
SHO-01	Centralization /Privileg	Major	Acknowledged
SHO-02	Logical Issue	Minor	Resolved
SHO-03	Gas Optimization	Informational	Acknowledged
SHO-04	Volatile Code	Informational	Acknowledged

MAJOR

SHO-01 | Centralization Related RisksSK

Category	Severity	Location	Status
Centralization / Privilege	Major	contracts/SHO.sol: 168, 236	Acknowledged

Description

In the contract SHO , the role owner has authority over the following functions:

whitelistUsers() : allows the owner to white list users;
eliminateUsers1() : allows the owner to removes all the future allocation of passed user type 1addresses.

Any compromise to the owner account may allow a hacker to take advantage of these functions.

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the securityoperation and level of decentralization, which in most cases can't be resolved entirely at the present stage.We advise the client to carefully manage the privileged account's private key to avoid any potential risks ofbeing hacked. In general, we strongly recommend centralized privileges or roles in the protocol beimproved via a decentralized mechanism or smart-contract-based accounts with enhanced securitypractices, e.g., Multisignature wallets.

Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a differentlevel in terms of short-term, long-term and permanent:

Short Term:
Timelock and Multi sign (,) combination mitigate by delaying the sensitive operation and avoiding asingle point of key management failure.

Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
AND
Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to theprivate key compromised;
AND
A medium/blog link for sharing the timelock contract and multi-signers addresses information withthe public audience.

Long Term:
Timelock and DAO, the combination, were able to mitigate by applying decentralization and transparency.

Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
AND
Introduction of a DAO/governance/voting module to increase transparency and user involvement.
AND
A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO informationwith the public audience.

Permanent:
Renouncing the ownership or removing the function can be considered fully resolved.

Renounce the ownership and never claim back the privileged roles
OR
Remove the risky-functionalities

Minor

SHO-02 | whitelistUsers Could Overwrite Existing Total Allocation

Category	Severity	Location	Status
Logical Issue	Minor	contracts/SHO.sol : 197~198	Resolved

Description

According to the code implementation, whitelistUsers() adds users and their corresponding allocationsthat will be distributed in the later stage. Variables globalTotalAllocation1 / globalTotalAllocation2record the total allocation for "option 1"/"option 2", which are used in the fee calculation.

```
321  uint120 globalAllocation1 = _applyPercentage(globalTotalAllocation1,
unlockPercentages[currentUnlock] - lastUnlockPercentage);
322  uint120 globalAllocation2 = _applyPercentage(globalTotalAllocation2,
unlockPercentages[currentUnlock] - lastUnlockPercentage);
```

However, these two variables could be overwritten if whitelistUsers() is successfully invoked multipletimes. Therefore, it could lead to incorrect fee calculation and the project would suffer unexpected losses

Exploit Scenario:
The owner called whitelistUsers() to add users A, B and C with "option 1" (each with 1000allocation). Currently, the total allocation for "option 1" (globalTotalAllocation1) is 3000 and thetotal allocation for "option 2" (globalTotalAllocation2) is 0.
The owner called whitelistUsers() again to add user D with "option 2" (each with 1000 allocation).Due to overwriting, the total allocation for "option 1" (globalTotalAllocation1) is 0 and the totalallocation for "option 2" (globalTotalAllocation2) is 1000.
In this case, the project would lose the fee from "option 1".

Recommendation

In the short term, avoid calling this function twice. Add restrictions so that whitelistUsers() could onlybe invoked once

In the long term, redesign the whitelistUsers() logic. For example, increasing the value ofglobalTotalAllocation1 / globalTotalAllocation2 every time adding whitelisted users intead ofoverwriting.

```
1 globalTotalAllocation1 += _globalTotalAllocation1;
2 globalTotalAllocation2 += _globalTotalAllocation2;
```

INFORMATIONAL

SHO-03 | Improper Usage Of public And external Type

Category	Severity	Location	Status
Gas Optimization	Informational	contracts/SHO.sol: 374~376	Acknowledged

Description

In general, external functions are more efficient than public functions. The getTotalUnlocksCountfunction is never called by the contract and thus could be declared as external.

Recommendation

Consider using the external attribute for getTotalUnlocksCount function.

INFORMATIONAL

SHO-04 | Third Party Dependencies

Category	Severity	Location	Status
Volatile Code	Informational	contracts/SHO.sol: 117, 122, 124	Acknowledged

Description

The contract is serving as the underlying entity to interact with the following third-party protocols.
shoToken : tokens to be claimed by users,
feeCollector : address to receive fees,
burnValley : address to "burn" tokens.

The scope of the audit treats 3rd party entities as black boxes and assumes their functional correctness.However, in the real world, 3rd parties can be compromised and this may lead to lost or stolen assets. Inaddition, upgrades of 3rd parties can possibly create severe impacts.

For example, as per the project design, the burnValley should be a contract without any function.Transferring tokens to burnValley contract will forever lock the token. However, if burnValley is initializedwith an EOA address, it might cause unexpected loss.

Recommendation

As the auditors do not know if the project will be deployed and initialized as planned, the status of thisissue will be updated after contract deployment upon request.

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

July 2023