DcentraLab Diligence

dcentralab.com/diligence

Poolz

Vault Audit Report

Poolz Finance

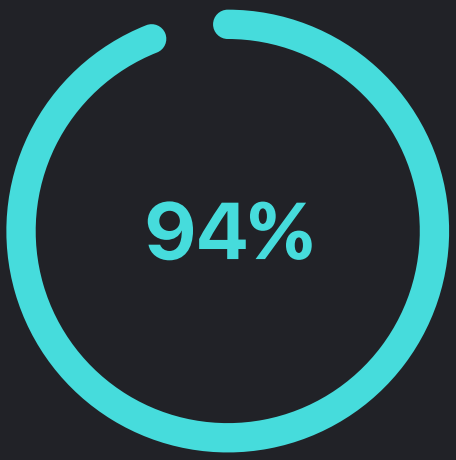
https://www.poolz.finance

Provided By DcentraLab Diligence on March 27, 2023

Security Audit Score

Pass

DcentraLab Diligence team has conducted an extensive audit on the Poolz Delay Vault Contracts and has found the code to be in high quality and low risk level given proper deployment and multi-sig permissioning



- Low Risk
- Small Risk
- Medium Risk
- High Risk

Scope

Audited Repository:

github.com/The-Poolz/DelayVault

Audited Branch:

master

Audited Commit Hash:

03490e948a05f6556d0571e8781547ec42387ed1

Audited Sub-Repository Of Imported Contracts:

Imported Contracts Repository:

github.com/The-Poolz/Poolz-Helper

Imported Contracts Branch:

master

Imported Contracts Audited Commit Hash:

cce239526836fb63928e7d5528a1187b63d39097

Fix Commit Hash:

github.com/The-Poolz/DelayVault/commit/8a75232861aacc3862e4718e959792308511766a

Contracts Audited:

DelayData.sol
DelayEvents.sol
DelayManageable.sol
DelayModifiers.sol
DelayVault.sol
DelayView.sol

Audited Imported Contracts:

GovManager.sol
ERC20Helper.sol
Array.sol
ILockedDealV2.sol

Scope

Final Fixes Commit Hash:

8a75232861aacc3862e4718e959792308511766a

Final Fixes Diff View:

https://github.com/The-Poolz/DelayVault/compare/V1.1.2...V1.2.1

Risks:

DcentraLab Diligence (DD) has performed all checks and verifications in its capacity to ascertain the safety of the code. However, it should be noted that misuse of the code, bad deployment practices, bad key management, exposing of private keys of the deployer and/or owner address and/or multi-sig signer addresses and/or fee collector address and/or any exposition of the code to malicious actors may result in an exploit of the code and loss of state and/or funds. Furthermore, there is always a chance that other Smart Contracts code could be written and deployed to cause the provided code by DD to act outside the intended scope by the client, to the point of causing state corruption or loss of funds to the client of the users of the code.

Intro/Description:

Delay Vault is a main contract of POOLZ Vault architecture, consisting of DelayData, DelayView, DelayModifiers, DelayManageable and DelayEvents.
It serves for POOLZ users to deposit funds and set cooldown/lock-time settings, which are applied on withdrawal. Staking tokens inside a vault enables users to earn tickets for the IDOs.

Token flow:
EOA -> DelayVault -> LockedDealV2 -> EOA

User Callable Functions:

- CreateVault -> Lets users create a vault for his tokens, or manage an already existing one.
- Withdraw -> Lets users enter a withdrawal period, which means their tokens will be sent to the LockedDealV2 Contract where they will stay for the cooldown period. If LockedDealV2 Contract is not set, tokens are being sent back to users directly, without applying the cooldown period flow.
- SwapBuyBackStatus -> Lets users approve the redemption of their tokens by the admin/ contract owner/governor contract.

Owner Or Governor Callable Functions:

Owner And Governor Are Playing The Same Role On These Functions

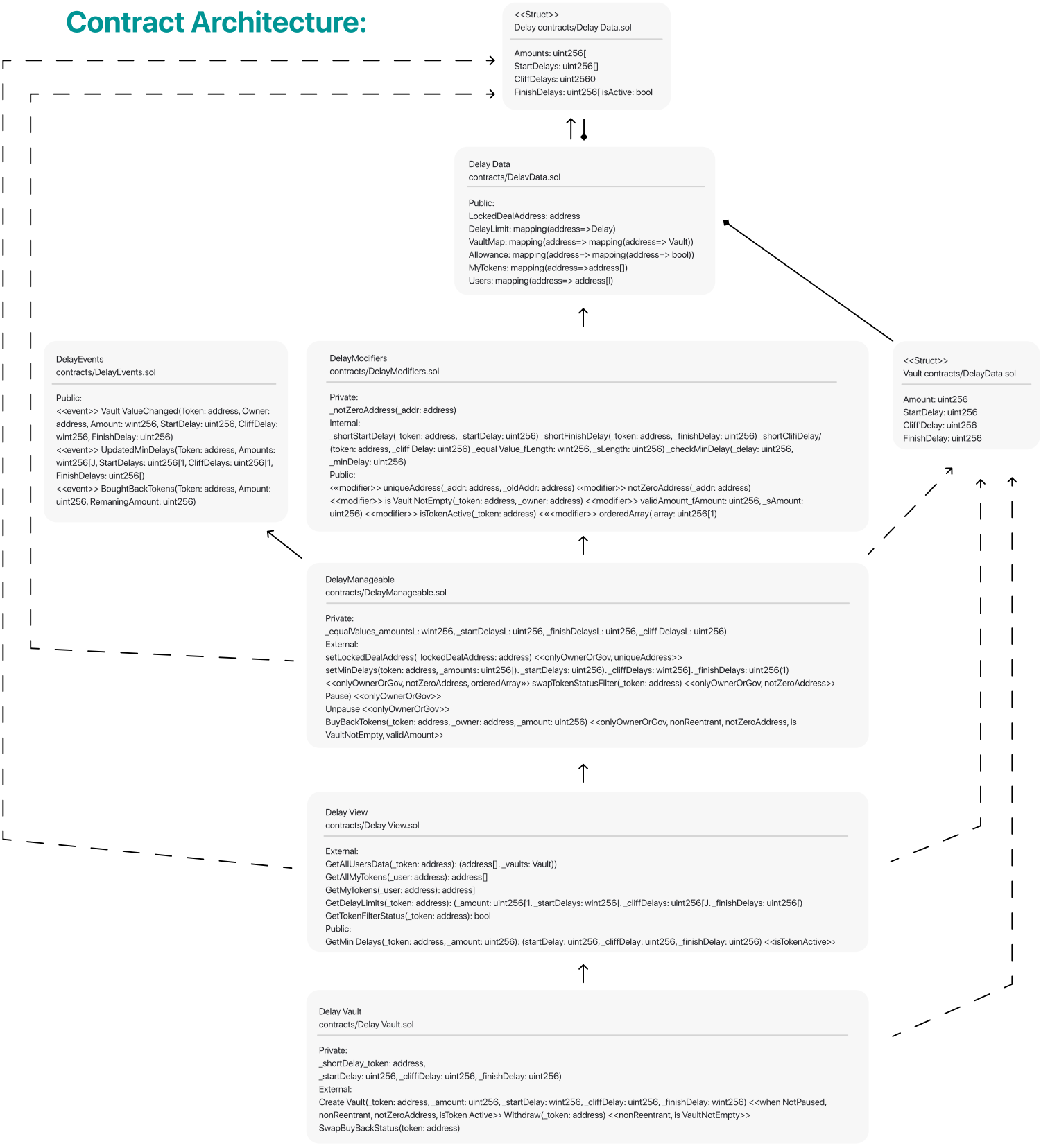
- BuyBackTokens -> Lets the governor to withdraw a user's tokens to the owner wallet. Can be used only if there is Allowance for the owner to do so.
- SetMinDelays -> Lets the governor the ability to set minimum delay to a specific token. A user needs to fit into the min delay requirements based on the amount he deposit and delay that was set by the governor for that amount range.SwapTokenStatusFilter -> Lets the governor change the isActive flag on the DelayLimit mapping. If the token is not active there is not possibility to create vaults for this token.
- Pause -> Lets the governor to pause the DelayVault contract.
- Unpause -> Lets the governor to unpause the DelayVault contract.

Owner Callable Functions:

Part Of OpenZeppelin Ownable Contract

- renounceOwnership -> Lets the owner the ability to remove himself from ownership.
- transferOwnership -> Lets the owner to transfer the ownership to a different wallet address.

Contract Architecture:



Issues Severity Reference Table

Type

Informational

This issue is not critical and does not pose an immediate threat to the functionality or security of the smart contract. It is simply an informational item that the auditors have identified and recommends addressing for best practices or to improve the overall performance of the contract.

Low

This issue is relatively minor and does not pose a significant risk to the functionality or security of the smart contract. While it is recommended to address these issues to ensure the highest level of quality and security, they are not likely to cause significant problems if left unaddressed.

Medium

This issue poses a moderate risk to the functionality or security of the smart contract. While it may not be immediately exploitable, it has the potential to cause problems in the future if left unaddressed. It is recommended to address these issues as soon as possible to prevent any potential negative impact on the contract.

High

This issue poses a significant risk to the functionality or security of the smart contract. Addressing these issues as soon as possible is recommended to prevent any potential negative impact on the contract. Failure to address these issues could result in significant problems and potential loss of funds or other assets.

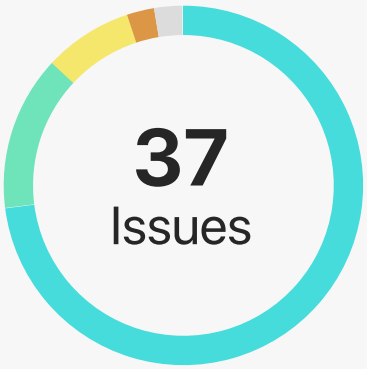
Critical

This issue poses an immediate and severe risk to the functionality or security of the smart contract. It is recommended to address these issues immediately to prevent any potential negative impact on the contract. Failure to address these issues could result in catastrophic problems and significant loss of funds or other assets.

Discussion

The issue severity is dependent on design, centralization, and product specifications of the project.

Findings Summary



- Informational
- High Risk
- Low Risk
- Discussion
- Medium Risk

ID	Title	Severity	Status
1	Unconventional naming	Informational	Acknowledged
2	Floating pragma version	Informational	Resolved
3	Invalid stack too deep error prevention measure	Informational	Resolved
4	DelayEvents is a contract	Informational	Resolved
5	Overcomplicated GetDelayLimits flow	Informational	Resolved
6	Invalid usage of storage keyword	Informational	Resolved
7	Unoptimized GetAllUsersData function	Informational	Resolved
8	Unconventional inheritance order	Informational	Acknowledged
9	Unnecessarily elaborate check	Informational	Resolved
10	Breakable getters	Medium	Resolved

Findings Summary

ID	Title	Severity	Status
11	Funds can be locked forever	High	Resolved
12	CEI Not Applied	Low	Resolved
13	The ability to increase only the time parameters will break	Medium	Resolved
14	Wrong error messenger	Informational	Resolved
15	Withdraw func - Hardcoded value when emitting an event	Informational	Resolved
16	Naming convention	Informational	Resolved
17	createVault - bad error message description	Informational	Resolved
18	createVault - double checking of isTokenActive modifier when calling GetMinDelays	Informational	Resolved
19	_shortDelay - function belongs to DelayModifiers contract and bad naming	Informational	Resolved
20	GetMinDelays - non optimal gas usage on search	Informational	Acknowledged
21	GetAllUsersData - return statement naming convention	Informational	Resolved
22	swapTokenStatusFilter - state change but no event emitted.	Informational	Resolved

Findings Summary

ID	Title	Severity	Status
23	swapTokenStatusFilter - state change but no event emitted.	Informational	Resolved
24	GovernerContract param typo	Informational	Resolved
25	Users mapping naming	Informational	Resolved
26	Improve performance by using binary search	Informational	Acknowledged
27	Unnecessary function	Informational	Resolved
28	setGovernerContract should emit an event	Informational	Resolved
29	setGovernerContract lack of zero address check	Informational	Acknowledged
30	Missing deployment script	Informational	Resolved
31	Lack of structure checksum scripts	Medium	Acknowledged
32	Upgradeability scheme	Discussion	Acknowledged
33	possible unintended state manipulation on swapBuyBackStatus	Low	Resolved
34	possible unintended state manipulation on swapTokenStatusFilter	Low	Resolved
35	function BuyBackTokens doesn't buy back tokens	Informational	Resolved

Findings Summary

ID	Title	Severity	Status
36	Use of require statements in ^0.8.4	Low	Acknowledged
37	In SetMinDelays function there is no MaxDelay validation	Low	Resolved

Complete Analysis

General Recommendations:

Add comments to functions (also @param and @return where needed) to improve readability, enhancing code documentation and ensuring code quality. Adding comments to functions in Solidity is the best practice for writing high-quality code.

ID 1: **Status: Acknowledged**

Informational | Unconventional naming

Present at: Across the whole architecture

Description: Naming / casing across the contract varies from one another. That makes syntax unusual / unconventional.

Recommendation: We recommend applying a conventional set of rules on your syntax casing, such as camel case which is one of the most popular standards in solidity.

Fix feedback: Project decided to fix this issue in some of their further iterations.

ID 2: **Status: Resolved**

Informational | Floating pragma version

Present at: Across the whole architecture

Description: Contracts can be compiled using multiple compiler versions which could lead to unexpected issues.

Recommendation: Set fixed pragma version.

Complete Analysis

ID 3: **Status: Resolved**

Informational | Informational - Invalid stack too deep error prevention measure

Present at: L35-43 @ DelayManageable Contract & L38-43 @ DelayVault Contract

Description: There's an unnecessary bracket addition to the function creating a sub block. In the current case there is no need for a sub block as _equalValues() is a pure function which does not return any value to be kept on stack. This approach is only useful when there is a variable which you use only in a specific place, so by using a sub block you can let the function continue with the rest of the flow without keeping that variable on stack.

Recommendation: Remove the unnecessary sub block brackets.

ID 4: **Status: Resolved**

Informational | DelayEvents is a contract

Present at: DelayEvents Contract

Description: DelayEvents is a contract containing events solely which means its entire functionality is not crossing possibilities of an interface.

Recommendation: Define DelayEvents as an interface instead of contract or add events into the DelayData contract.

ID 5: **Status: Resolved**

Informational | Overcomplicated GetDelayLimits flow

Present at: L44-60 @ DelayView Contract

Complete Analysis

Description: GetDelayLimits is an external getter function which includes unnecessarily big flow for returning variables. This does not represent an issue of importance, but the flow can be much more simplified and still provide the same experience.

Recommendation: Consider simplifying the flow by removing the variable definition and setting - instead just return variables you need directly from storage.

ID 6: **Status: Resolved**

Informational | Invalid usage of storage keyword

Present at: L33 @ DelayView Contract

Description: allTokens variable is defined with storage keyword, which is in this case completely unnecessary as the storage will not undergo any changes. Storage keyword usage is helpful when you want to edit complex types in multiple places, so you get the storage pointer of it using this keyword.

Recommendation: Use memory keyword instead to appropriate and optimize the function flow.

ID 7: **Status: Resolved**

Informational | Unoptimized GetAllUsersData function

Present at: L8-18 @ DelayView Contract

Description: Function contains multiple unoptimized storage readings.

Recommendation: Instantiate local variables for arrays you are reading from, so that you don't need to read from storage using multiple pointers each time.

Complete Analysis

ID 8: **Status: Acknowledged**

Informational | Unconventional inheritance order

Present at:DelayManageable Contract

Description: Contract inheritance does not have a particular order or standard in which they're sorted.

Recommendation: Apply a standard to order of inheritances, by contract type or creator. (This would be far more important in case of upgradeability schemes.)

Fix feedback: Project decided to fix this issue in some of their further iterations.

ID 9: **Status: Resolved**

Informational | Unnecessarily elaborate check

Present at: L102 @ DelayManageable Contract

Description: Check being performed at line 102 is contained in multiple operations and can be optimized.

Recommendation: Check if vault.Amount is equal to _amount instead of performing a multi operation check.

ID 10: **Status: Resolved**

Medium | Breakable getters

Present at: Delay View Contract

Complete Analysis

Description: Getters such as GetAllUsers data and GetAllMyTokens are returning full dynamic arrays. Since there's a computational limit to the on-chain EVM functions, such functions can become unusable once arrays reach certain sizes.

Recommendation: Add arguments 'from' and 'to' to the functions, enabling selection of data sets to return (instead of returning whole array at all times, you will be able to select from which to which array index you want to return the data).

ID 11: **Status: Resolved**

High | Funds can be locked forever

Present at: CreateVault function @ Delay Vault Contract

Description: There is no proper sanitization to the lock times the user is setting. Therefore upon not cautious using or issue present on frontend, in the case of miscalculation of timestamps, the user can lock his tokens forever / or for an unwanted amount of time.

Recommendation: Introduce proper argument sanitization that will prevent described behavior.

ID 12: **Status: Resolved**

Low | CEI Not Applied

Present at: CreateVault function @ Delay Vault Contract

Description: Code is not ordered by the checks-effects-interactions standard which can lead to reentrancy. In the current case nonReentrant modifier prevents reentrancy, but current function flow is not a good practice.

Recommendation: Apply CEI standard to the CreateVault function flow.

Complete Analysis

ID 13: **Status: Resolved**

Medium | The ability to increase only the time parameters will break

Present at: CreateVault function @ DelayVault Contract

Description: line 44. In the require statement, it is possible to pass _amount = 0 to the function while ensuring that the time parameters are greater than 0. However, in line 44, there is a call to TransferInToken, which requires the _amount parameter to be greater than 0. As a result, if the desired behavior is to modify the time parameters only, this function will not work and will revert.

Recommendation: Use a different function to change time parameters or make sure you are not calling TransferInToken with a 0 amount.

ID 14: **Status: Resolved**

Informational | Wrong error messenger

Present at: TransferToken function @ ERC20Helper Contract

Description: line 32. Wrong error message on require statement.
"receive wrong amount of tokens"
This function is sending tokens and not receiving them.

Recommendation: change error message

ID 15: **Status: Resolved**

Informational | Withdraw func - Hardcoded value when emitting an event

Present at: line 88. VaultValueChanged event @ DelayVault Contract

Description: event is emitted with hardcoded values (zero's). For more readability and to avoid mistakes on future development you use read the actual values from the Vault struct.

Complete Analysis

Recommendation: use the Vault struct values instead of the hard coded 0.

ID 16: **Status: Resolved**

Informational | Naming convention

Present at: line 97. _shortDelay function @ DelayVault Contract

Recommendation: Consider adding "check" or "validate" to the function name to make it more readable, as this function is only used for validating parameters.

ID 17: **Status: Resolved**

Informational | createVault - bad error message description

Present at: line 30. createVault function @ DelayVault Contract

Description: error message is referring only to the amount parameters but there are another 3 parameters that might be relevant to this error message.

Recommendation: consider a more generic error message.

ID 18: **Status: Resolved**

Informational | createVault - double checking of isTokenActive modifier when calling GetMinDelays.

Present at: line 37. createVault function @ DelayVault Contract

Description: createVault function includes the isTokenActive modifier, and it also calls the GetMinDelays function, which also has this modifier.

Complete Analysis

Recommendation: Create an internal function called "_getMinDelays" that does not call the isTokenActive modifier. The external GetMinDelays function can then be a wrapper that uses the isTokenActive modifier.

ID 19: **Status: Resolved**

Informational | _shortDelay - function belongs to DelayModifiers contract and bad naming.

Present at: line 97. _shortDelay function @ DelayVault Contract

Description: _shortDelay function is only a wrapper of the 4 other _short functions on the DelayModifiers contract and for more readability it should be inside the DelayModifiers contract. Function name is not clear enough.

Recommendation: move function to DelayModifiers.sol and change all _short functions names to have “validate” or “check” in its name for example: “_validateDelay” or “_checkDelayMeetsMinRequierments”

ID 20: **Status: Acknowledged**

Informational | Informational - GetMinDelays - non optimal gas usage on search

Present at: line 62. _shortDelay function @ DelayView Contract

Description: Using iterative search cost O(n). can be reduced to O(log n)

Recommendation: consider user binary search instead of iterative loop to optimize gas usage

Fix feedback: There are too many changes required for a relatively low impact.

Complete Analysis

ID 21: **Status: Resolved**

Informational | GetAllUsersData - return statement naming convention

Present at: line 11. GetAllUsersData function @ DelayView Contract

Description: One of the return values has a name and the other doesn't.

Recommendation: consider one convention for returns statements for the entire contract

ID 22: **Status: Resolved**

Informational | swapTokenStatusFilter - state change but no event emitted.

Present at: line 11. swapTokenStatusFilter function @ DelayManageable Contract

Description: line 60. State is changed but no event has been emitted.

Recommendation: consider emitting an event.

ID 23: **Status: Resolved**

Informational | swapTokenStatusFilter - state change but no event emitted.

Present at: line 60. swapTokenStatusFilter function @ DelayManageable Contract

Description: State is changed but no event has been emitted.

Recommendation: consider emitting an event.

Complete Analysis

ID 24: **Status: Resolved**

Informational | GovernerContract param typo.

Recommendation: Governor instead of 'Governer'

ID 25: **Status: Resolved**

Informational | Users mapping naming.

Description: line 11. Naming does not describe that it is a mapping of users array for each token address.

Recommendation: change param name to tokenToUsers

ID 26: **Status: Acknowledged**

Informational | Improve performance by using binary search

Present at: line 67. isInArray function @ Array Contract

Description: isInArray function is using an interactive search. Consider using a binary search. Can be taken from Openzepplin.

Recommendation: Use OpenZeppelin array contract and specifically findUpperBound functionLink: https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/utils/Arrays.sol

Fix feedback: There are too many changes required for a relatively low impact.

Complete Analysis

ID 27: **Status: Resolved**

Informational | Unnecessary function

Present at: CheckBalance function @ ERC20Helper Contract

Description: line 36. CheckBalance is not necessary. You can use the ERC20 function balanceOf directly. Also "checkBalance" is not a good naming convention as it is not checking the balance it is only returning the balance.

Recommendation: remove function.

ID 28: **Status: Resolved**

Informational | setGovernerContract should emit an event.

Present at: setGovernerContract function @ GovManager Contract

Recommendation: Emit an event when setGovernerContract is called

ID 29: **Status: Acknowledged**

Informational | setGovernerContract lack of zero address check

Present at: setGovernerContract function @ GovManager Contract
Description: function is lack of the zero address check.

Recommendation: In case you would like to always have a valid governor, add a zero address check.

Fix feedback: Project want to be able to set Governor as the 0 address

Complete Analysis

ID 30: **Status: Resolved**

High | Missing deployment script

Description: There is no deployment script to verify proper deployment flow and transfer ownership over vault to a cold storage multisig.

Recommendation: Add deployment script which makes deployment flow transparent and clear, and also take care of ownership transfer to pre-deployed cold wallet.

ID 31: **Status: Acknowledged**

Medium | Lack of structure checksum scripts

Description: There's a noticeable lack of structure checksum scripts to verify proper ownership by cold wallet multi-signature contract and other important parameters initialization state.

Recommendation: Introduce checksum flow that will make sure all states are initialized with proper values and that contracts are under ownership of multi-signature wallet.

Fix feedback: Project decide not to fix. They are using a dedicated UI for this purpose.

ID 32: **Status: Acknowledged**

Discussion (Medium-High) | Upgradeability scheme

Description: Lack of upgradability leaves no possibility to fix bugs. The probability of bugs occurring always exists. The upgradability itself can be permissioned to a congress of cold storage wallets in varying degrees of decentralization to the project's likings. Without such schema, the project is left vulnerable to the likely case that bugs will be discovered or functionalities will be required to be added.

Complete Analysis

Recommendation: Consider implementing such a scheme to make architecture more secure.

Fix feedback: Project decided not to fix due to company policy

ID 33: **Status: Resolved**

Low | possible unintended state manipulation on swapBuyBackStatus

Present at: swapBuyBackStatus function @ DelayVault Contract

Description: The function doesn't have an explicit boolean parameter, which could lead to user errors and setting incorrect state, leading to potential loss of funds

Recommendation: Consider adding a boolean parameter to set the Allowance directly

ID 34: **Status: Resolved**

Medium | possible unintended state manipulation on swapTokenStatusFilter

Present at: swapBuyBackStatus function @ DelayVault Contract

Description: The function doesn't have an explicit boolean parameter, which could lead to user errors and setting incorrect state, leading to potential loss of funds

Recommendation: Consider adding a boolean parameter to set the Allowance directly

Complete Analysis

ID 35: **Status: Resolved**

Informational | function BuyBackTokens doesn't buy back tokens

Present at: BuyBackTokens function @ DelayManageable Contract

Description: mismatch between function name and function.

Recommendation: match function name to function

ID 36: **Status: Acknowledged**

Low | Use of require statements in ^0.8.4

Present at: Throughout the whole architecture.

Description: Since your chosen pragma version is above 0.8.4, which is the version that introduced solidity errors, you can greatly reduce gas consumption by introducing this concept instead of require statements. After introduction of 'errors', require statements are considered redundant as there is no additional value that they provide. You can still use them, but there are not any benefits from it and at this point it is a convention to use errors instead in versions ^0.8.4.

Recommendation: Replace require statements with solidity errors.

Fix feedback: Project decided to fix this issue in some of their further iterations.
There are too many changes required for a relatively low impact and Bscscan is not supporting this type of error at the moment.

Complete Analysis

ID 37: **Status: Resolved**

Low | In SetMinDelays function there is no MaxDelay validation

Present at: setMinDelays @ DelayManageable contract.

Description: When calling setMinDelay there is no validation that the highest delay values in the last delay index of the arrays is below the maxDelay value allowed.

Recommendation: add a require to check that the value in the last index of every delay array is below the MaxDelay value

Disclaimer:

DcentraLab Diligence (DD) has provided the code to the client as is and assumes no responsibility nor legal liability for any use client may do with the code. Any and all usage and/or deployment of the code provided by DcentraLab Diligence will be done solely by the client, at the sole discretion, responsibility, risk, and legal liability of the Client, and DD will not be held accountable or liable for any loss of funds, security exploits or incidents, or any other unintended or negative outcome that may occur in relation to the code provided by DD.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts DD to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model, or legal compliance.

This report and the provided code or services as part of the SOW pertaining to this report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should it be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. DD's position is that each company and individual are responsible for their own due diligence and continuous security. DD's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by DD are subject to dependencies and are under continuing development. You agree that your access and/or use, including but not limited to any services, code, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, DcentraLab Diligence (DD) HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, DD SPECIFICALLY DISCLAIMS

ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM THE COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

WITHOUT LIMITING THE FOREGOING, DD MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT / VERIFICATION REPORT, WORK PRODUCT, CODE OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET THE CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE.

WITHOUT LIMITATION TO THE DISCLAIMER [ASSESSMENT NAME] FOREGOING, DD PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET THE CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR-FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER DD NOR ANY OF DD'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION, CODE OR CONTENT PROVIDED THROUGH THE SERVICE. DD WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT OR CODE, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, CODE, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS," AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN THE CUSTOMER AND THE THIRD-PARTY
OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS. THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO THE CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT DD'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST DD WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS. THE REPRESENTATIONS AND WARRANTIES OF DD CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF THE CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST DD WITH RESPECT TO SUCH
REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.
FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS, CODE, OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL,
REGULATORY, OR OTHER ADVICE.

dcentralab.com/diligence

DcentraLab Diligence

Provided By DcentraLab Diligence on March 27, 2023