SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

Ridotto

Audit

Security Assessment
06. June, 2022

For

RIDOTTO

SolidProof_io

@solidproof_io

Disclaimer

Version	Date	Description
1.0	06. May 2022	• Layout project • Automated- /Manual-Security Testing • Summary

Network
Binance Smart Chain (BEP20)

Website
https://ridotto.io/

Telegram
https://t.me/ridotto_community

Twitter
https://twitter.com/ridotto_io

Reddit
https://www.reddit.com/r/ridotto_io/

Medium
https://ridotto-io.medium.com/

Discord
https://discord.com/invite/ridotto

Youtube
https://www.youtube.com/channel/UCxumaSF7pnu29f5kU4FAJbw

Description

TBA

Project Engagement

During the 2nd of June 2022, **Ridotto 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

RIDOTTO

Contract Link

v1.0

- Provided by files
    - https://drive.google.com/file/d/1Hwnmpm2b8pVp0DzZrCD93va83zcxAM4a/view?usp=sharing

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 aspossible.
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.

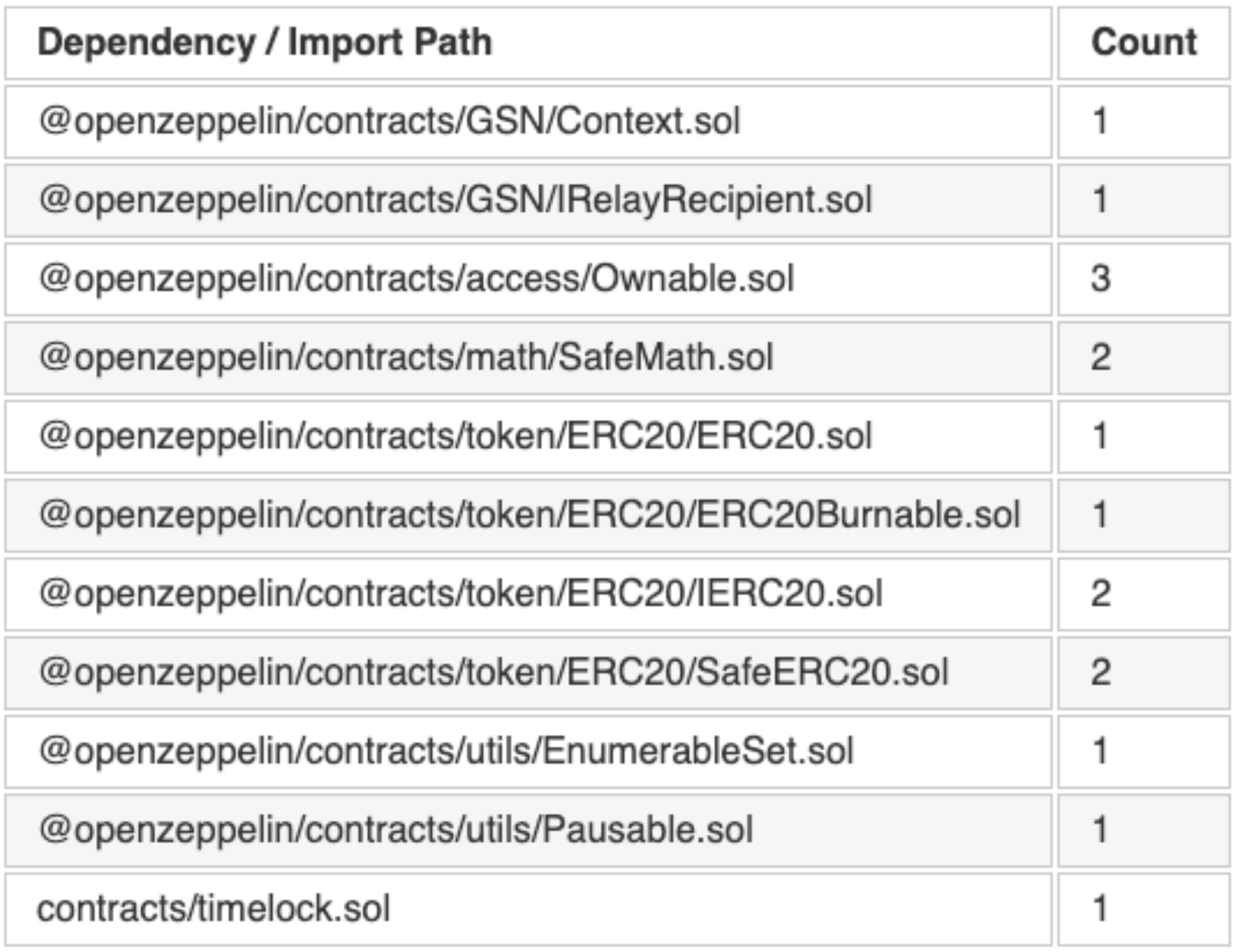
2. Testing and automated analysis that includes the following:
   i) Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
   ii) Symbolic execution, which is analysing a program to 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/GSN/Context.sol	1
@openzeppelin/contracts/GSN/IRelayRecipient.sol	1
@openzeppelin/contracts/access/Ownable.sol	3
@openzeppelin/contracts/math/SafeMath.sol	2
@openzeppelin/contracts/token/ERC20/ERC20.sol	1
@openzeppelin/contracts/token/ERC20/ERC20Burnable.sol	1
@openzeppelin/contracts/token/ERC20/IERC20.sol	2
@openzeppelin/contracts/token/ERC20/SafeERC20.sol	2
@openzeppelin/contracts/utils/EnumerableSet.sol	1
@openzeppelin/contracts/utils/Pausable.sol	1
contracts/timelock.sol	1

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.

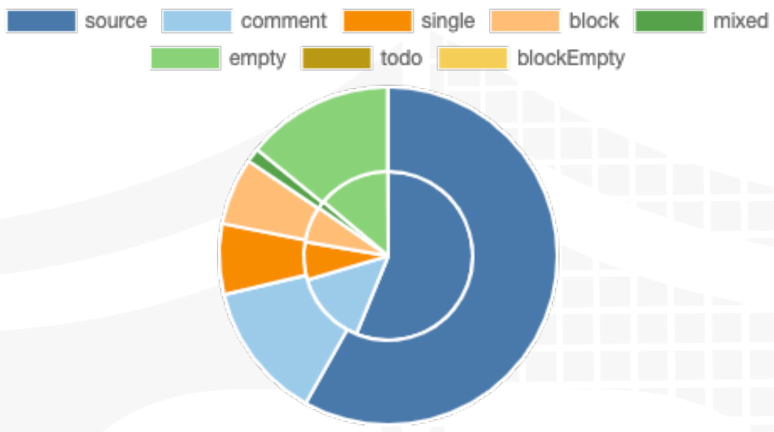
v1.0

File Name	SHA-1 Hash
contracts/interfaces/IGov.sol	0ef68f7c8380a233594de23757c360523e8b26c9
contracts/interfaces/IAutoVault.sol	58d5d8369ad2466f950cdfe7254e951a8a4a0daf
contracts/interfaces/IMasterChef.sol	791c15b762f72743420e6e60149c33d067f9ef59
contracts/MasterChef.sol	5cb5085c2f0153de1fc2c9e1d9ae886128afcfaa
contracts/AutoVault.sol	d20766872d7166ce8e0fdc2a2f58590de6265d28
contracts/Timelock.sol	eaee1ad8092b1143e80e8e497c3f73d817c64e83
contracts/GovernorAlpha.sol	d6fe947de83cb7d338bb8632aaeb31c2eb61bdfe
contracts/Gov.sol	6376931622e1905ca48af3349c2f49815b456bc5

Metrics

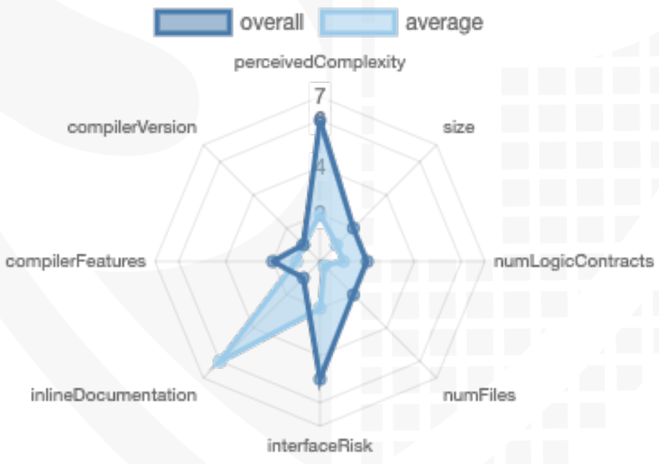
Source Lines

v1.0



Risk Level

v1.0



Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	5	0	4	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	105	2

Version	External	Internal	Private	Pure	View
1.0	60	92	0	13	30

State Variables

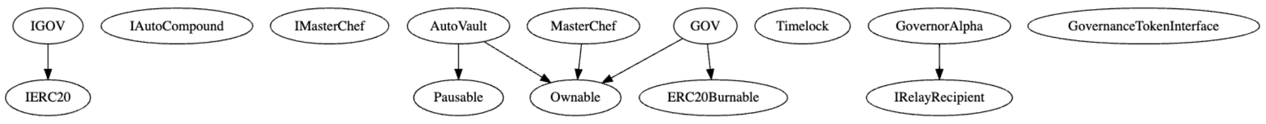
Version	Total	Public
1.0	46	44

Capabilities

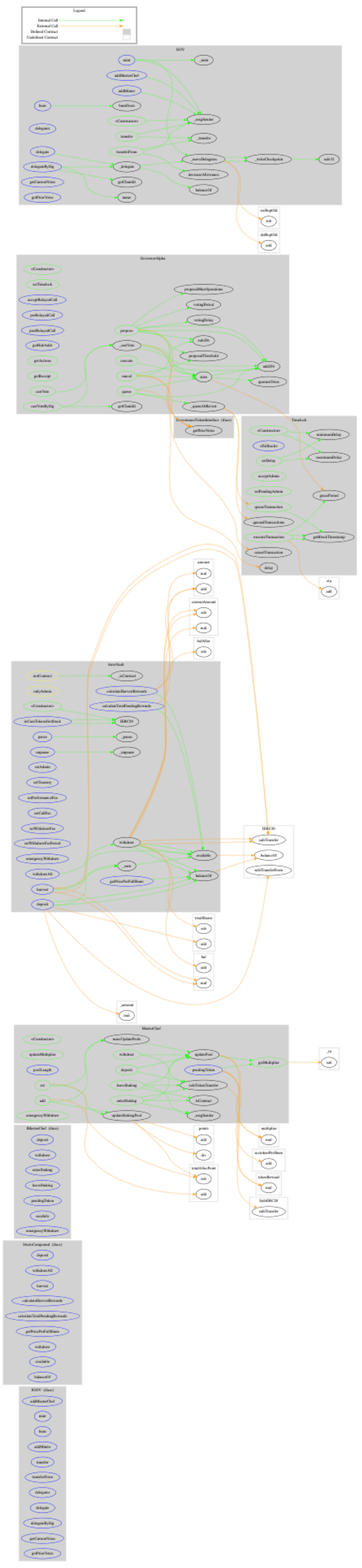
Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	`>=0.6.0` `<0.8.0`	`ABIEncoderV2`	`yes`	`yes` (4 asm blocks)	

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

Inheritance Graph
v1.0



CallGraph
v1.0



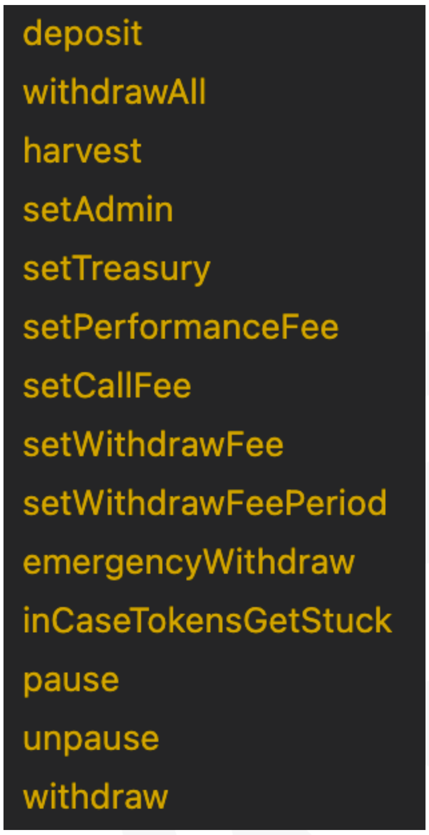
Scope of Work/Verify Claims

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).

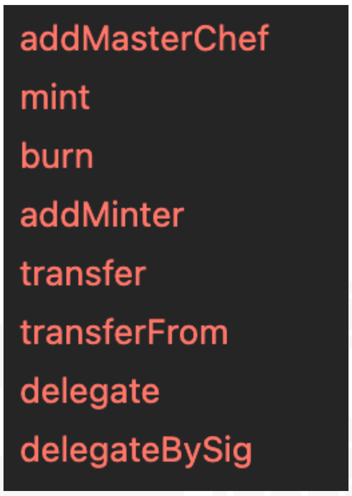
We will verify the following claims:

1. Overall checkup (Smart Contract Security)

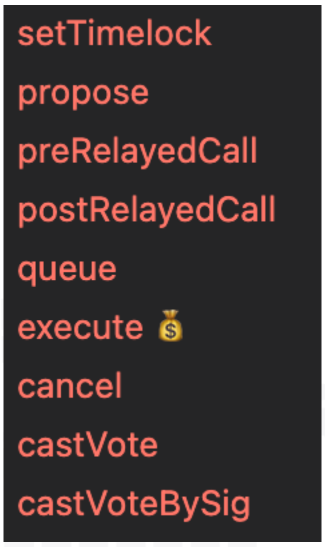
Write functions of contract v1.0



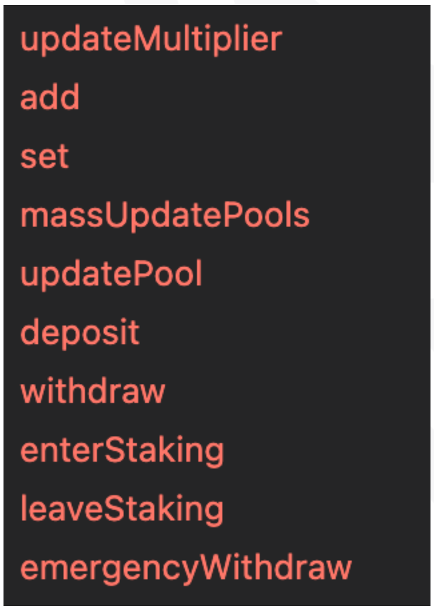
```
deposit
withdrawAll
harvest
setAdmin
setTreasury
setPerformanceFee
setCallFee
setWithdrawFee
setWithdrawFeePeriod
emergencyWithdraw
inCaseTokensGetStuck
pause
unpause
withdraw
```



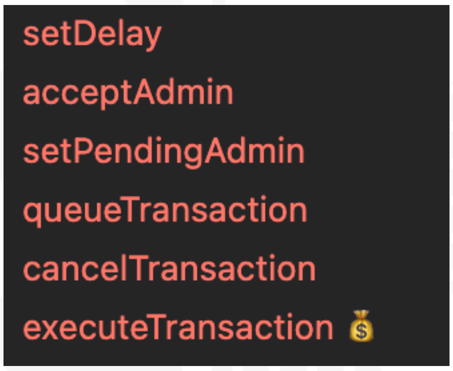
```
addMasterChef
mint
burn
addMinter
transfer
transferFrom
delegate
delegateBySig
```



```
setTimelock
propose
preRelayedCall
postRelayedCall
queue
execute 💰
cancel
castVote
castVoteBySig
```



```
updateMultiplier
add
set
massUpdatePools
updatePool
deposit
withdraw
enterStaking
leaveStaking
emergencyWithdraw
```



```
setDelay
acceptAdmin
setPendingAdmin
queueTransaction
cancelTransaction
executeTransaction 💰
```

Overall checkup (Smart Contract Security)

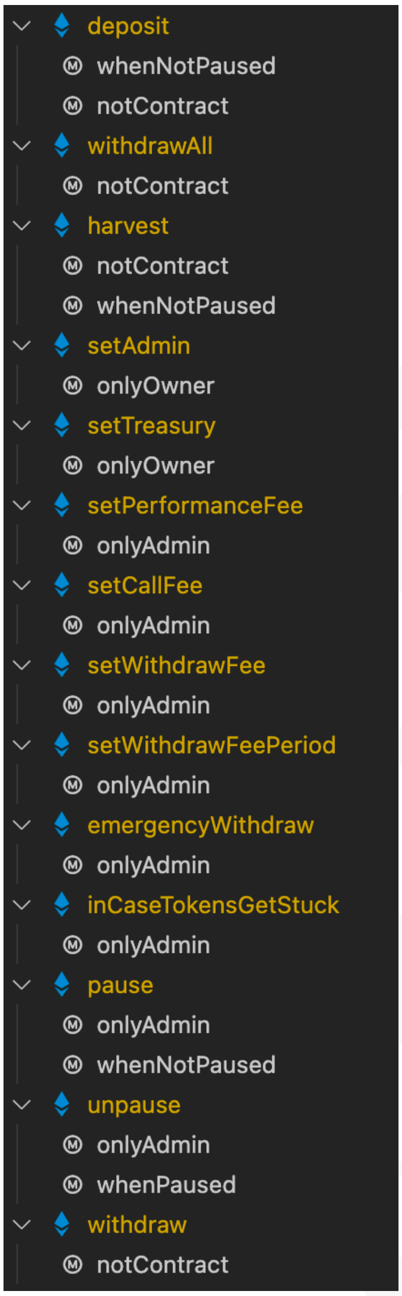
Tested	Verified
✓	✓

Legend

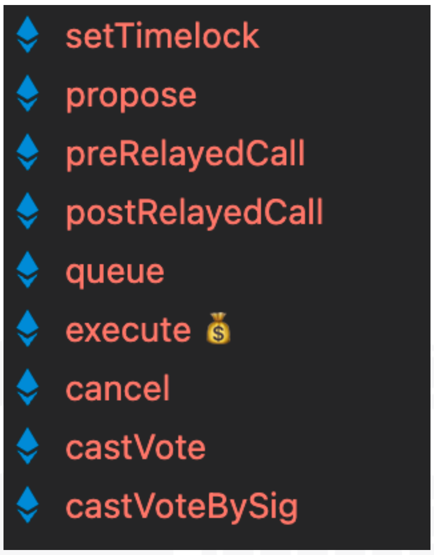
Attribute	Symbol
Verfified / Checked	✓
Partly Verified	⚑
Unverified / Not checked	✗
Not available	–

Modifiers and public functions

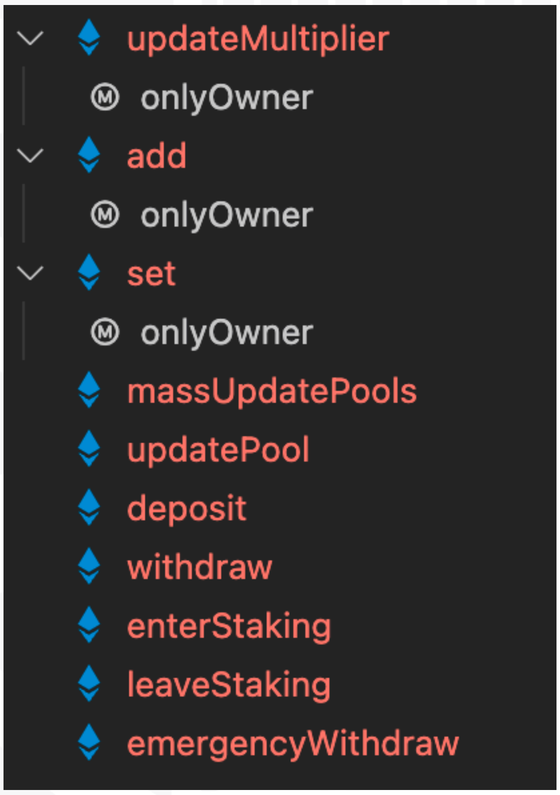
v1.0



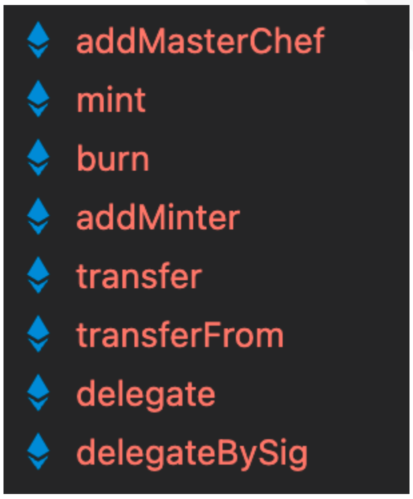
```
deposit
    whenNotPaused
    notContract
withdrawAll
    notContract
harvest
    notContract
    whenNotPaused
setAdmin
    onlyOwner
setTreasury
    onlyOwner
setPerformanceFee
    onlyAdmin
setCallFee
    onlyAdmin
setWithdrawFee
    onlyAdmin
setWithdrawFeePeriod
    onlyAdmin
emergencyWithdraw
    onlyAdmin
inCaseTokensGetStuck
    onlyAdmin
pause
    onlyAdmin
    whenNotPaused
unpause
    onlyAdmin
    whenPaused
withdraw
    notContract
```



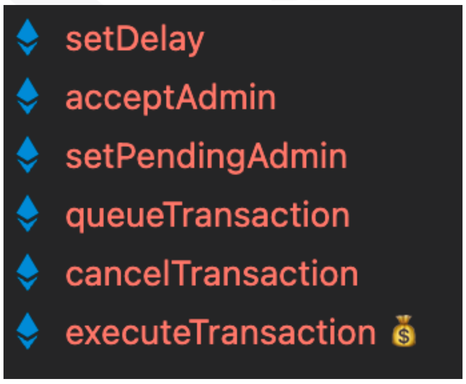
```
setTimelock
propose
preRelayedCall
postRelayedCall
queue
execute 💰
cancel
castVote
castVoteBySig
```



```
updateMultiplier
    onlyOwner
add
    onlyOwner
set
    onlyOwner
massUpdatePools
updatePool
deposit
withdraw
enterStaking
leaveStaking
emergencyWithdraw
```



```
addMasterChef
mint
burn
addMinter
transfer
transferFrom
delegate
delegateBySig
```



```
setDelay
acceptAdmin
setPendingAdmin
queueTransaction
cancelTransaction
executeTransaction 💰
```

Note: Not listed functions are imported from libraries

Comments

- *Deployer can set following state variables without any limitations*
  - BONUS_MULTIPLIER
    - If it's set to 0 every mathematical operations will be 0 as result which is multiplied by this variable
  - poolInfo[_pid].allocPoint

- *Deployer can enable/disable following state variables*
  - _paused
  - allowedMinters

- *Deployer can set following addresses*
  - admin
  - treasury
  - MasterChef
  - timelock
    - Can be set once

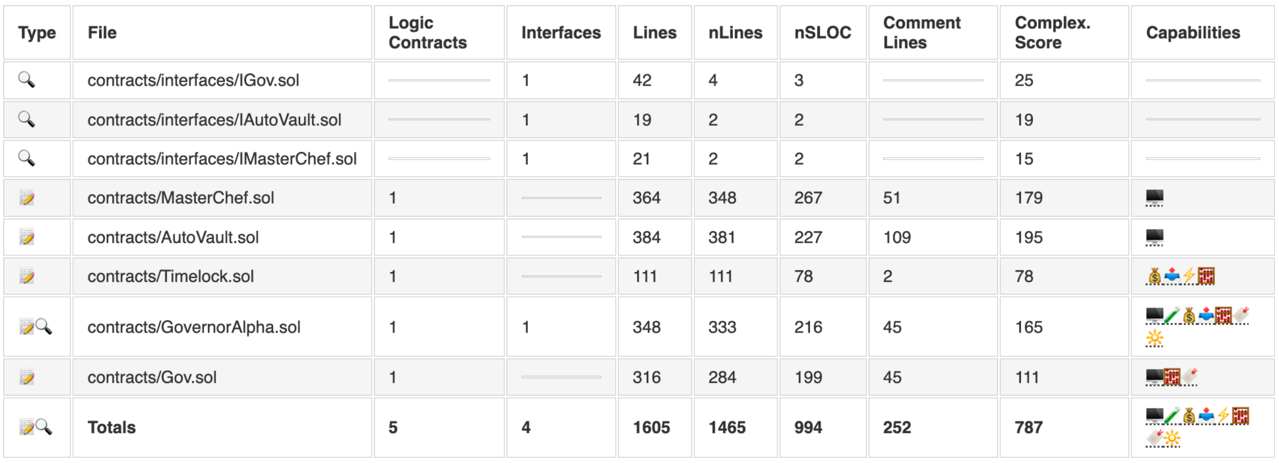
- *Existing Modifiers*
  - onlyAdmin
  - notContract

- We recommend to set state before transferring all the time
  - Look at MasterChef L341 and L343
- Gov.sol
  - Can mint new tokens by allowed minters
  - Anyone can burn from passed addresses
  - Owner can add new minter
- MasterChef
  - Owner can add new pool info
- General proposal for stakings: implement a delay between depositing and withdrawing of rewards because it can be used contracts to automate the functions calls with multicall contracts

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/interfaces/IGov.sol		1	42	4	3		25	
🔍	contracts/interfaces/IAutoVault.sol		1	19	2	2		19	
🔍	contracts/interfaces/IMasterChef.sol		1	21	2	2		15	
📝	contracts/MasterChef.sol	1		364	348	267	51	179	🖥️
📝	contracts/AutoVault.sol	1		384	381	227	109	195	🖥️
📝	contracts/Timelock.sol	1		111	111	78	2	78	💰📤⚡🧮
📝🔍	contracts/GovernorAlpha.sol	1	1	348	333	216	45	165	🖥️🧪💰📤🧮🔖☀️
📝	contracts/Gov.sol	1		316	284	199	45	111	🖥️🧮🔖
📝🔍	**Totals**	**5**	**4**	**1605**	**1465**	**994**	**252**	**787**	🖥️🧪💰📤⚡🧮🔖☀️

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

Issue	File	Type	Line	Description
#1	Main	Contract doesn’t import npm packages from source (like OpenZeppelin etc.)	-	We recommend to import all packages from npm directly without flatten the contract. Functions could be modified or can be susceptible to vulnerabilities
#2	All	A floating pragma is set	Top of file	The current pragma Solidity directive is „“>=0.6.0 <0.8.0 "".
#3	AutoVault	Missing Zero Address Validation (missing-zero-check)	71, 72	Check that the address is not zero
#4	Gov	Missing Zero Address Validation (missing-zero-check)	64	Check that the address is not zero
#5	GovernorAlpha	Missing Zero Address Validation (missing-zero-check)	137-141	Check that the address is not zero

#6	MasterChef	Missing Zero Address Validation (missing-zero-check)	77	Check that the address is not zero
#7	Timelock	Missing Zero Address Validation (missing-zero-check)	32, 83, 56	Check that the address is not zero
#8	GovernorAlpha	Declaration shadows an existing declaration	250	Modify the name of the local variable in that way that it is not shadowing another variable
#9	Gov	State variables shadowing	51	Rename the state variables that shadow another component
#10	Gov	Local variables shadowing	57, 92, 106	Rename the local variables that shadow another component
#11	AutoVault	Missing Events Arithmetic	167, 200	Emit an event for critical parameter changes
#12	MasterChef	Missing Events Arithmetic	118, 139-141, 98	Emit an event for critical parameter changes
#13	GovernorAlpha	Creating Proposal with mapping	167	Struct containing (nested) mapping cannot be constructed

Informational issues

Issue	File	Type	Line	Description
#1	Main	State variables that could be declared constant (constable-states)		Add the `constant` attributes to state variables that never change
#2	Gov	Error message is missing	65	Provide an error message for require statement
#3	All	SPDX-License	See description	SPDX-License-Identifier is missing. Add a license to the top of the source file.
#4	GovernorAlpha	Docstring	39-93, 134	Only state variables or file-level variables can have a docstring. Remove doctoring in that case
#5	GovernorAlpha	Wrong import	2	Fix import path.

#6	GovernorAlpha	Not completed function	191-219	Function is not completed. Remove or complete the function.
#7	GovernorAlpha	Missing using	Top of source file	SafeMath is missing in contract.
#8	IMasterChef	Pragma version missing	Top of source file	Provide a pragma version at the source of the file
#9	IGov	Pragma version missing	Top of source file	Provide a pragma version at the source of the file
#10	IAutoVault	Pragma version missing	Top of source file	Provide a pragma version at the source of the file
#11	Timelock	Deprecated	99	Using ".value(...)" is deprecated. Use "{value: ...}" instead.
#12	GovernorAlpha	Deprecated	244	Using ".value(...)" is deprecated. Use "{value: ...}" instead.

Note: Tested with pragma version 0.6.12

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.

07. June 2022:

- Read whole report and modifiers section for more information

SWC Attacks

ID	Title	Relationships	Status
SWC-136	Unencrypted Private Data On-Chain	CWE-767: Access to Critical Private Variable via Public Method	**PASSED**
SWC-135	Code With No Effects	CWE-1164: Irrelevant Code	**PASSED**
SWC-134	Message call with hardcoded gas amount	CWE-655: Improper Initialization	**PASSED**
SWC-133	Hash Collisions With Multiple Variable Length Arguments	CWE-294: Authentication Bypass by Capture-replay	**PASSED**
SWC-132	Unexpected Ether balance	CWE-667: Improper Locking	**PASSED**
SWC-131	Presence of unused variables	CWE-1164: Irrelevant Code	**NOT PASSED**
SWC-130	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	**PASSED**
SWC-129	Typographical Error	CWE-480: Use of Incorrect Operator	**PASSED**
SWC-128	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	**PASSED**

SWC-127	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	**PASSED**
SWC-125	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	**PASSED**
SWC-124	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	**PASSED**
SWC-123	Requirement Violation	CWE-573: Improper Following of Specification by Caller	**PASSED**
SWC-122	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	**PASSED**
SWC-121	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	**PASSED**
SWC-120	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	**PASSED**
SWC-119	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	**NOT PASSED**
SWC-118	Incorrect Constructor Name	CWE-665: Improper Initialization	**PASSED**
SWC-117	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	**PASSED**

SWC-116	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	**PASSED**
SWC-115	Authorization through tx.origin	CWE-477: Use of Obsolete Function	**PASSED**
SWC-114	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	**PASSED**
SWC-113	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	**PASSED**
SWC-112	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	**PASSED**
SWC-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	**PASSED**
SWC-110	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	**PASSED**
SWC-109	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	**PASSED**
SWC-108	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	**PASSED**
SWC-107	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	**PASSED**
SWC-106	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	**PASSED**

SWC-105	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	PASSED
SWC-104	Unchecked Call Return Value	CWE-252: Unchecked Return Value	PASSED
SWC-103	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	NOT PASSED
SWC-102	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	PASSED
SWC-101	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	PASSED
SWC-100	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED

SolidProof_io

@solidproof_io

Solid Proofed

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY