SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

Audit

Security Assessment
26. November, 2021

For

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	26. November 2021	• Layout project • Automated- /Manual-Security Testing • Summary

Network
Binance Smart Chain (BEP20)

Website
https://ibee.finance/#/

Telegram
https://t.me/ibeefinance

Twitter
https://twitter.com/ibeefinance

Description

iBee.Finance is a yield optimizer platform on Binance Smart Chain with an innovative vault system.
iBee Finance is Defi 2.0 (Next Generation Defi). We are revolutionizing the Defi industry. Not only are we the best single asset yield optimizer platform, but we are adding LP farming with IL(Impermanent loss) Protection. IL protection will help farmers keep their earnings and eliminate much of the risk to their principal.

Project Engagement

During the 18th of November 2021, **IBEE 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

Contract Link

v1.0

TBA

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.

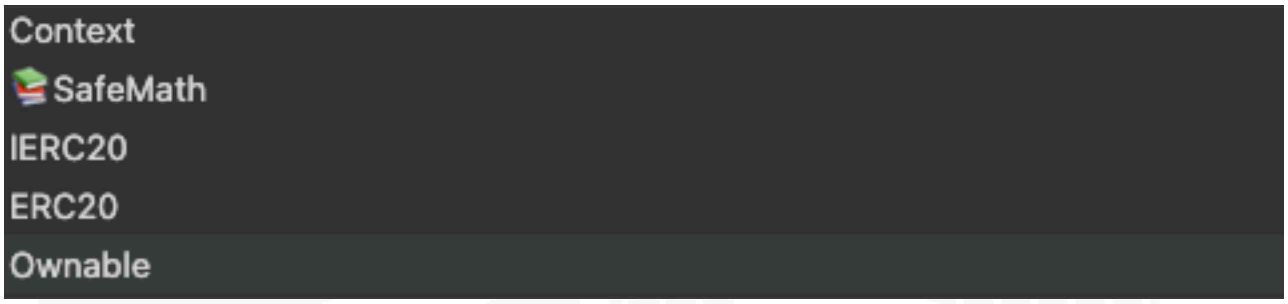
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.

4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

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

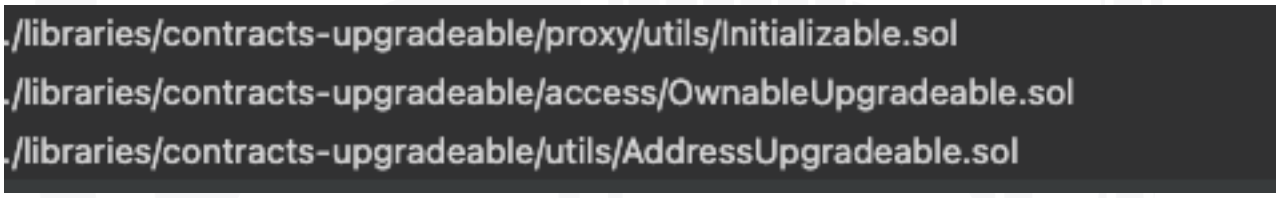
Imported packages:

- IBEEToken

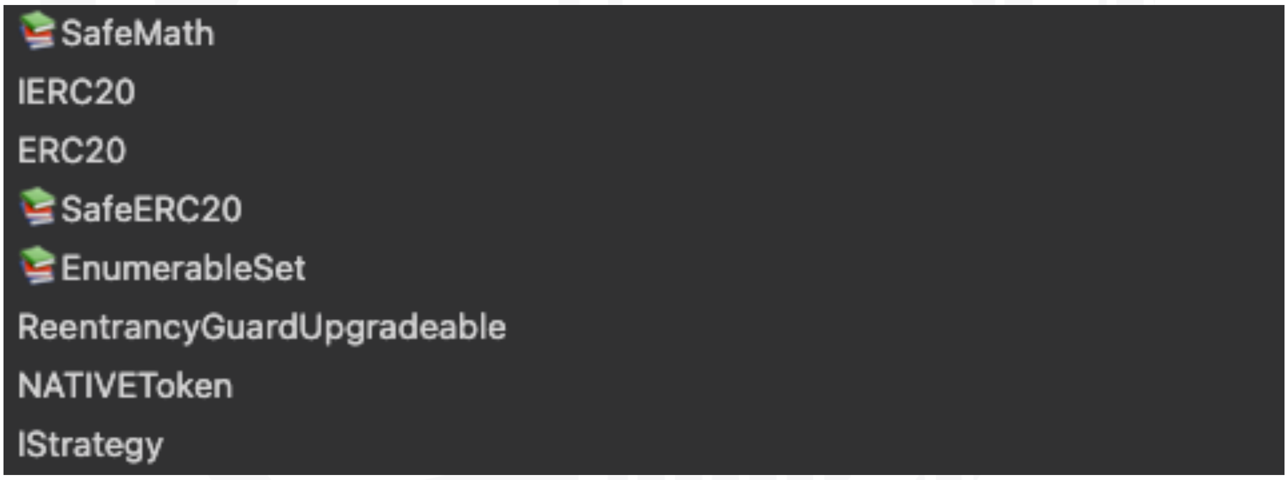


```
Context
SafeMath
IERC20
ERC20
Ownable
```

- MasterChef

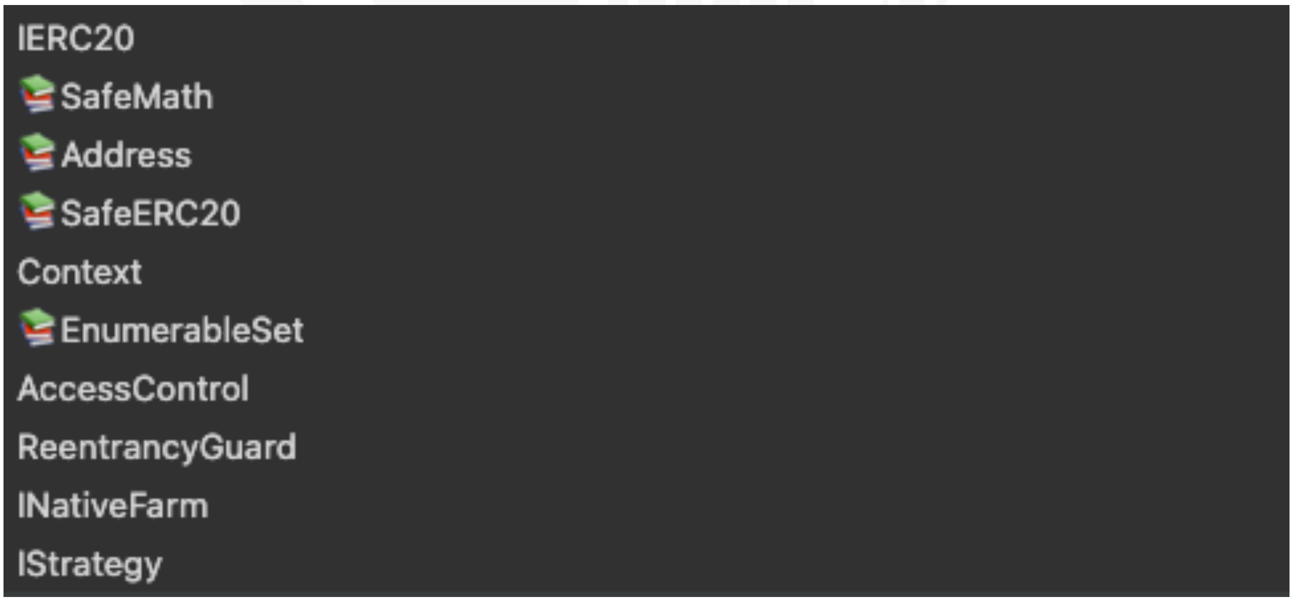


```
./libraries/contracts-upgradeable/proxy/utils/Initializable.sol
./libraries/contracts-upgradeable/access/OwnableUpgradeable.sol
./libraries/contracts-upgradeable/utils/AddressUpgradeable.sol
```



```
SafeMath
IERC20
ERC20
SafeERC20
EnumerableSet
ReentrancyGuardUpgradeable
NATIVEToken
IStrategy
```

- TimelockController



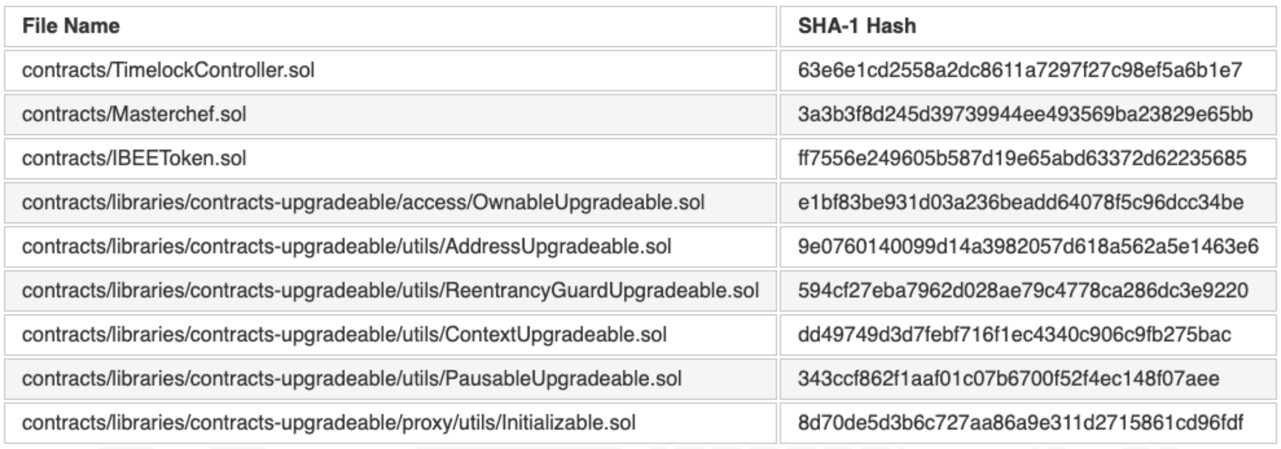
```
IERC20
SafeMath
Address
SafeERC20
Context
EnumerableSet
AccessControl
ReentrancyGuard
INativeFarm
IStrategy
```

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

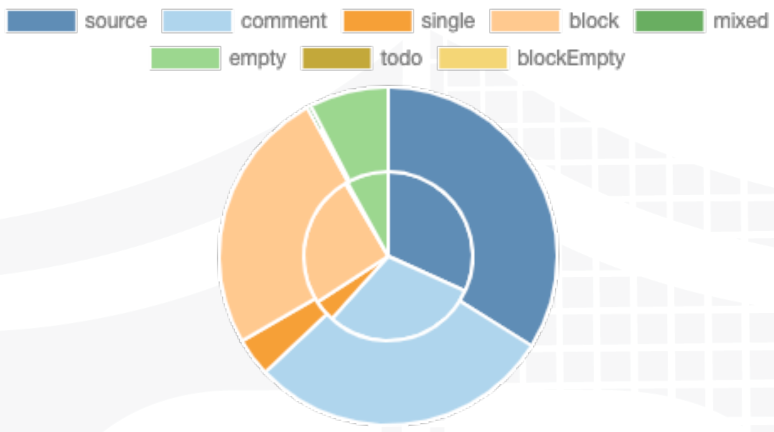


File Name	SHA-1 Hash
contracts/TimelockController.sol	63e6e1cd2558a2dc8611a7297f27c98ef5a6b1e7
contracts/Masterchef.sol	3a3b3f8d245d39739944ee493569ba23829e65bb
contracts/IBEEToken.sol	ff7556e249605b587d19e65abd63372d62235685
contracts/libraries/contracts-upgradeable/access/OwnableUpgradeable.sol	e1bf83be931d03a236beadd64078f5c96dcc34be
contracts/libraries/contracts-upgradeable/utils/AddressUpgradeable.sol	9e0760140099d14a3982057d618a562a5e1463e6
contracts/libraries/contracts-upgradeable/utils/ReentrancyGuardUpgradeable.sol	594cf27eba7962d028ae79c4778ca286dc3e9220
contracts/libraries/contracts-upgradeable/utils/ContextUpgradeable.sol	dd49749d3d7febf716f1ec4340c906c9fb275bac
contracts/libraries/contracts-upgradeable/utils/PausableUpgradeable.sol	343ccf862f1aaf01c07b6700f52f4ec148f07aee
contracts/libraries/contracts-upgradeable/proxy/utils/Initializable.sol	8d70de5d3b6c727aa86a9e311d2715861cd96fdf

Metrics

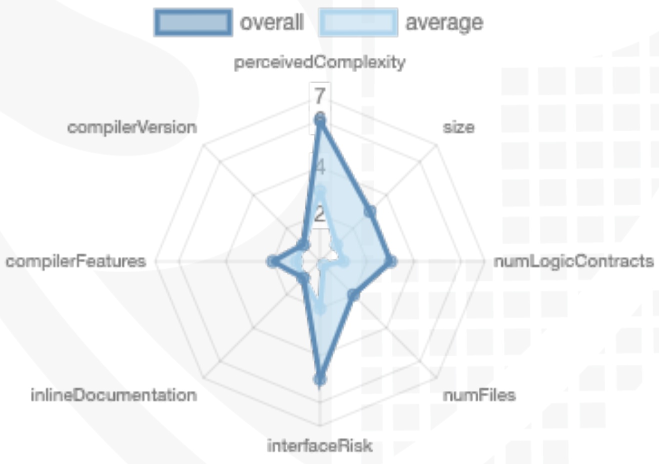
Source Lines

v1.0



Risk Level

v1.0



Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	5	9	6	12

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	124	7

Version	External	Internal	Private	Pure	View
1.0	42	269	20	28	76

State Variables

Version	Total	Public
1.0	54	20

Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	`^0.6.12` `0.6.12` `>=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`		

Scope of Work

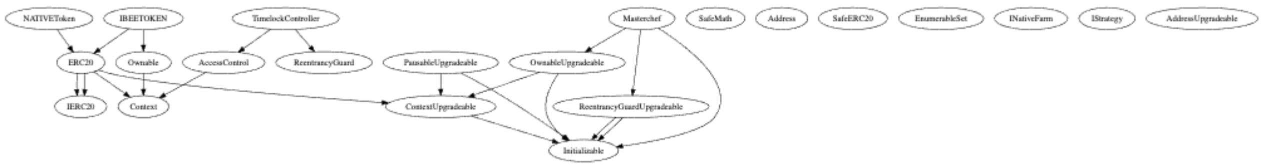
The above token Team provided us with the files that needs to be tested (Github, Bscscan, Etherscan, files, etc.). The scope of the audit is the main contract (usual the same name as team appended with .sol).

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)

Inheritance Graph

v1.0



Verify Claims

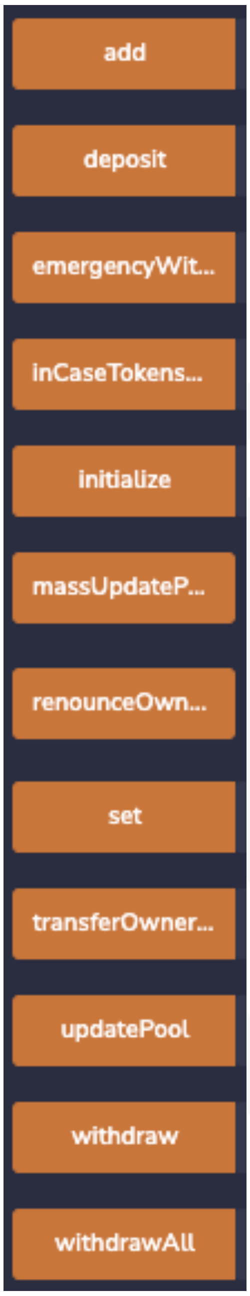
Correct implementation of Token standard

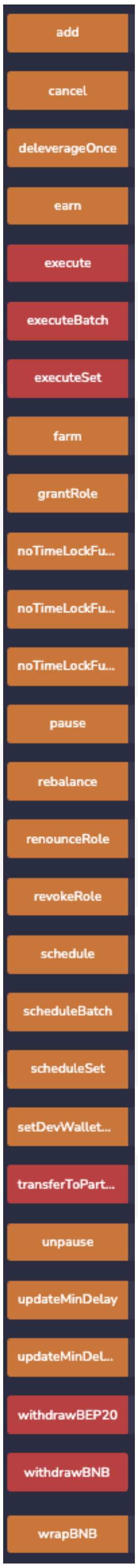
Tested	Verified
✓	✓

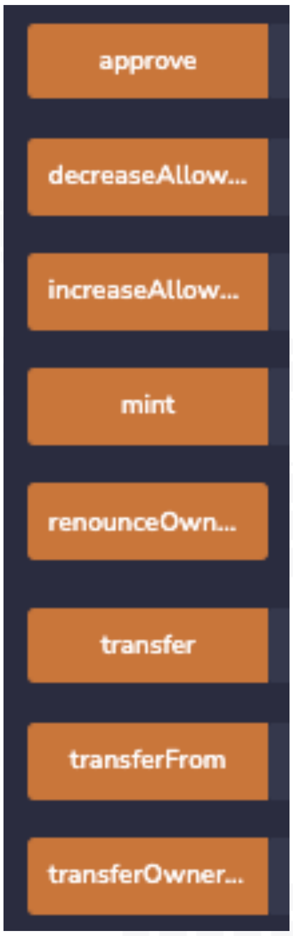
Function	Description	Exist	Tested	Verified
TotalSupply	provides information about the total token supply	✓	✓	✓
BalanceOf	provides account balance of the owner's account	✓	✓	✓
Transfer	executes transfers of a specified number of tokens to a specified address	✓	✓	✓
TransferFrom	executes transfers of a 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

MasterChef: TimelockController: IBEEToken:


add
deposit
emergencyWit...
inCaseTokens...
initialize
massUpdateP...
renounceOwn...
set
transferOwner...
updatePool
withdraw
withdrawAll


add
cancel
deleverageOnce
earn
execute
executeBatch
executeSet
farm
grantRole
noTimeLockFu...
noTimeLockFu...
noTimeLockFu...
pause
rebalance
renounceRole
revokeRole
schedule
scheduleBatch
scheduleSet
setDevWallet...
transferToPart...
unpause
updateMinDelay
updateMinDel...
withdrawBEP20
withdrawBNB
wrapBNB


approve
decreaseAllow...
increaseAllow...
mint
renounceOwn...
transfer
transferFrom
transferOwner...

Deployer cannot mint any new tokens

File	Name	Exist	Tested	Verified
MasterChef	Deployer cannot mint	–	–	–
TimelockController	Deployer cannot mint	–	–	–
IBEEToken	Deployer cannot mint	✓	✓	✗

Max / Total Supply: 1.000.000

Comments:

v1.0

- Masterchef using NATIVEToken with token address to mint tokens
    - With updatePool function

Deployer cannot burn or lock user funds

File	Name	Exist	Teste d	Verified
MasterChef	Deployer cannot lock	✓	✓	✓
	Deployer cannot burn	✓	✓	✓
TimelockContr oller	Deployer cannot lock	✓	✓	✓
	Deployer cannot burn	–	–	–
IBEEToken	Deployer cannot lock	✓	✓	✓
	Deployer cannot burn	✓	✓	✓

Deployer cannot pause the contract

File	Name	Exist	Tested	Verified
MasterChef	Deployer cannot pause	–	–	–
TimelockController	Deployer cannot pause	✓	✓	✗
IBEEToken	Deployer cannot pause	–	–	–

Overall checkup (Smart Contract Security)

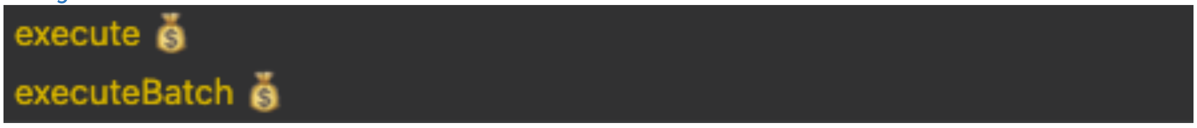
Tested	Verified
✓	✓

Legend

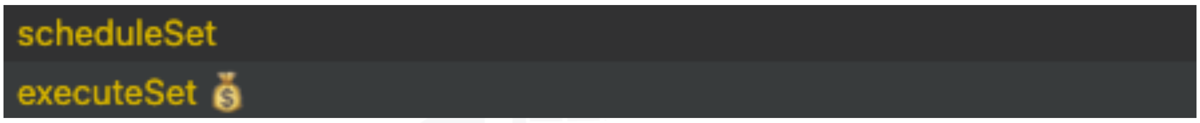
Attribute	Symbol
Verfified / Checked	✓
Partly Verified	⚑
Unverified / Not checked	✗
Not available	–

OnlyOwner functions

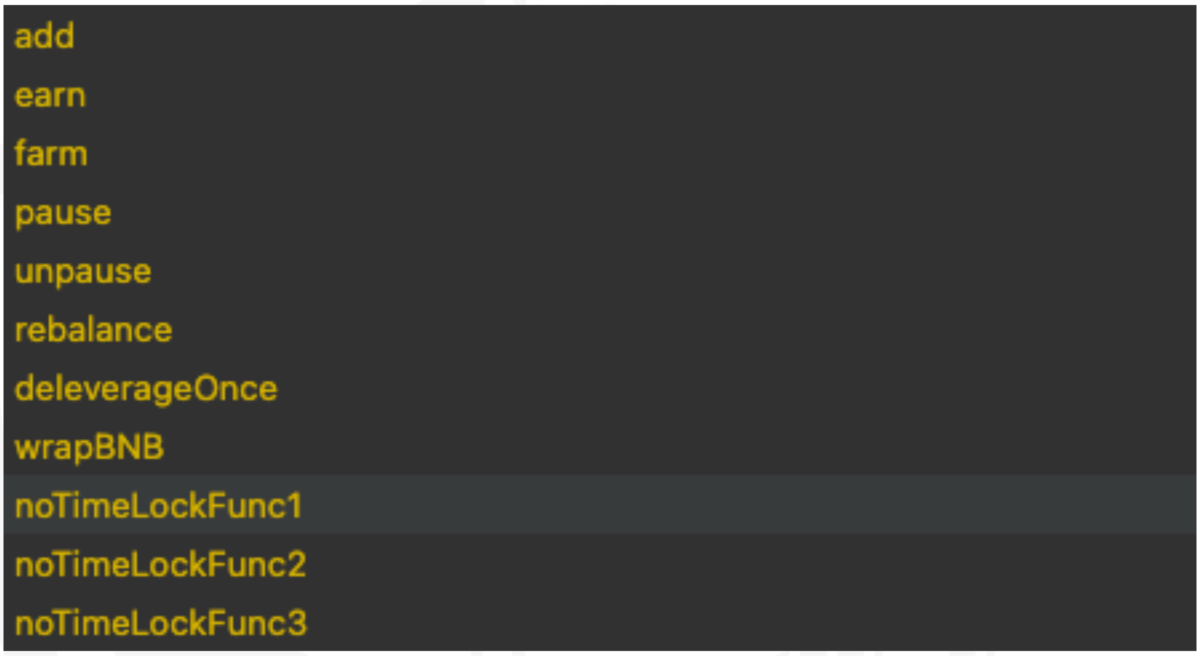
- TimelockController
  - onlyRole -> EXECUTOR_ROLE



```
execute 💰
executeBatch 💰
```

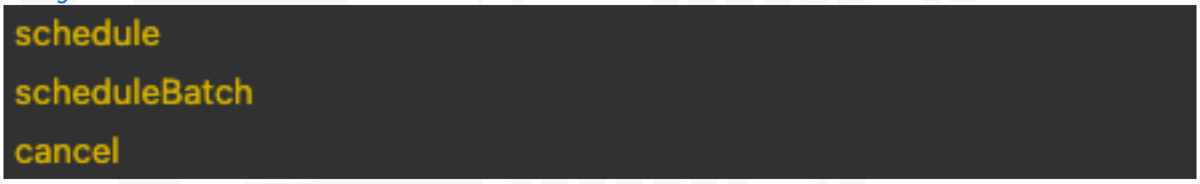


```
scheduleSet
executeSet 💰
```



```
add
earn
farm
pause
unpause
rebalance
deleverageOnce
wrapBNB
noTimeLockFunc1
noTimeLockFunc2
noTimeLockFunc3
```

  - onlyRole -> PROPOSER_ROLE



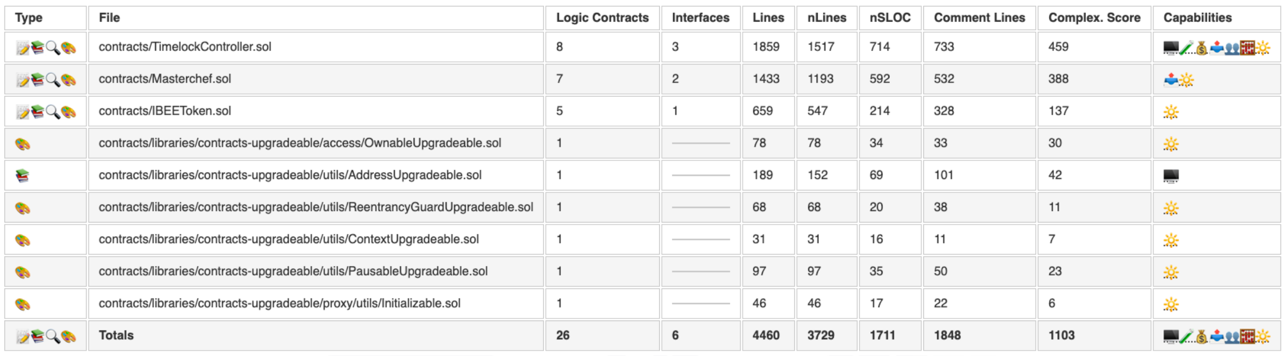
```
schedule
scheduleBatch
cancel
```

- IBEEToken
  - onlyOwner
    - Mint
- MasterChef
  - Initializer
    - Initialize
  - onlyOwner
    - add
    - Set
    - inCaseTokensGetStuck

CallGraph

Source Units in Scope

v1.0



Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝📚🔍🎨	contracts/TimelockController.sol	8	3	1859	1517	714	733	459	🖥🧪💰📤👥🧮🔆
📝📚🔍🎨	contracts/Masterchef.sol	7	2	1433	1193	592	532	388	📤🔆
📝📚🔍🎨	contracts/IBEEToken.sol	5	1	659	547	214	328	137	🔆
🎨	contracts/libraries/contracts-upgradeable/access/OwnableUpgradeable.sol	1	——	78	78	34	33	30	🔆
📚	contracts/libraries/contracts-upgradeable/utils/AddressUpgradeable.sol	1	——	189	152	69	101	42	🖥
🎨	contracts/libraries/contracts-upgradeable/utils/ReentrancyGuardUpgradeable.sol	1	——	68	68	20	38	11	🔆
🎨	contracts/libraries/contracts-upgradeable/utils/ContextUpgradeable.sol	1	——	31	31	16	11	7	🔆
🎨	contracts/libraries/contracts-upgradeable/utils/PausableUpgradeable.sol	1	——	97	97	35	50	23	🔆
🎨	contracts/libraries/contracts-upgradeable/proxy/utils/Initializable.sol	1	——	46	46	17	22	6	🔆
📝📚🔍🎨	**Totals**	**26**	**6**	**4460**	**3729**	**1711**	**1848**	**1103**	🖥🧪💰📤👥🧮🔆

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 found -

High issues

- no high issues found -

Medium issues

- no medium issues found -

Low issues

Issue	File	Type	Line	Description
#1	IBEEToken	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	MasterChef	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
#3	TimelockController	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
#4	IBEEToken	A floating pragma is set	1	The current pragma Solidity directive is „"^0.6.12"".
#5	MasterChef	A floating pragma is set	1	The current pragma Solidity directive is „"^0.6.12"".

#6	TimelockController	A floating pragma is set	1	The current pragma Solidity directive is „“^0.6.12”".
#7	MasterChef	Missing Zero Address Validation (missing-zero-check)	1161, 1160	Check that the address is not zero

Informational issues

Issue	File	Type	Line	Description
#1	IBEEToken	SPDX license identifier not provided in source file	-	Consider adding a SPDX license in source file
#2	MasterChef	SPDX license identifier not provided in source file	-	Consider adding a SPDX license in source file
#3	TimelockController	SPDX license identifier not provided in source file	-	Consider adding a SPDX license in source file
#4	IBEEToken	Functions that are not used	520, 565	Remove unused functions

Commented Code exist

There are some instances of code being commented out in the following files that should be removed:

File	Line	Comment
TimelockController	206	`// assert(a == b * c + a % b); // There is no case in which this doesn't hold`

TimelockController	1385	`// register proposers`
		`// for (uint256 i = 0; i < proposers.length; ++i) {`
		`//     _setupRole(PROPOSER_ROLE, proposers[i]);`
		`// }`

TimelockController	1391	`// // register executors`
		`// for (uint256 i = 0; i < executors.length; ++i) {`
		`//     _setupRole(EXECUTOR_ROLE, executors[i]);`
		`// }`

Recommendation

Remove the commented code, or address them properly.

Audit Comments

19. November 2021:

- For more information please read the report and do your own research

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	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	**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	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	CWE-330: Use of Insufficiently Random Values	PASSED
SWC-119	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	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

Solid Proofed

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY