SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

Audit

Security Assessment

10. January, 2022

For

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

Scope of Work 13

Inheritance Graph 13

Verify Claims 14

Modifiers 17

CallGraph 21

Source Units in Scope 22

Critical issues 23

High issues 23

Medium issues 23

Low issues 23

Informational issues 24

Commented Code exist 24

Audit Comments 24

SWC Attacks 25

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	10. January 2022	• Layout project • Automated- /Manual-Security Testing • Summary

Network

Fantom Opera

Website

https://devilfinance.io/

Telegram

https://t.me/devilfinancechat

Twitter

https://twitter.com/DevilFinance_io

Github

https://github.com/devilfinanceio/devil-official-contracts

Discord

https://discord.gg/wWX6A52YZ7

Description

In its essence Devil Finance is a **Decentralized Yield Optimizer platform** that allows its users to earn compound interest on their crypto holdings. Devil Finance runs on the Fantom blockchain and offers one of the leading market yield strategies and operations.

Our mission is to give our users (beginner & professional) the opportunity to **save, grow, and accumulate assets** for the future **in a safe, efficient, and user friendly way**.

We will be providing access to collections of non-fiat inflationary assets. We **provide rewards in return for utilizing our platform**, namely in the form of 'staking' tokens and our native token (DEVIL).

We will collect tokens from many users and stake them on a mass scale on their behalf. Through more frequent compounding, more efficient gas utilization, and other creative automations we will **save users fees and most of all maximize their returns**.

Project Engagement

During the 8th of January 2022, **Devil 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

- As file

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.

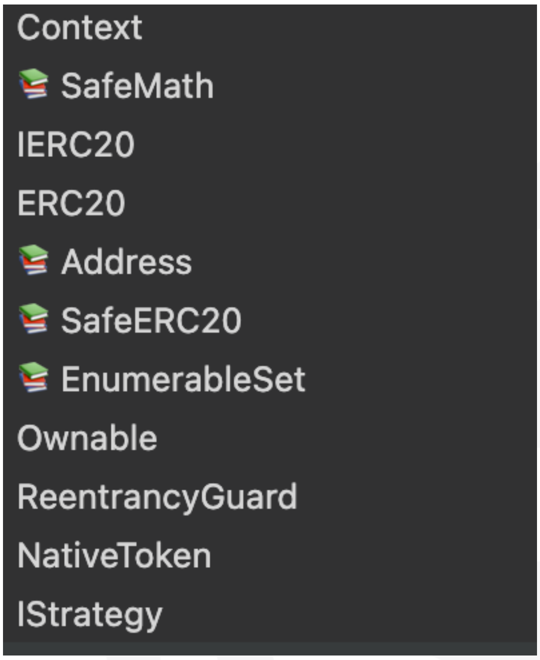
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.

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

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

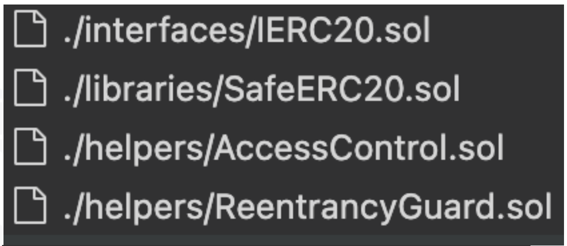
Imported packages:

NativeFarm

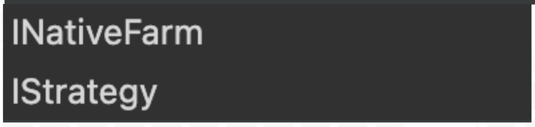


Context
SafeMath
IERC20
ERC20
Address
SafeERC20
EnumerableSet
Ownable
ReentrancyGuard
NativeToken
IStrategy

TimelockController

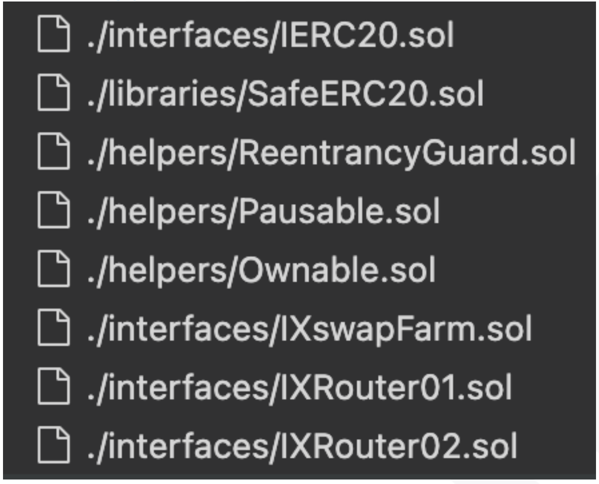


./interfaces/IERC20.sol
./libraries/SafeERC20.sol
./helpers/AccessControl.sol
./helpers/ReentrancyGuard.sol

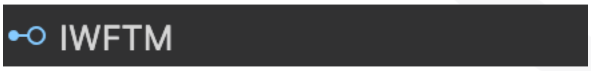


INativeFarm
IStrategy

Strategy



./interfaces/IERC20.sol
./libraries/SafeERC20.sol
./helpers/ReentrancyGuard.sol
./helpers/Pausable.sol
./helpers/Ownable.sol
./interfaces/IXswapFarm.sol
./interfaces/IXRouter01.sol
./interfaces/IXRouter02.sol



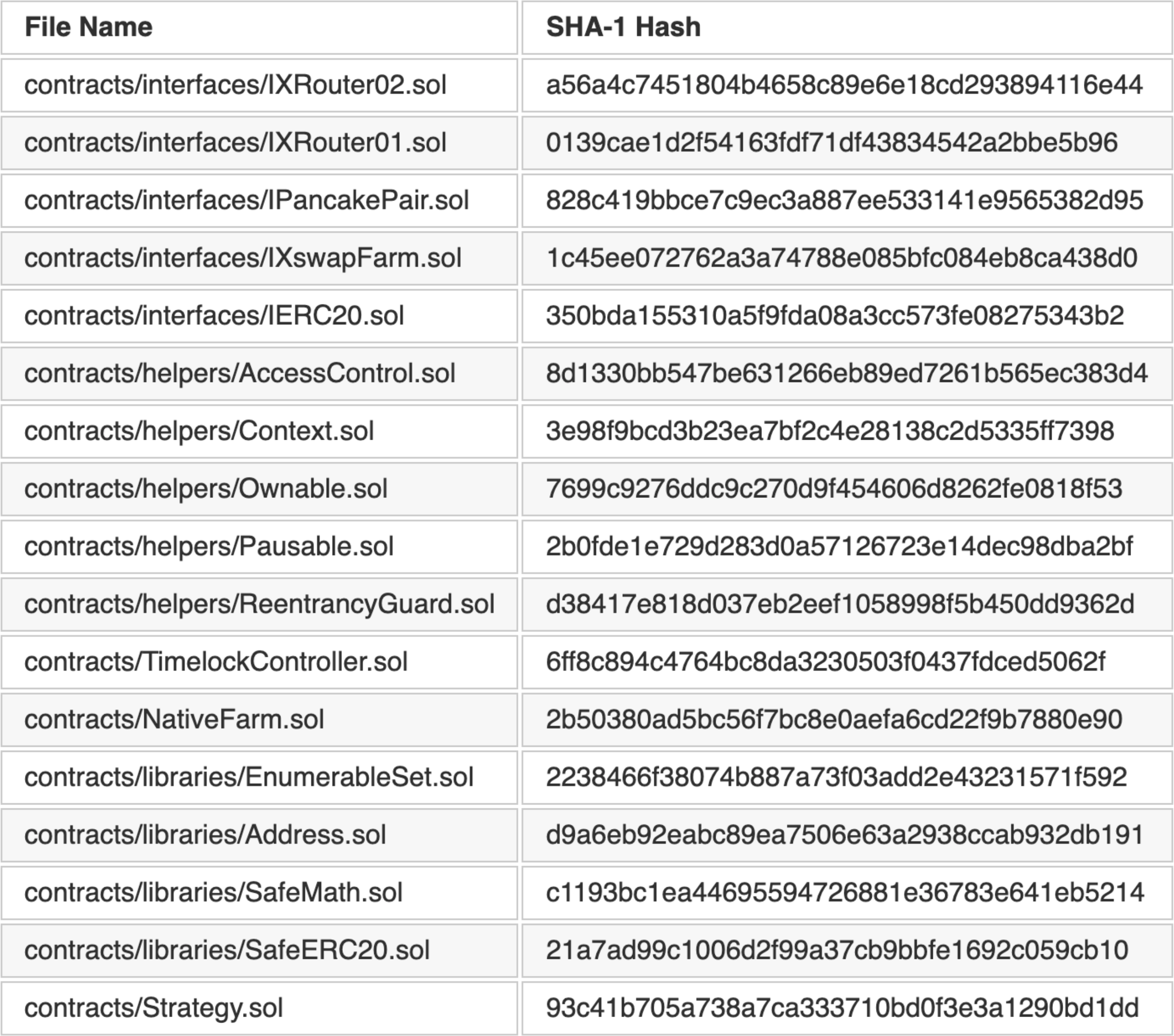
IWFTM

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

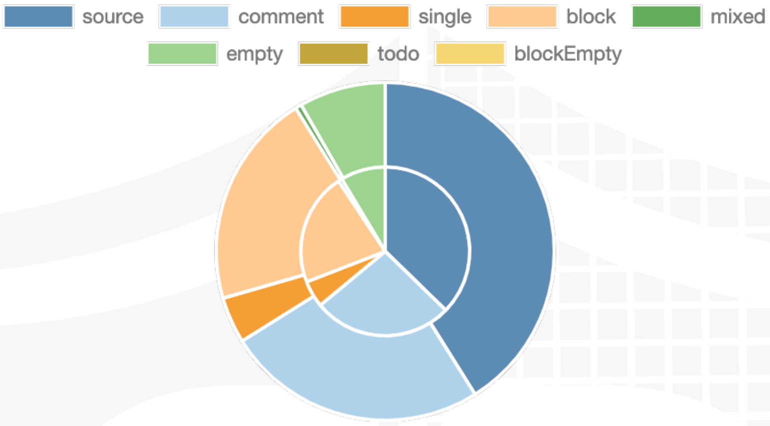


File Name	SHA-1 Hash
contracts/interfaces/IXRouter02.sol	a56a4c7451804b4658c89e6e18cd293894116e44
contracts/interfaces/IXRouter01.sol	0139cae1d2f54163fdf71df43834542a2bbe5b96
contracts/interfaces/IPancakePair.sol	828c419bbce7c9ec3a887ee533141e9565382d95
contracts/interfaces/IXswapFarm.sol	1c45ee072762a3a74788e085bfc084eb8ca438d0
contracts/interfaces/IERC20.sol	350bda155310a5f9fda08a3cc573fe08275343b2
contracts/helpers/AccessControl.sol	8d1330bb547be631266eb89ed7261b565ec383d4
contracts/helpers/Context.sol	3e98f9bcd3b23ea7bf2c4e28138c2d5335ff7398
contracts/helpers/Ownable.sol	7699c9276ddc9c270d9f454606d8262fe0818f53
contracts/helpers/Pausable.sol	2b0fde1e729d283d0a57126723e14dec98dba2bf
contracts/helpers/ReentrancyGuard.sol	d38417e818d037eb2eef1058998f5b450dd9362d
contracts/TimelockController.sol	6ff8c894c4764bc8da3230503f0437fdced5062f
contracts/NativeFarm.sol	2b50380ad5bc56f7bc8e0aefa6cd22f9b7880e90
contracts/libraries/EnumerableSet.sol	2238466f38074b887a73f03add2e43231571f592
contracts/libraries/Address.sol	d9a6eb92eabc89ea7506e63a2938ccab932db191
contracts/libraries/SafeMath.sol	c1193bc1ea44695594726881e36783e641eb5214
contracts/libraries/SafeERC20.sol	21a7ad99c1006d2f99a37cb9bbfe1692c059cb10
contracts/Strategy.sol	93c41b705a738a7ca333710bd0f3e3a1290bd1dd

Metrics

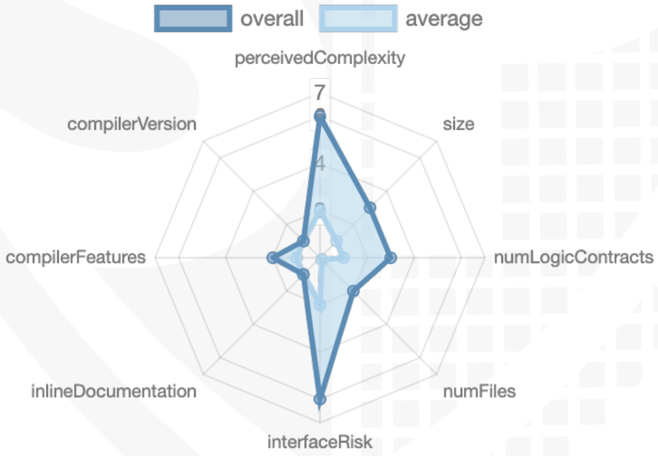
Source Lines

v1.0



Risk Level

v1.0



Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	4	8	10	9

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	176	11

Version	External	Internal	Private	Pure	View
1.0	104	280	20	31	82

State Variables

Version	Total	Public
1.0	76	58

Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts

1.0	`0.6.12`
	`>=0.6.12`
	`^0.6.12`
		`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

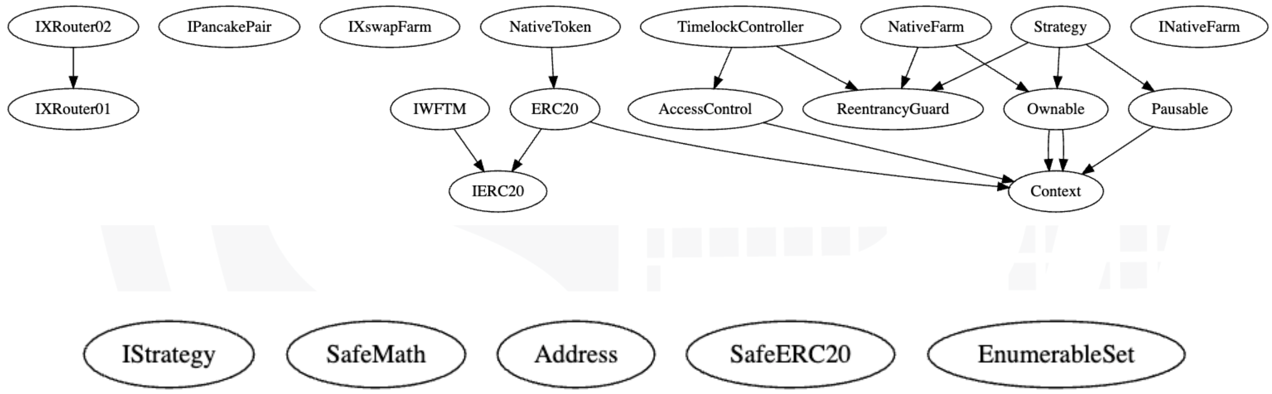
The above token Team provided us with the files that needs to be tested (Github, Bscscan, Etherscan, files, etc.). The scope of the audit is the main contract (usual the same name as team appended with .sol).

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

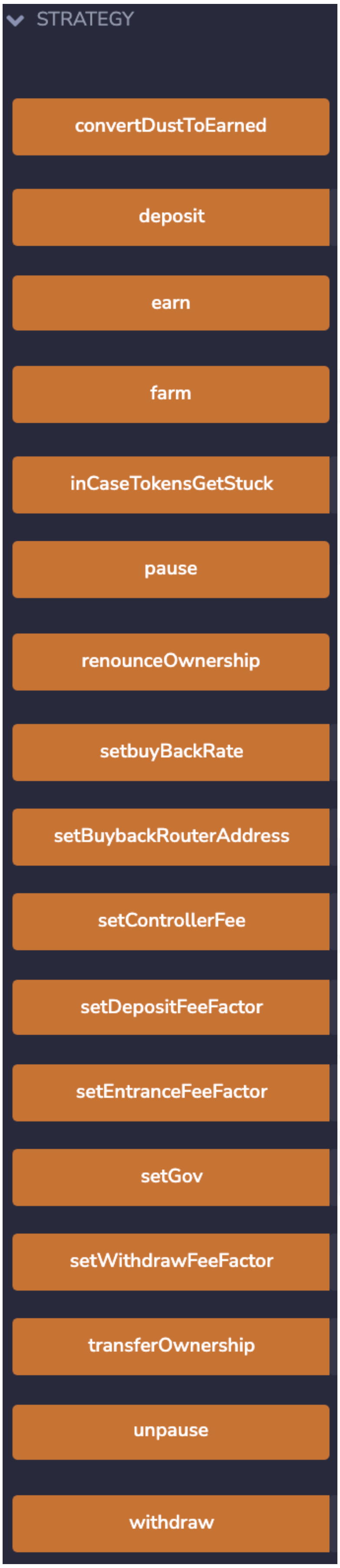


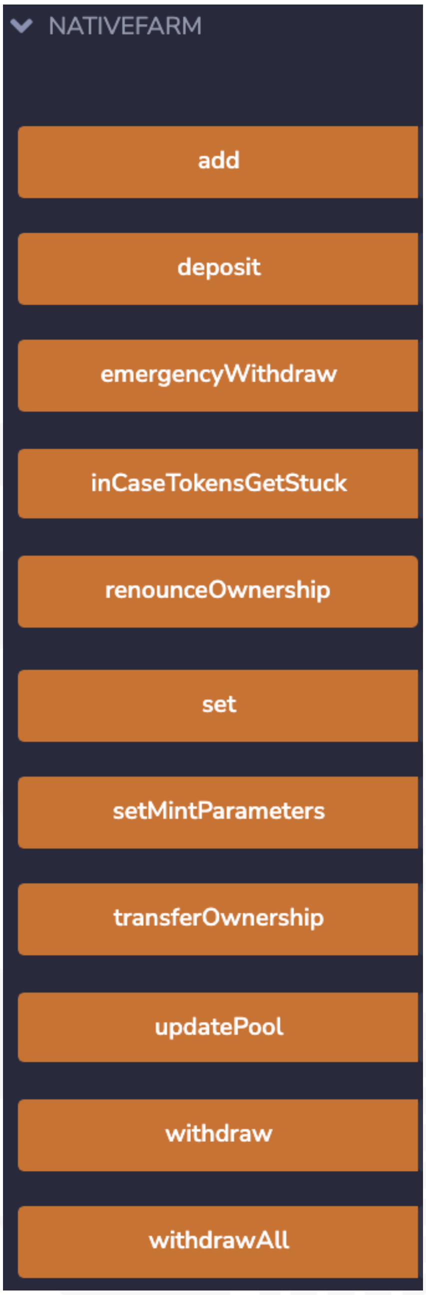
Verify Claims

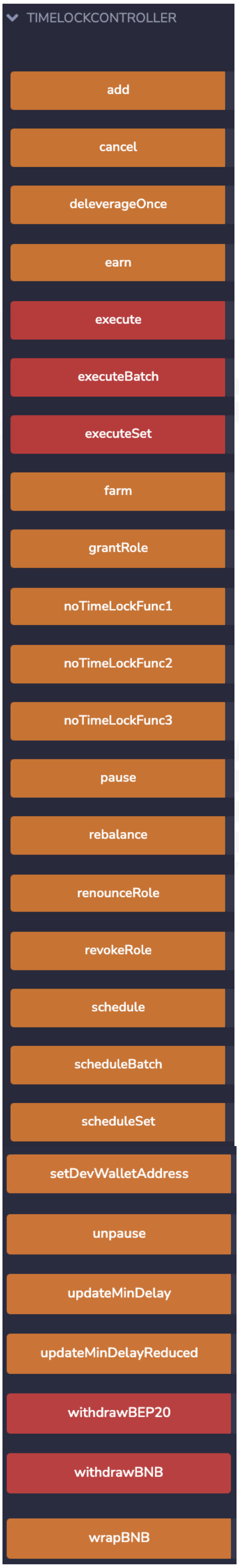
Correct implementation of Token standard

Tested	Verified
✓	✓

Write functions of contract







Overall checkup (Smart Contract Security)

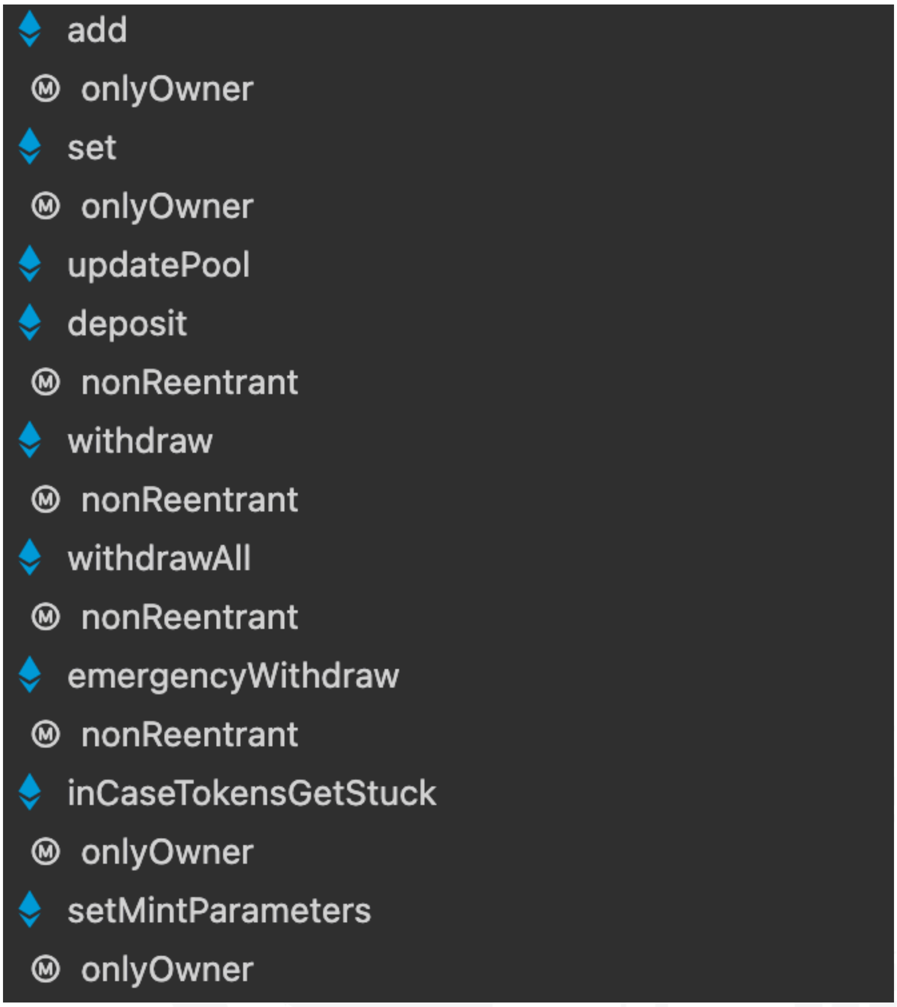
Tested	Verified
✓	✓

Legend

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

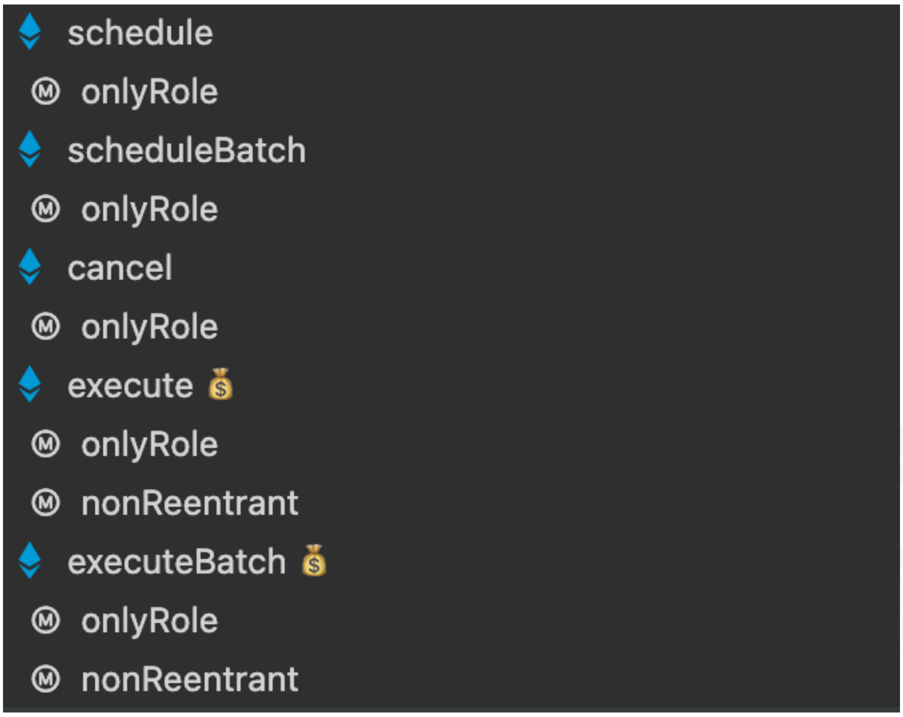
Modifiers

NativeFarm

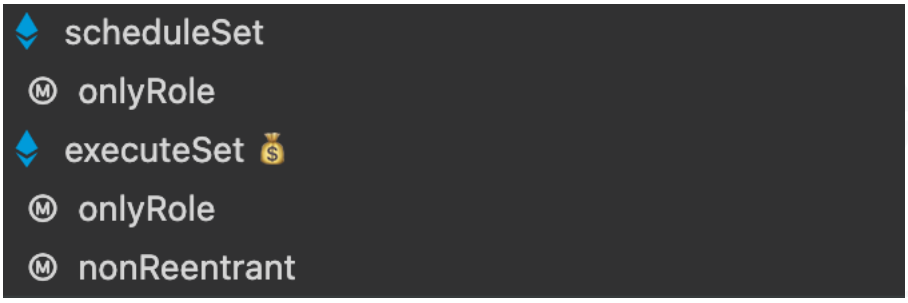


- add
  - Ⓜ onlyOwner
- set
  - Ⓜ onlyOwner
- updatePool
- deposit
  - Ⓜ nonReentrant
- withdraw
  - Ⓜ nonReentrant
- withdrawAll
  - Ⓜ nonReentrant
- emergencyWithdraw
  - Ⓜ nonReentrant
- inCaseTokensGetStuck
  - Ⓜ onlyOwner
- setMintParameters
  - Ⓜ onlyOwner

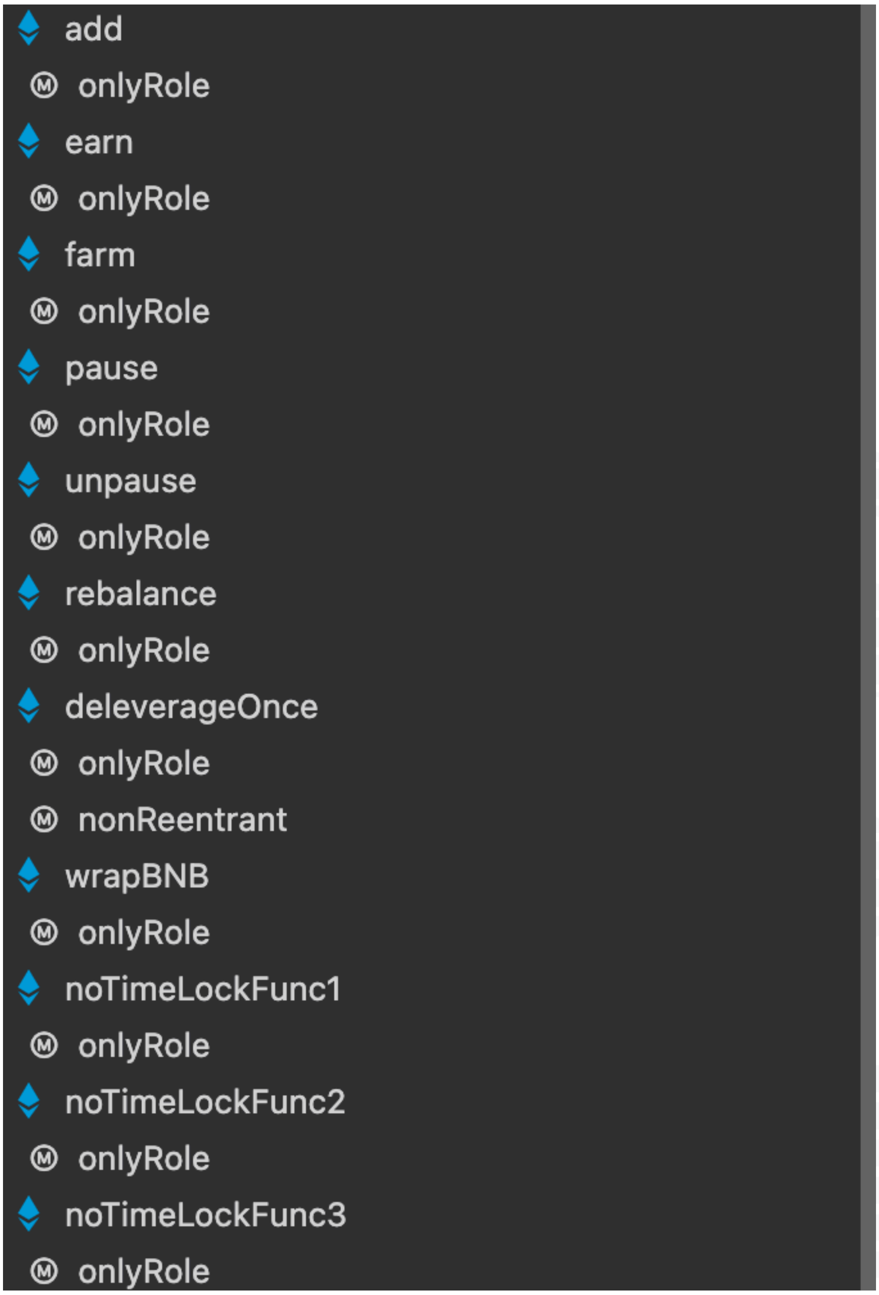
TimelockController



```
◆ schedule
  Ⓜ onlyRole
◆ scheduleBatch
  Ⓜ onlyRole
◆ cancel
  Ⓜ onlyRole
◆ execute 💰
  Ⓜ onlyRole
  Ⓜ nonReentrant
◆ executeBatch 💰
  Ⓜ onlyRole
  Ⓜ nonReentrant
```

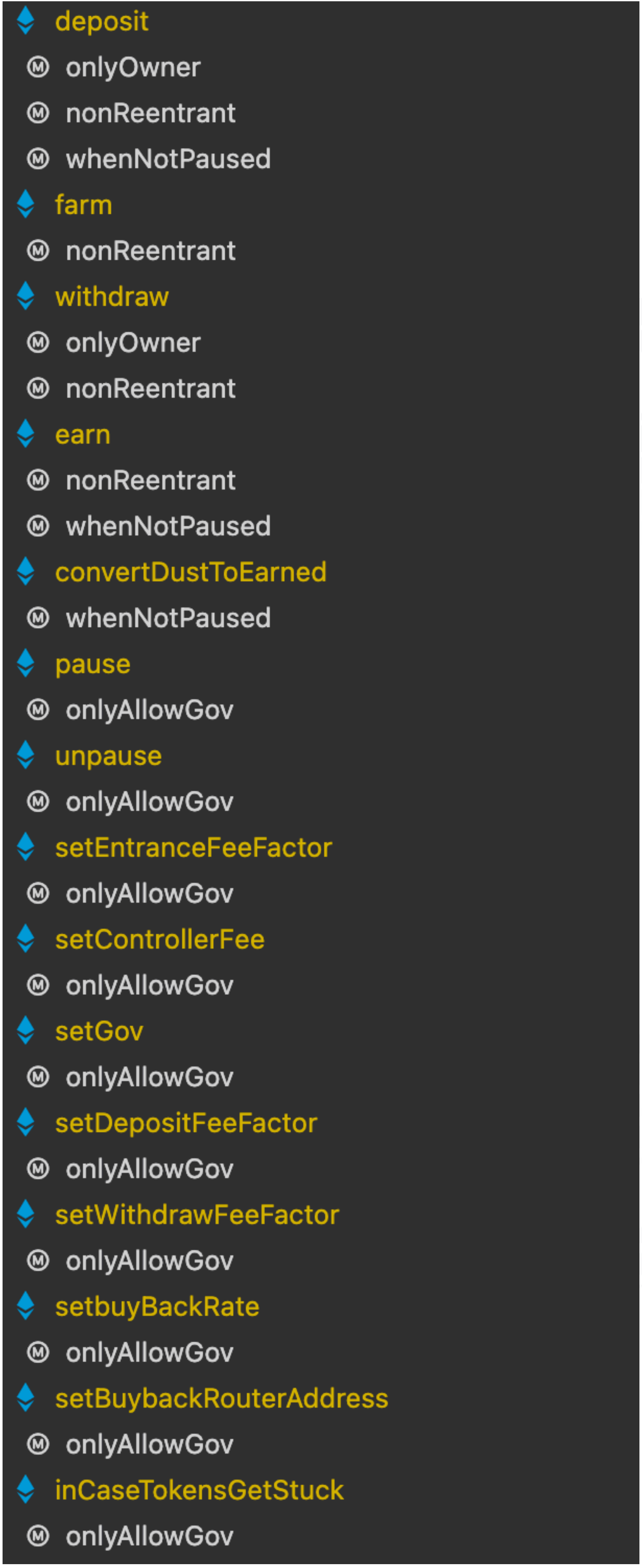


```
◆ scheduleSet
  Ⓜ onlyRole
◆ executeSet 💰
  Ⓜ onlyRole
  Ⓜ nonReentrant
```



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

Strategy



- deposit
  - Ⓜ onlyOwner
  - Ⓜ nonReentrant
  - Ⓜ whenNotPaused
- farm
  - Ⓜ nonReentrant
- withdraw
  - Ⓜ onlyOwner
  - Ⓜ nonReentrant
- earn
  - Ⓜ nonReentrant
  - Ⓜ whenNotPaused
- convertDustToEarned
  - Ⓜ whenNotPaused
- pause
  - Ⓜ onlyAllowGov
- unpause
  - Ⓜ onlyAllowGov
- setEntranceFeeFactor
  - Ⓜ onlyAllowGov
- setControllerFee
  - Ⓜ onlyAllowGov
- setGov
  - Ⓜ onlyAllowGov
- setDepositFeeFactor
  - Ⓜ onlyAllowGov
- setWithdrawFeeFactor
  - Ⓜ onlyAllowGov
- setbuyBackRate
  - Ⓜ onlyAllowGov
- setBuybackRouterAddress
  - Ⓜ onlyAllowGov
- inCaseTokensGetStuck
  - Ⓜ onlyAllowGov

CallGraph

Source Units in Scope

v1.0

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
🔍	contracts/interfaces/IXRouter02.sol	—	1	50	6	4	—	16	💰
🔍	contracts/interfaces/IXRouter01.sol	—	1	160	4	3	—	48	💰
🔍	contracts/interfaces/IPancakePair.sol	—	1	52	7	5	—	55	—
🔍	contracts/interfaces/IXswapFarm.sol	—	1	28	4	3	9	13	—
🔍	contracts/interfaces/IERC20.sol	—	1	86	26	21	54	13	🔆
🎨	contracts/helpers/AccessControl.sol	1	—	206	202	79	101	43	🔆
🎨	contracts/helpers/Context.sol	1	—	13	13	10	2	1	🔆
🎨	contracts/helpers/Ownable.sol	1	—	62	62	33	21	23	—
📝	contracts/helpers/Pausable.sol	1	—	80	80	29	41	14	🔆
🎨	contracts/helpers/ReentrancyGuard.sol	1	—	45	45	15	22	5	—
📝🔍	contracts/TimelockController.sol	1	2	621	500	290	173	267	🧪💰📤🧮
📝📚🔍🎨	contracts/NativeFarm.sol	10	2	1710	1407	685	674	426	🖥📤👥🔆
📚	contracts/libraries/EnumerableSet.sol	1	—	321	274	98	141	34	🔆
📚	contracts/libraries/Address.sol	1	—	262	190	96	116	49	🖥👥
📚	contracts/libraries/SafeMath.sol	1	—	157	145	39	93	10	🔆
📚	contracts/libraries/SafeERC20.sol	1	—	131	106	66	31	25	🔆
🔍🎨	contracts/Strategy.sol	1	1	486	457	355	38	269	💰
📝📚🔍🎨	**Totals**	**21**	**10**	**4470**	**3528**	**1831**	**1516**	**1311**	🖥🧪💰📤👥🧮🔆

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

Issue	File	Type	Line	Description
#1	NativeFarm	Unchecked tokens transfer	1693	Use `SafeERC20`, or ensure that the transfer/ transferFrom return value is checked

Low issues

Issue	File	Type	Line	Description
#1	All	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	NativeFarm	Constant cannot be set	1708	Constant variable cannot be set in a function NATIVEPerBlock
#3	Strategy	Contract cannot be compiled	22	Remove abstract from contract if you want to compile contract

Informational issues

Issue	File	Type	Line	Description
#1	All	SPDX License not provided	-	Provide SPDX License to source code
#2	Strategy	Remove unused parameter	88, 143	Variable was not used

Commented Code exist

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

File	Line	Comment
NativeFarm	129	// assert(a == b * c + a % b); // There is no case in which this doesn't hold

TimelockController	163	// for (uint256 i = 0; i < proposers.length; ++i) {
		// _setupRole(PROPOSER_ROLE, proposers[i]);
		// }

TimelockController	169	// for (uint256 i = 0; i < executors.length; ++i) {
		// _setupRole(EXECUTOR_ROLE, executors[i]);
		// }

IXSwapFarm	15	// function pendingCake(uint256 _pid, address _user) // external // view // returns (uint256);
SafeMath	116	// assert(a == b * c + a % b); // There is no case in which this doesn't hold

Recommendation

Remove the commented code, or address them properly.

Audit Comments

11. January 2022:

- Deployer can pause following contracts
  - Strategy
- Naming Convention
  - Constant variables should be uppercased
- Read whole report 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	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	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	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