



SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

v1.0: 25. January, 2022

v1.1: 28. January, 2022

v1.2: 03. February, 2022

Audit

Security Assessment

11. November, 2022

For

CULT.DAO

Disclaimer	3
Description	5
Project Engagement	5
Logo	5
Contract Link	5
Methodology	7
Used Code from other Frameworks/Smart Contracts (direct imports)	8
Tested Contract Files	9
Source Lines	10
Risk Level	10
Capabilities	11
Inheritance Graph	12
CallGraph	13
Scope of Work/Verify Claims	14
Modifiers and public functions	21
Source Units in Scope	25
Critical issues	26
High issues	26
Medium issues	26
Low issues	26
Informational issues	26
Audit Comments	26
SWC Attacks	28
Unit test results	32

Disclaimer

SolidProof.io reports are not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team. SolidProof.io do not cover testing or auditing the integration with external contract or services (such as Unicrypt, Uniswap, PancakeSwap etc’...)

SolidProof.io Audits do not provide any warranty or guarantee regarding the absolute bug- free nature of the technology analyzed, nor do they provide any indication of the technology proprietors. SolidProof Audits should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

SolidProof.io Reports represent an extensive auditing 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. SolidProof’s position is that each company and individual are responsible for their own due diligence and continuous security. SolidProof in no way claims any guarantee of security or functionality of the technology we agree to analyze.

Version	Date	Description
1.0	25. January 2022	<ul style="list-style-type: none">• Layout project• Automated- /Manual-Security Testing• Summary
1.1	28. January 2022	<ul style="list-style-type: none">• Reaudit
1.2	03. February 2022	<ul style="list-style-type: none">• Reaudit changes
1.3	11. November 2022	<ul style="list-style-type: none">• Require statements have been added

Network

Ethereum

Website

<https://cultdao.io/>

Telegram

<https://t.me/cultdao>

Twitter

<https://twitter.com/wearecultdao>

Medium

<https://wearecultdao.medium.com/>

Discord

<https://discord.com/invite/hHDBvNnXqe>

Reddit

<http://reddit.com/r/cultdao/>

Description

The purpose of CULT is to empower and fund those building and contributing towards our decentralized future. Our society is built to make it as difficult as possible to break away from societal, economic and other norms,

Project Engagement

During the 25th of January 2022, **CultDAO Team** engaged Solidproof.io to audit smart contracts that they created. The engagement was technical in nature and focused on identifying security flaws in the design and implementation of the contracts. They provided Solidproof.io with access to their code repository and whitepaper.

Logo

The logo for CULT.DAO features the text "CULT.DAO" in a large, white, outlined, sans-serif font. The text is centered and has a subtle drop shadow. In the background, there is a faint, light gray watermark of a Bitcoin symbol, which is a circle with a vertical line through it and a grid pattern.

Contract Link

v1.0

- Github
 - <https://github.com/cultdao-developer/cultdao>
 - Commit: 003fc9119cd0fce1a56c3b53157d706c77800b5a

v1.2

- Github
 - <https://github.com/cultdao-developer/cultdao>
 - Commit: 007e946a420ff5f27ca73e76308331b1b55802af

V1.3

- Governance
 - Implementation
 - <https://etherscan.io/address/0xc6df585f8721bfafbb1580bd4034315696eab9ca#code>

Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system. Risk Level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
Critical	9 - 10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
High	7 – 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon as possible.
Medium	4 – 6.9	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.	Implementation of corrective actions in a certain period.
Low	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. To do so, reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as there were discovered.

Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
 - i) Review of the specifications, sources, and instructions provided to SolidProof to make sure we understand the size, scope, and functionality of the smart contract.
 - ii) Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
 - iii) Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to SolidProof describe.
2. Testing and automated analysis that includes the following:
 - i) Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
 - ii) Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

Used Code from other Frameworks/Smart Contracts (direct imports)

Imported packages:

Dependency / Import Path	Count
@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol	3
@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol	5
@openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol	5
@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol	3
@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol	2
@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol	1
@openzeppelin/contracts-upgradeable/token/ERC20/extensions/ERC20VotesCompUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/token/ERC20/extensions/ERC20VotesUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/token/ERC20/extensions/draft-ERC20PermitUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol	1
@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol	4

Tested Contract Files

This audit covered the following files listed below with a SHA-1 Hash.

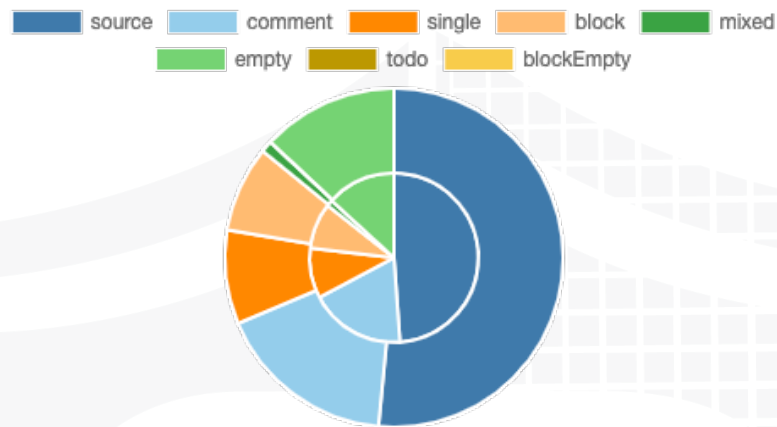
A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

v1.0

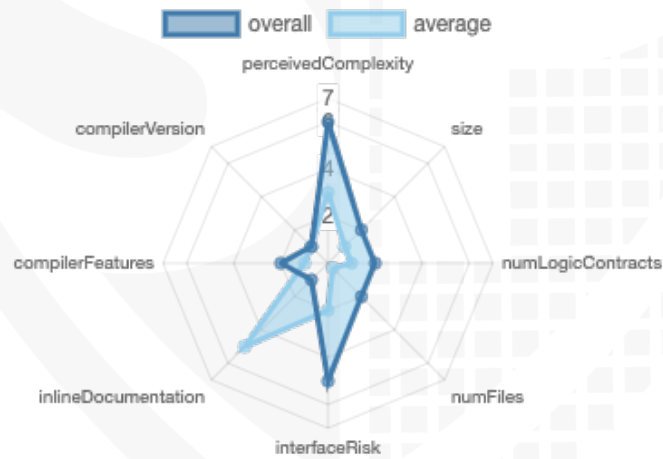
File Name	SHA-1 Hash
contracts/treasury.sol	f1a563489f19b6d44dd6206ed23389391649ecfc
contracts/dcult.sol	7f48779286bef63b631be4d46e4df82fbbdd2cc8
contracts/cult.sol	34bdad1e1cdc3737bce26970ceda92b6369e2f68
contracts/GovernorBravoInterfaces.sol	af29733fb1be18fad38c5f48efef618a4627153e
contracts/timelock.sol	a90b6764254751beac08ea4a4324c9f0d2884604
contracts/governance.sol	6e2ab39b2022697419b07d7883d772fe2e38d3f8

Metrics

Source Lines v1.0



Risk Level v1.0



Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	9	0	7	0

Exposed Functions

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

Version	Public	Payable
1.0	73	5

Version	External	Internal	Private	Pure	View
1.0	49	82	2	7	21

State Variables

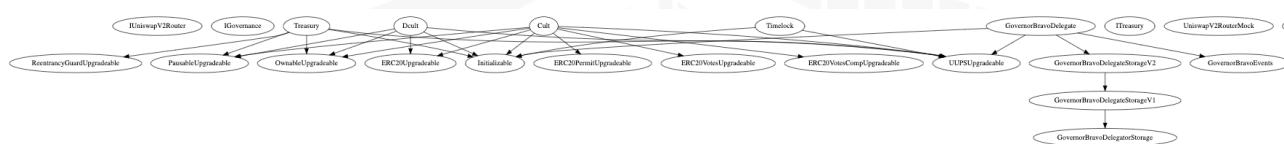
Version	Total	Public
1.0	60	59

Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	0.8.2 ^0.8.2	ABIEncoderV2	yes	yes (1 asm blocks)	

Version	Transfers ETH	Low-Level Calls	DelegateCall	Uses Hash Functions	EC Recover	New/Create/Create2
1.0	yes			yes	yes	

Inheritance Graph v1.0



TimelockInterface

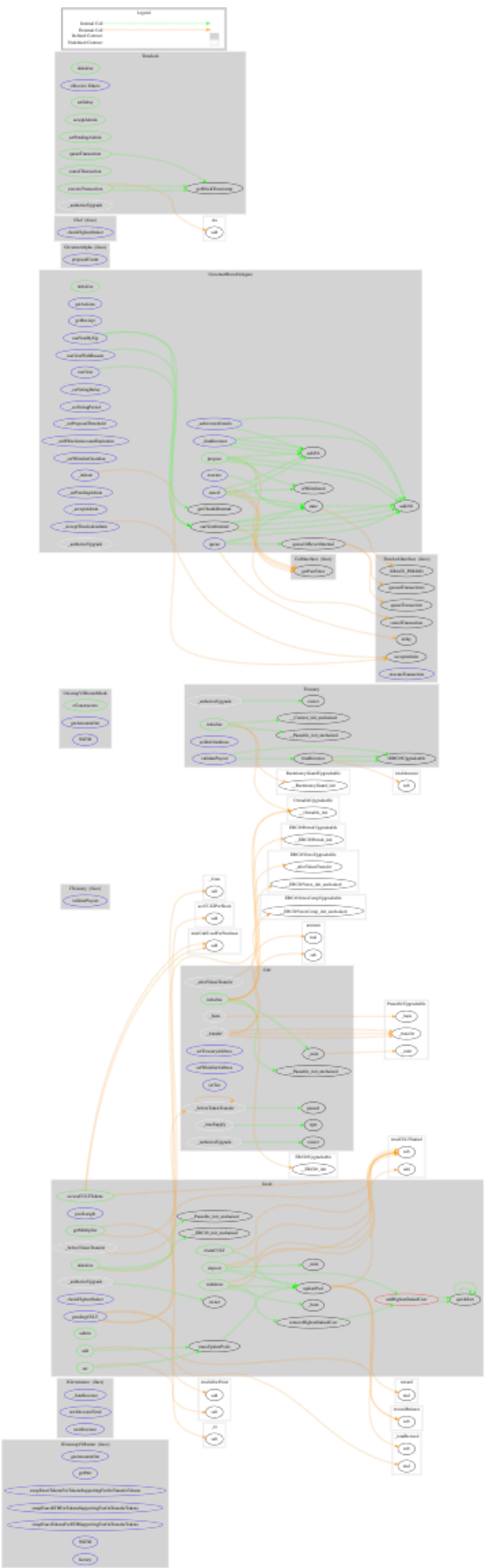
CultInterface

GovernorAlpha

Chef

CallGraph

v1.0



Scope of Work/Verify Claims

The above token Team provided us with the files that needs to be tested (Github, Bscscan, Etherscan, files, etc.). The scope of the audit is the main contract (usual the same name as team appended with .sol).

We will verify the following claims:

1. Correct implementation of Token standard
2. Deployer cannot mint any new tokens
3. Deployer cannot burn or lock user funds
4. Deployer cannot pause the contract
5. Overall checkup (Smart Contract Security)

Correct implementation of Token standard

Function	Description	Exist	Tested	Verified
TotalSupply	provides information about the total token supply	✓	✓	✓
BalanceOf	provides account balance of the owner's account	✓	✓	✓
Transfer	executes transfers of a specified number of tokens to a specified address	✓	✓	✓
TransferFrom	executes transfers of a specified number of tokens from a specified address	✓	✓	✓
Approve	allow a spender to withdraw a set number of tokens from a specified account	✓	✓	✓
Allowance	returns a set number of tokens from a spender to the owner	✓	✓	✓

Write functions of contract v1.0

▼ CULT	▼ DCULT	▼ GOVERNORBRAVODELEGATE
approve	accessCULTTokens	_acceptAdmin
decreaseAllowance	add	_AcceptTimelockAdmin
delegate	admin	_fundInvestee
delegateBySig	approve	_Initiate
increaseAllowance	claimCULT	_setInvesteeDetails
initialize	decreaseAllowance	_setPendingAdmin
permit	deposit	_setProposalThreshold
renounceOwnership	IncreaseAllowance	_setVotingDelay
setTax	initialize	_setVotingPeriod
setTreasuryAddress	massUpdatePools	_setWhitelistAccountExpiration
setWhitelistAddress	renounceOwnership	_setWhitelistGuardian
transfer	set	cancel
transferFrom	transfer	castVote
transferOwnership	transferFrom	castVoteBySig
upgradeTo	transferOwnership	castVoteWithReason
upgradeToAndCall	updatePool	execute
	upgradeTo	initialize
	upgradeToAndCall	propose
	withdraw	queue
		upgradeTo
		upgradeToAndCall

▼ TIMELOCK

acceptAdmin

cancelTransaction

executeTransaction

initialize

queueTransaction

setDelay

setPendingAdmin

upgradeTo

upgradeToAndCall

▼ TREASURY

initialize

renounceOwnership

setDAOAddress

transferOwnership

upgradeTo

upgradeToAndCall

validatePayout

Deployer cannot mint any new tokens

Name	Exist	Tested	Status
Deployer cannot mint	✓	✓	✓
Max / Total Supply	Can set while deploying		



Deployer cannot burn or lock user funds

Name	Exist	Tested	Status
Deployer cannot lock	✓	✓	✗
Deployer cannot burn	✓	✓	✓

Comments:

v1.0

- Deployer can lock user funds if address sender or receiver is not whitelisted address by setting tax amount to high value (e.g. 1000)

Deployer cannot pause the contract

Name	Exist	Tested	Status
Deployer cannot pause	✓	✓	✗

Comments:

v1.0

- Contract can be paused



Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

Legend


Attribute	Symbol
Verified / Checked	✓
Partly Verified	⚠
Unverified / Not checked	✗
Not available	—

Modifiers and public functions

v1.0

Cult

- initialize
 - initializer
- setTreasuryAddress
 - onlyOwner
- setWhitelistAddress
 - onlyOwner
- setTax
 - onlyOwner

- upgradeTo
 - onlyProxy
- upgradeToAndCall 
 - onlyProxy

- transfer
- approve
- transferFrom
- increaseAllowance
- decreaseAllowance

- permit

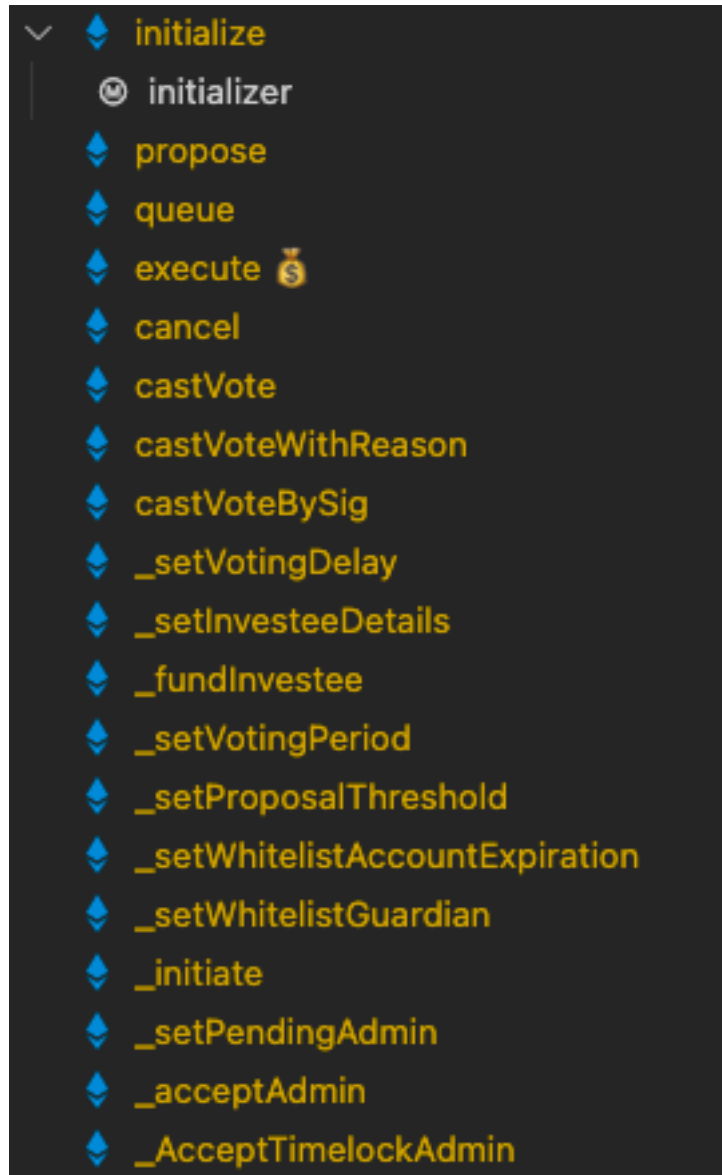
- delegate
- delegateBySig

- renounceOwnership
 - onlyOwner
- transferOwnership
 - onlyOwner

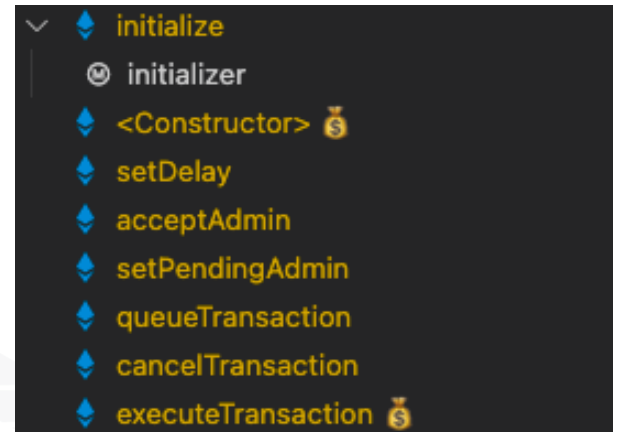
DCult

- initialize
 - initializer
- add
 - onlyOwner
- set
 - onlyOwner
- massUpdatePools
- updatePool
- deposit
- withdraw
- claimCULT
- accessCULTTokens
- admin

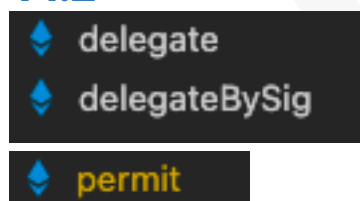
Governance



Timelock



V1.2



Comments

- Deployer can set following state variables without any limitations
 - Cult
 - tax
 - poolInfo[_pid].allocPoint
- Deployer can enable/disable following state variables
 - Cult














- whitelistedAddress[_whitelist]
- Cult
 - Exclude old treasury address from whitelistedAddress while setting new treasury address
- DCult
 - While deposit contract will mint new tokens
 - While withdraw contract will burn tokens
 - accessCULTTokens function can only be called from the admin address
 - Only Admin can set new admin address
- Governance
 - Only admin address can call following functions
 - _setVotingDelay
 - _setInvesteeDetails
 - _setVotingPeriod
 - _setProposalThreshold
 - _setWhitelistAccountExpiration
 - _setWhitelistGuardian
 - _initiate
 - _setPendingAdmin
 - Only treasury address can call following functions
 - _fundInfestee
 - Only whitelistGuardian address can call following functions
 - _setWhitelistAccountExpiration
 - Only pendingAdmin can call following functions
 - _acceptAdmin
 - _AcceptTimelockAdmin
- Timelock
 - Only admin address can call following functions
 - queueTransaction
 - cancelTransaction
 - executeTransaction
 - Only contract itself can call following functions
 - setDelay
 - Only pendingAdmin can call following functions
 - acceptAdmin
 - If admin is initialized only contract itself can call setPendingAdmining otherwise admin has to call setPendingAdmin function
- Treasury
 - onlyAdmin can call following functions
 - setDAOAddress

Please check if an OnlyOwner or similar restrictive modifier has been forgotten.



Source Units in Scope

v1.0

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
	contracts/treasury.sol	1	2	111	59	50	1	84	
	contracts/dcult.sol	1	————	398	377	261	89	203	
	contracts/cult.sol	1	1	106	83	68	1	64	————
	contracts/GovernorBravoInterfaces.sol	4	4	207	191	76	62	49	
	contracts/timelock.sol	1	————	122	122	89	2	81	
	contracts/governance.sol	1	————	489	489	265	155	237	
	Totals	9	7	1433	1321	809	310	718	

Legend

Attribute	Description
Lines	total lines of the source unit
nLines	normalized lines of the source unit (e.g. normalizes functions spanning multiple lines)
nSLOC	normalized source lines of code (only source-code lines; no comments, no blank lines)
Comment Lines	lines containing single or block comments
Complexity Score	a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...)

Audit Results

AUDIT PASSED

Critical issues

No critical issues

High issues

No high issues

Medium issues

No medium issues

Low issues

No low issues

Informational issues

No informational issues

Audit Comments

We recommend you to use the special form of comments (NatSpec Format, Follow link for more information <https://docs.soliditylang.org/en/v0.5.10/natspec-format.html>) for your contracts to provide rich documentation for functions, return variables and more. This helps investors to make clear what that variables, functions etc. do.

25. January 2022:

- Read whole report for more information

28. January 2022:

- Issues were fixed by the CULT.DAO team

03. February 2022:

- Interface Chef has been removed and checkHighestStaker has been moved to dCultInterface (was renamed from cultInterface)
- Following libraries has been imported to dCult
 - ERC20Upgradeable
 - draft-ERC20PermitUpgradeable
 - ERC20VotesUpgradeable
- Following functions were added and override
 - _mint
 - _burn
 - _afterTokenTransfer
 - _delegate

SWC Attacks

ID	Title	Relationships	Status
SW C-1 36	Unencrypted Private Data On-Chain	CWE-767: Access to Critical Private Variable via Public Method	PASSED
SW C-1 35	Code With No Effects	CWE-1164: Irrelevant Code	PASSED
SW C-1 34	Message call with hardcoded gas amount	CWE-655: Improper Initialization	PASSED
SW C-1 33	Hash Collisions With Multiple Variable Length Arguments	CWE-294: Authentication Bypass by Capture-replay	PASSED
SW C-1 32	Unexpected Ether balance	CWE-667: Improper Locking	PASSED
SW C-1 31	Presence of unused variables	CWE-1164: Irrelevant Code	PASSED
SW C-1 30	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	PASSED
SW C-1 29	Typographical Error	CWE-480: Use of Incorrect Operator	PASSED
SW C-1 28	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	PASSED

SW C-1 27	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	PASSED
SW C-1 25	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	PASSED
SW C-1 24	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	PASSED
SW C-1 23	Requirement Violation	CWE-573: Improper Following of Specification by Caller	PASSED
SW C-1 22	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	PASSED
SW C-1 21	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	PASSED
SW C-1 20	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	PASSED
SW C-11 9	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	PASSED
SW C-11 8	Incorrect Constructor Name	CWE-665: Improper Initialization	PASSED
SW C-11 7	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	PASSED

SW C-11 6	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SW C-11 5	Authorization through tx.origin	CWE-477: Use of Obsolete Function	PASSED
SW C-11 4	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	PASSED
SW C-11 3	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	PASSED
SW C-11 2	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SW C-11 1	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	PASSED
SW C-11 0	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	PASSED
SW C-1 09	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	PASSED
SW C-1 08	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED
SW C-1 07	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	PASSED
SW C-1 06	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	PASSED

SW C-1 05	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	PASSED
SW C-1 04	Unchecked Call Return Value	CWE-252: Unchecked Return Value	PASSED
SW C-1 03	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	PASSED
SW C-1 02	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	PASSED
SW C-1 01	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	PASSED
SW C-1 00	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED

Unit test results

69 passing (9s)

ERC20 tokens

Transfer functionality

- ✓ Transfer from Account 1 to Account 2 (64ms)
- ✓ Account 1 balance should be increased
- ✓ Transfer from Account 1 to Account 2
- ✓ Account 1 balance should be decreased

Transfer from

- ✓ WithOut Approve (42ms)
- ✓ Transfer from Account 1 to Account 2
- ✓ Account 1 balance should be increased
- ✓ Transfer from Account 1 to Account 2
- ✓ Account 1 balance should be decreased

Approve/Allowance

- ✓ Initial allowance will be 0
- ✓ Allowance increase when approve
- ✓ Increase Allowance
- ✓ Decrease Allowance
- ✓ Allowance will be 0 of tx account
- ✓ TransferFrom failed without allowance
- ✓ TransferFrom with allowance

dcult contract

Deployment

- ✓ Should set the right owner CULT token
- ✓ Should set the right owner of dCult

Add Cult pool

- ✓ Should revert if non owner tries to add pool
- ✓ Should set the right owner of dCult

Check dCult ERC20 token

- ✓ User should have should have dCULT token
- ✓ User should have should have total token supply
- ✓ User should have should have dCULT token after
- ✓ dCULT token should be burned on withdraw (46ms)
- ✓ Token should be non transferable

Check top stakers

- ✓ First User should have should be highest staker
- ✓ All user under the limit should be top staker
- ✓ User with more amount should remove the user with less staked amount (61ms)
- ✓ User shoul be removed from top staker list on withdrawal (76ms)

Check Cult distribution with one user

- ✓ User pending should be correct
- ✓ User can claim token
- ✓ Second cannot claim for deposit/stake after reward send to contract (51ms)
- ✓ User rewards will be claimed during deposit (40ms)

Check Cult distribution with multiple address user

- ✓ User first pending should be correct
- ✓ User second pending should be correct
- ✓ User first should claim half Reward
- ✓ User second should claim half Reward
- ✓ Second cannot claim extra rewards for deposit/stake after reward send to contract

(51ms)

- ✓ Second cannot claim after withdrawal (80ms)
- ✓ Third user can only claim rewards after deposit (119ms)

GovernorBravo_Propose

Non top staker tries to create proposal

✓

simple initialization

- ✓ ID is set to a globally unique identifier
- ✓ Proposer is set to the sender
- ✓ ForVotes and AgainstVotes are initialized to zero
- ✓ Executed and Canceled flags are initialized to false
- ✓ ETA is initialized to zero
- ✓ Targets, Values, Signatures, Calldatas are set according to parameters

This function must revert if

- ✓ the length of the values, signatures or calldatas arrays are not the same length, (63ms)
- ✓ or if that length is zero or greater than Max Operations.

Additionally, if there exists a pending or active proposal from the same proposer, we must revert.

- ✓ reverts with pending

GovernorBravo#state/1

- ✓ Invalid for proposal not found
- ✓ Pending
- ✓ Active
- ✓ Canceled

Caste Vote

- ✓ Caste Vote(True)
- ✓ Caste Vote(False)
- ✓ Caste Vote(Try to vote again)

Treasury contract

Deployment

- ✓ Should set the right owner CULT token
- ✓ Should set the right owner of governance

Check Fees

- ✓ 0.4 percent should be deducted on transfer from one account to another account
- ✓ No fees for whitelisted
- ✓ Only owners can whitelist

Add Investee

- ✓ Only admin/Timelock can add
- ✓ Should add from admin account

- ✓ Multiple investee

Fund Investee

- ✓ Only treasury can fund
- ✓ Should fund investee
- ✓ Should fund to other investee (52ms)
- ✓ Should update the mapping (73ms)



The logo features the words "Solid Proofed" in a white, elegant script font. The word "Solid" is positioned above "Proofed". Behind the text is a faint, stylized shield emblem with a grid-like pattern, rendered in a darker shade of blue. The entire composition is set against a solid blue background.

Solid
Proofed

Blockchain Security | Smart Contract Audits | KYC

A small horizontal bar representing the German flag, with black, red, and gold stripes.

MADE IN GERMANY