SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

Unicorn Hunter

Audit

Security Assessment

25. August, 2022

For

Unicorn

HUNTER

SolidProof_io

@solidproof_io

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

Network

Binance Smart Chain (BEP20)

Website

https://unicornhunter.io/

Telegram

https://t.me/+wiAE-USt9BU0N2Jl

Twitter

https://twitter.com/Unicorn8668

Facebook

https://www.facebook.com/UnicornHunterCapital

Youtube

https://www.youtube.com/channel/UClGKpp_cnRqoSB2y0Lqov4w

Description

Unicorn Hunter is an Asia-based investment firm that was established by a group of experts who have experiences in cryptocurrency & blockchain industry since 2014 and high-return investments since 2006.

Project Engagement

During the 23rd of August 2022, **Unicorn Hunter 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

- Github
    - https://github.com/Unicorn-Hunter-Venture-Capital/contracts
    - Commit: b3d8bf7625c30be913e1f03e65f11b7497f8fc2b

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.

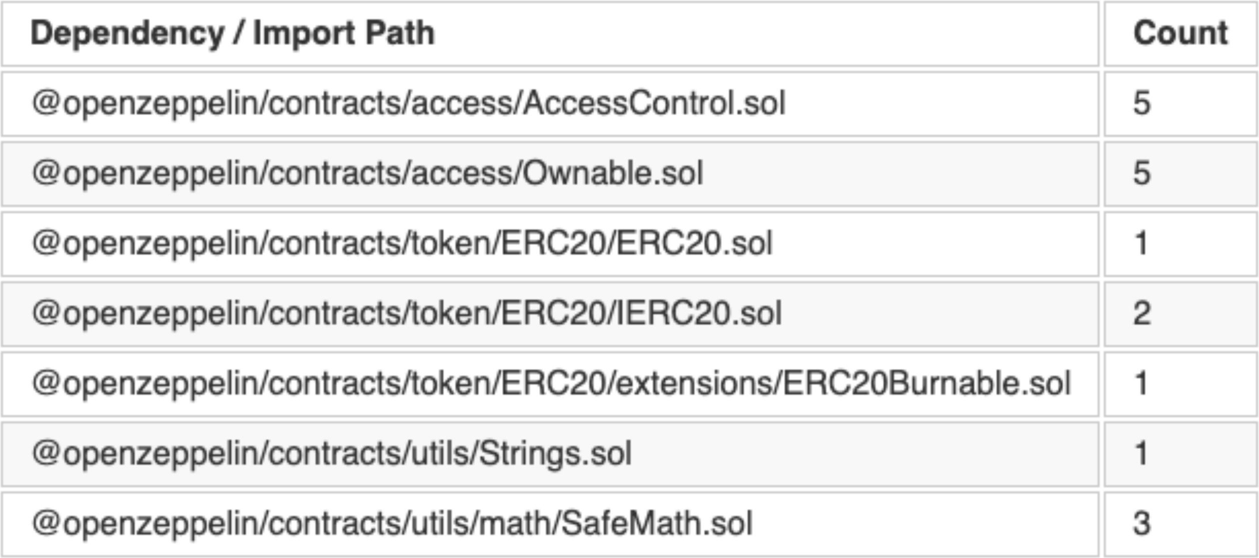
2. Testing and automated analysis that includes the following:
    i) Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
    ii) Symbolic execution, which is analysing a program 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:



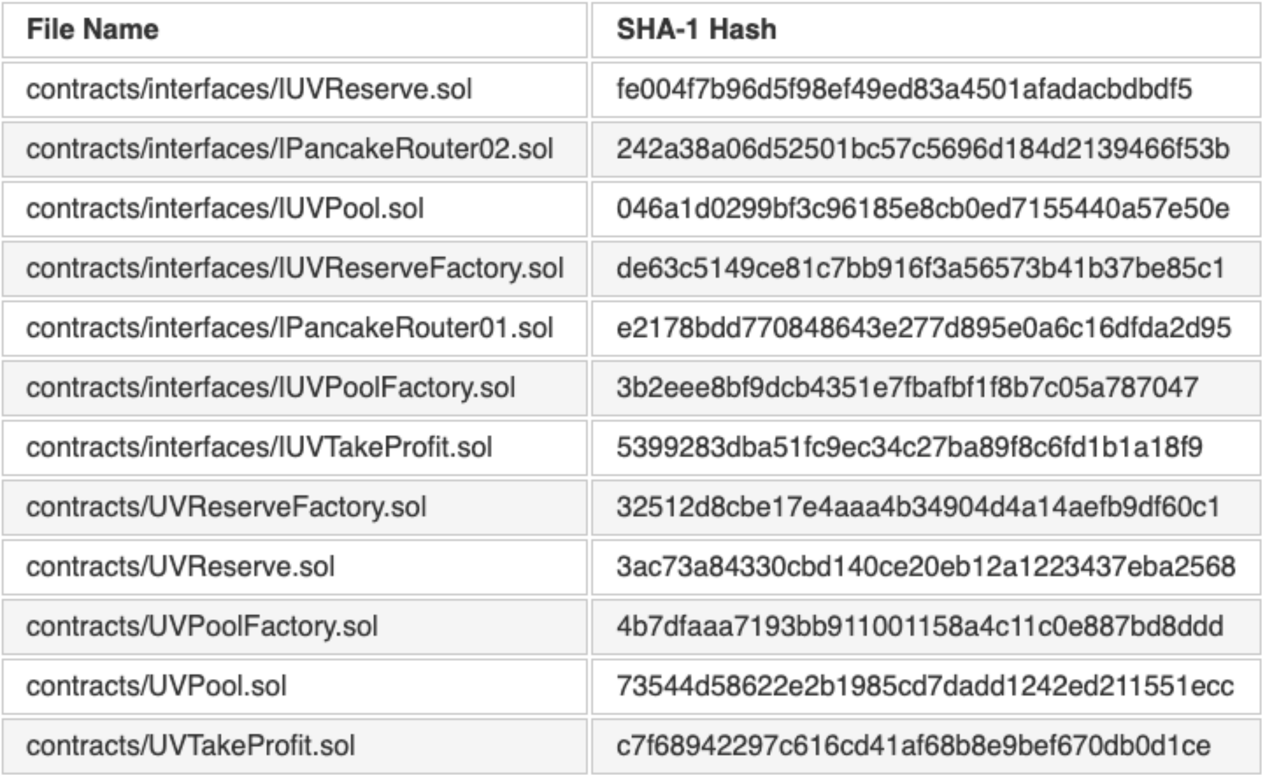
Dependency / Import Path	Count
@openzeppelin/contracts/access/AccessControl.sol	5
@openzeppelin/contracts/access/Ownable.sol	5
@openzeppelin/contracts/token/ERC20/ERC20.sol	1
@openzeppelin/contracts/token/ERC20/IERC20.sol	2
@openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.sol	1
@openzeppelin/contracts/utils/Strings.sol	1
@openzeppelin/contracts/utils/math/SafeMath.sol	3

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

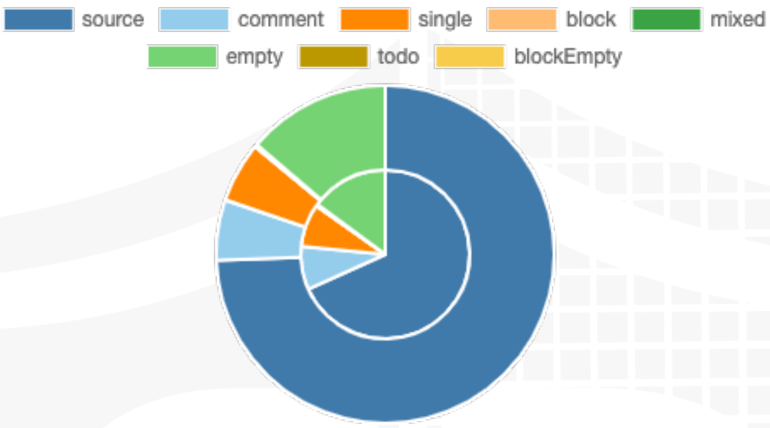


File Name	SHA-1 Hash
contracts/interfaces/IUVReserve.sol	fe004f7b96d5f98ef49ed83a4501afadacbdbdf5
contracts/interfaces/IPancakeRouter02.sol	242a38a06d52501bc57c5696d184d2139466f53b
contracts/interfaces/IUVPool.sol	046a1d0299bf3c96185e8cb0ed7155440a57e50e
contracts/interfaces/IUVReserveFactory.sol	de63c5149ce81c7bb916f3a56573b41b37be85c1
contracts/interfaces/IPancakeRouter01.sol	e2178bdd770848643e277d895e0a6c16dfda2d95
contracts/interfaces/IUVPoolFactory.sol	3b2eee8bf9dcb4351e7fbafbf1f8b7c05a787047
contracts/interfaces/IUVTakeProfit.sol	5399283dba51fc9ec34c27ba89f8c6fd1b1a18f9
contracts/UVReserveFactory.sol	32512d8cbe17e4aaa4b34904d4a14aefb9df60c1
contracts/UVReserve.sol	3ac73a84330cbd140ce20eb12a1223437eba2568
contracts/UVPoolFactory.sol	4b7dfaaa7193bb911001158a4c11c0e887bd8ddd
contracts/UVPool.sol	73544d58622e2b1985cd7dadd1242ed211551ecc
contracts/UVTakeProfit.sol	c7f68942297c616cd41af68b8e9bef670db0d1ce

Metrics

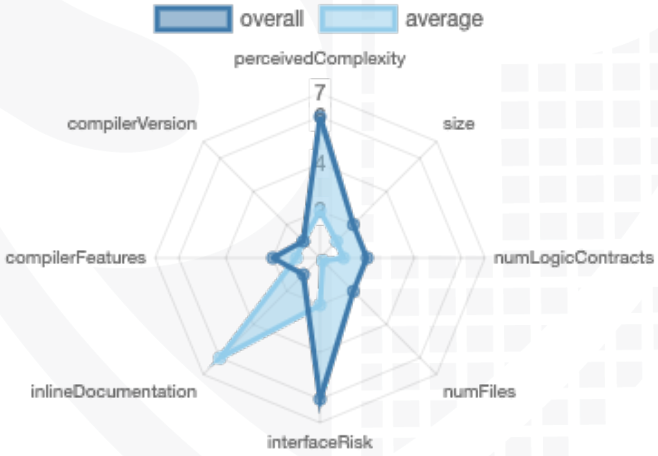
Source Lines

v1.0



Risk Level

v1.0



Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	5	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	139	12

Version	External	Internal	Private	Pure	View
1.0	85	112	0	5	17

State Variables

Version	Total	Public
1.0	55	51

Capabilities

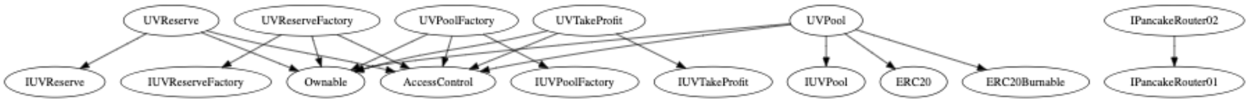
Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts

1.0	`^0.8.12`
	`>=0.6.2`
			`yes`		

Version	Transfers ETH	Low-Level Calls	DelegateCall	Uses Hash Functions	ECRecover	New/Create/Create2
1.0	yes			yes		yes → NewContract:UVTakeProfit → NewContract:UVReserve

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).

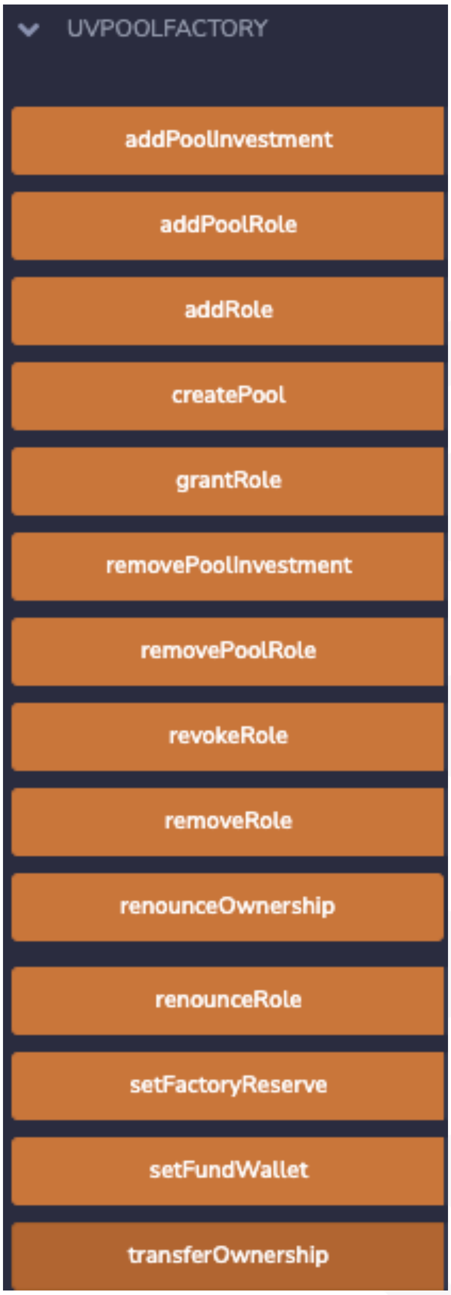
We will verify the following claims:

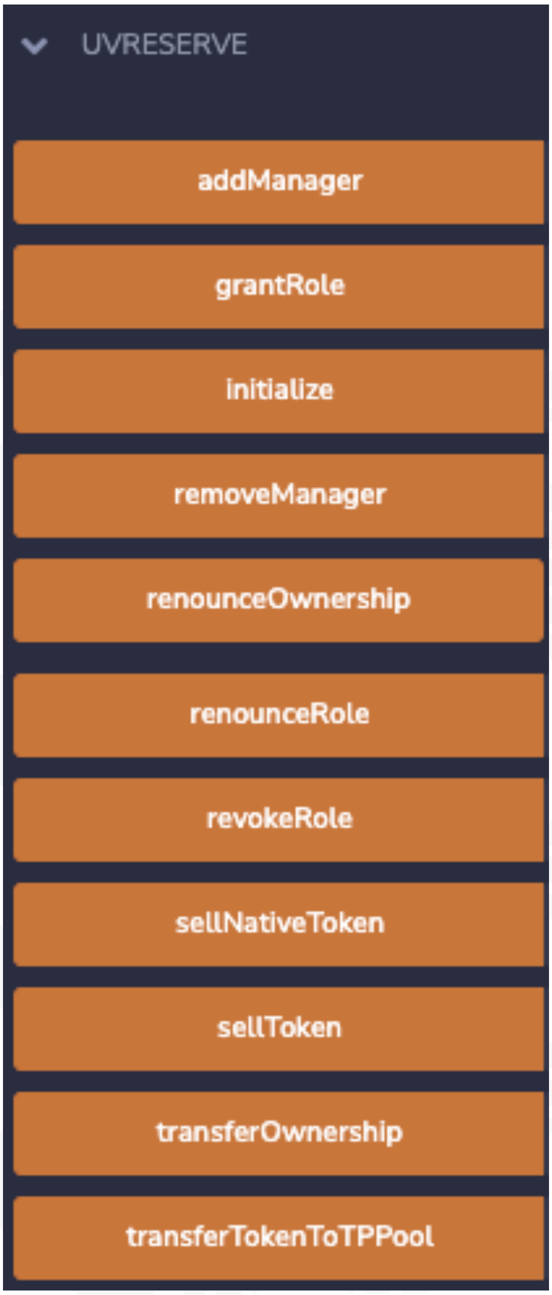
1. Is contract an upgradeable
2. Deployer cannot mint any new tokens
3. Deployer cannot burn or lock user funds
4. Deployer cannot pause the contract
5. Deployer cannot set fees
6. Deployer cannot blacklist/antisnipe addresses
7. Overall checkup (Smart Contract Security)

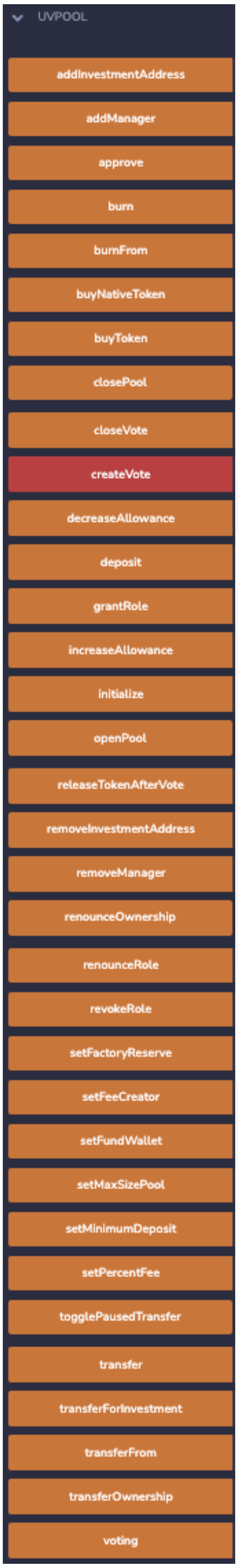
Is contract an upgradeable

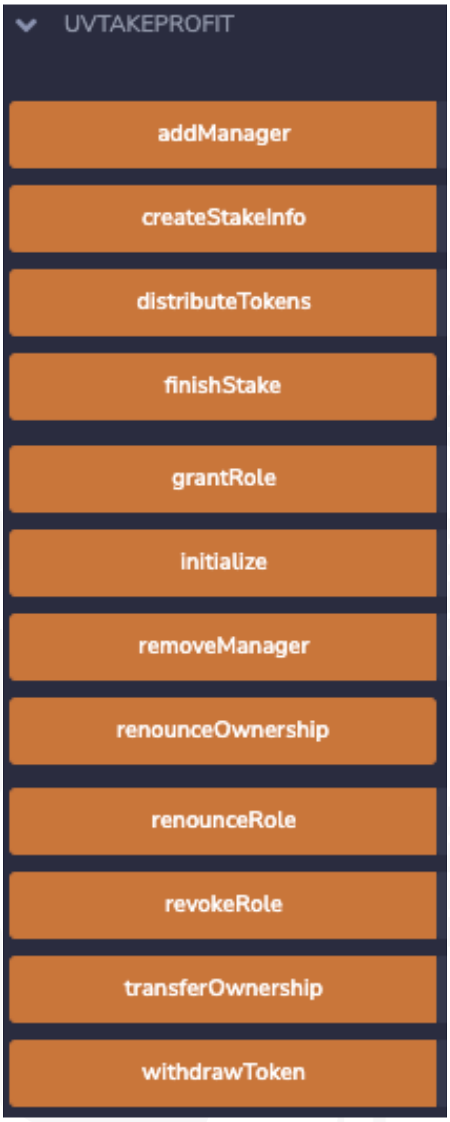
Name	
Is contract an upgradeable?	No

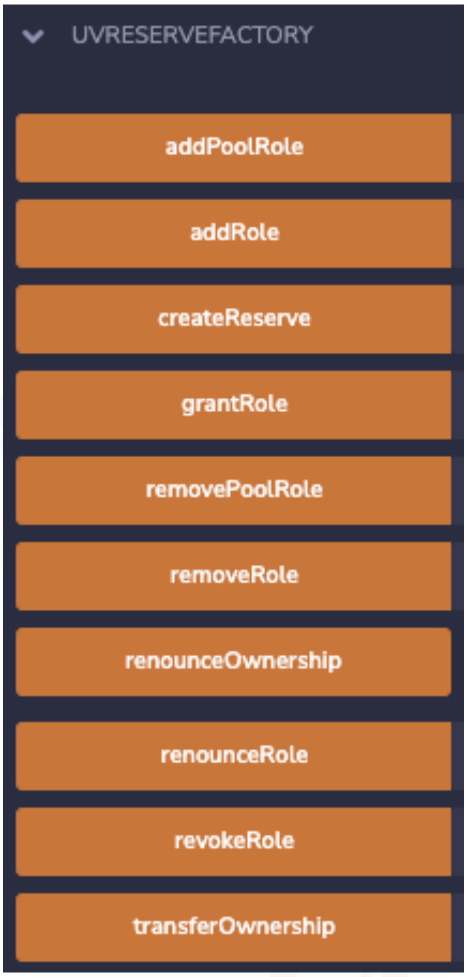
Write functions of contract v1.0






UVPOOL
addInvestmentAddress
addManager
approve
burn
burnFrom
buyNativeToken
buyToken
closePool
closeVote
createVote
decreaseAllowance
deposit
grantRole
increaseAllowance
initialize
openPool
releaseTokenAfterVote
removeInvestmentAddress
removeManager
renounceOwnership
renounceRole
revokeRole
setFactoryReserve
setFeeCreator
setFundWallet
setMaxSizePool
setMinimumDeposit
setPercentFee
togglePausedTransfer
transfer
transferForInvestment
transferFrom
transferOwnership
voting


UVTAKEPROFIT
addManager
createStakeInfo
distributeTokens
finishStake
grantRole
initialize
removeManager
renounceOwnership
renounceRole
revokeRole
transferOwnership
withdrawToken


UVRESERVEFACTORY
addPoolRole
addRole
createReserve
grantRole
removePoolRole
removeRole
renounceOwnership
renounceRole
revokeRole
transferOwnership

Deployer cannot mint any new tokens

Name	Exist	Tested	Status
Deployer cannot mint	✓	✓	✓

Comments:

v1.0

- Tokens will be minted while depositing in UVPool

Deployer cannot burn or lock user funds

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

Comments:

v1.0

- Owner can lock user funds by
  - Pausing in UVPool
  - Setting maxSizePool to 0 in UVPool

- Tokens
  - can be burned by msg.sender in UVPool

Deployer cannot pause the contract

Name	Exist	Tested	Status
Deployer cannot pause	✓	✓	✗

Comments:

v1.0

- Manager can pause contract in UVPool

Deployer cannot set fees

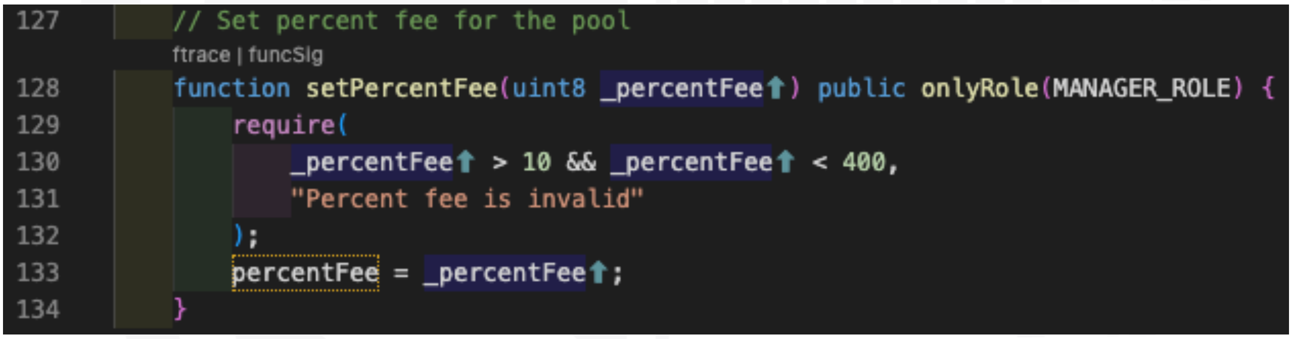
Name	Exist	Tested	Status
Deployer cannot set fees over 25%	✓	✓	✗
Deployer cannot set fees to nearly 100% or to 100%	✓	✓	✓

Comments:

v1.0

- Fees can be set up to 40%

UVPool



```
// Set percent fee for the pool
function setPercentFee(uint8 _percentFee) public onlyRole(MANAGER_ROLE) {
    require(
        _percentFee > 10 && _percentFee < 400,
        "Percent fee is invalid"
    );
    percentFee = _percentFee;
}
```

Deployer can blacklist/antisnipe addresses

Name	Exist	Tested	Status
Deployer cannot blacklist/antisnipe addresses	–	–	–

Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

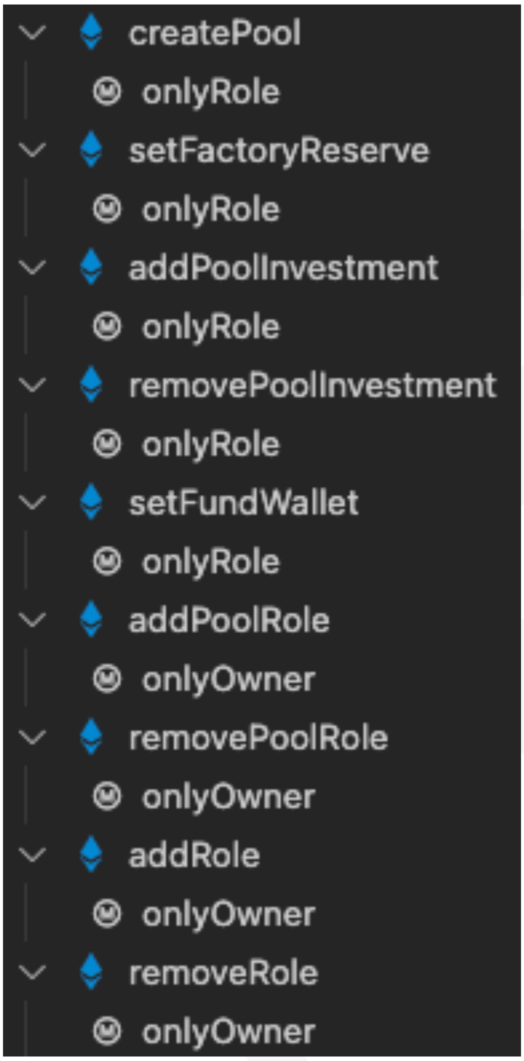
Legend

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

Modifiers and public functions

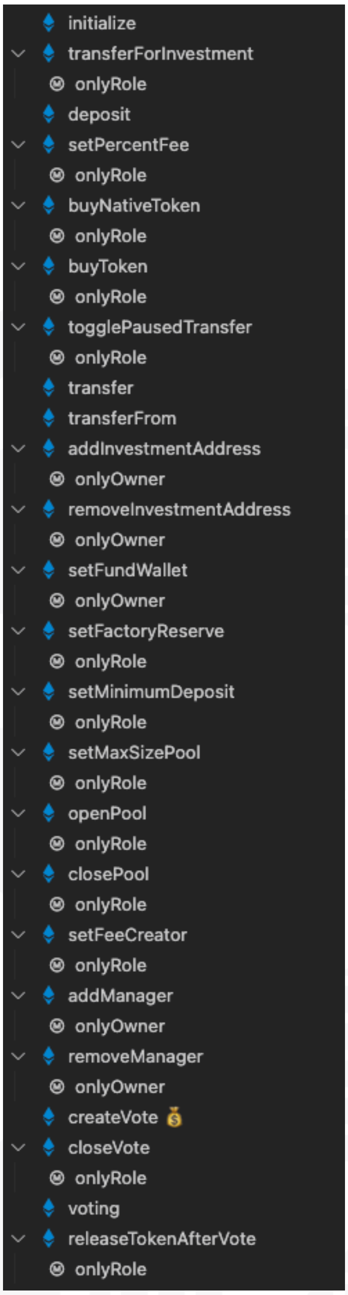
v1.0

UVPoolFactory



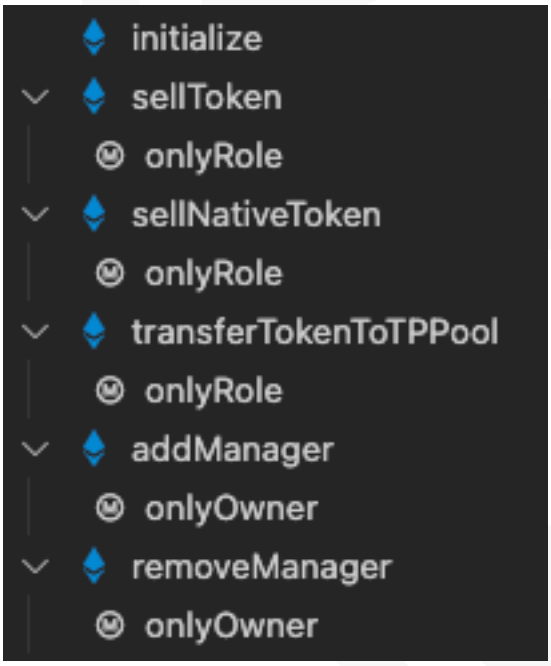
- createPool
  - onlyRole
- setFactoryReserve
  - onlyRole
- addPoolInvestment
  - onlyRole
- removePoolInvestment
  - onlyRole
- setFundWallet
  - onlyRole
- addPoolRole
  - onlyOwner
- removePoolRole
  - onlyOwner
- addRole
  - onlyOwner
- removeRole
  - onlyOwner

UVPool



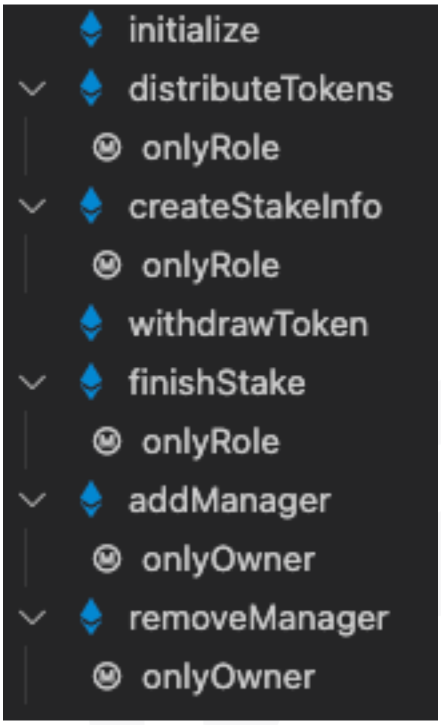
```
  initialize
⌄ transferForInvestment
    onlyRole
  deposit
⌄ setPercentFee
    onlyRole
⌄ buyNativeToken
    onlyRole
⌄ buyToken
    onlyRole
⌄ togglePausedTransfer
    onlyRole
  transfer
  transferFrom
⌄ addInvestmentAddress
    onlyOwner
⌄ removeInvestmentAddress
    onlyOwner
⌄ setFundWallet
    onlyOwner
⌄ setFactoryReserve
    onlyRole
⌄ setMinimumDeposit
    onlyRole
⌄ setMaxSizePool
    onlyRole
⌄ openPool
    onlyRole
⌄ closePool
    onlyRole
⌄ setFeeCreator
    onlyRole
⌄ addManager
    onlyOwner
⌄ removeManager
    onlyOwner
  createVote 💰
⌄ closeVote
    onlyRole
  voting
⌄ releaseTokenAfterVote
    onlyRole
```

UVReserve



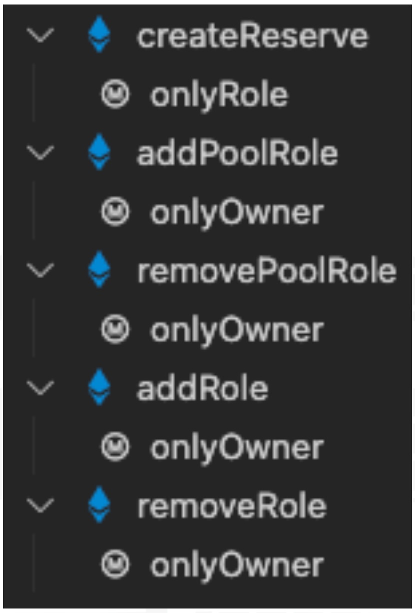
```
  initialize
⌄ sellToken
    onlyRole
⌄ sellNativeToken
    onlyRole
⌄ transferTokenToTPPool
    onlyRole
⌄ addManager
    onlyOwner
⌄ removeManager
    onlyOwner
```

UVTakeProfit



```
  initialize
˅ distributeTokens
    onlyRole
˅ createStakeInfo
    onlyRole
  withdrawToken
˅ finishStake
    onlyRole
˅ addManager
    onlyOwner
˅ removeManager
    onlyOwner
```

UVReserveFactory



```
˅ createReserve
    onlyRole
˅ addPoolRole
    onlyOwner
˅ removePoolRole
    onlyOwner
˅ addRole
    onlyOwner
˅ removeRole
    onlyOwner
```

Note: Not listed functions/modifiers was implemented from libraries

Comments

- *Deployer can set following state variables without any limitations*
  - UVPool
    - feeCreator
    - maxSizePool
    - minimumDeposit
  - UVTakeProfit
    - stakeInfos[_id].totalToken
    - stakeInfos[_id].remainingToken
    - stakeInfos[_id].closeTimestamp
    - stakeInfos[_id].openTimestamp
    - stakeInfos[_id].feePercent

- *Deployer can enable/disable following state variables*
  - UVPool
    - voteCreateable
    - allVotes[_orderNumber].isActive
    - isClose
    - investmentAddreses
    - pausedTransfer
  - UVPoolFactory
    - managers[_orderNumber]

- *Deployer can set following addresses*
    - UVPool
        - factoryReserve
        - fundWallet
    - UVPoolFactory
        - factoryReserve

    - UVTakeProfit
        - stakeInfos[_id].tokenAddress

Comments

- UVPool
    - Initialize function can be called more than once
    - Owner can revoke/add manager
    - Only manager can call all functions except
        - addInvestmentAddress
        - removeInvestmentAddress
        - setFundWallet
        - addManager
        - removeManager
- UVPoolFactory
    - Admin can
        - create pools
        - Add/remove investment
        - Set fund wallet
        - Add admin role to address
- UVReserve
    - Manager can
        - Sell tokens
        - Sell native tokens
        - Transfer tokens to staking pool
    - Admin can
        - Grant manager role
- UVReserveFactory
    - Admin can
        - Create new reserve
    - Owner can
        - Add/remove role
- UVTakeProfit
    - Manager can
        - Distribute tokens
        - Create stake info

    - Finish pool with “finishStake” function. This will send every poolToken balance to burn address
    - Add/remove manager role
- Unnecessary “require” statement in “withdrawToken” L145. If the “_amountStake” is above 0 the function will be continued. “_amountStake” will be set then to “users[_stakeId][msg.sender].amount” which is also not 0.

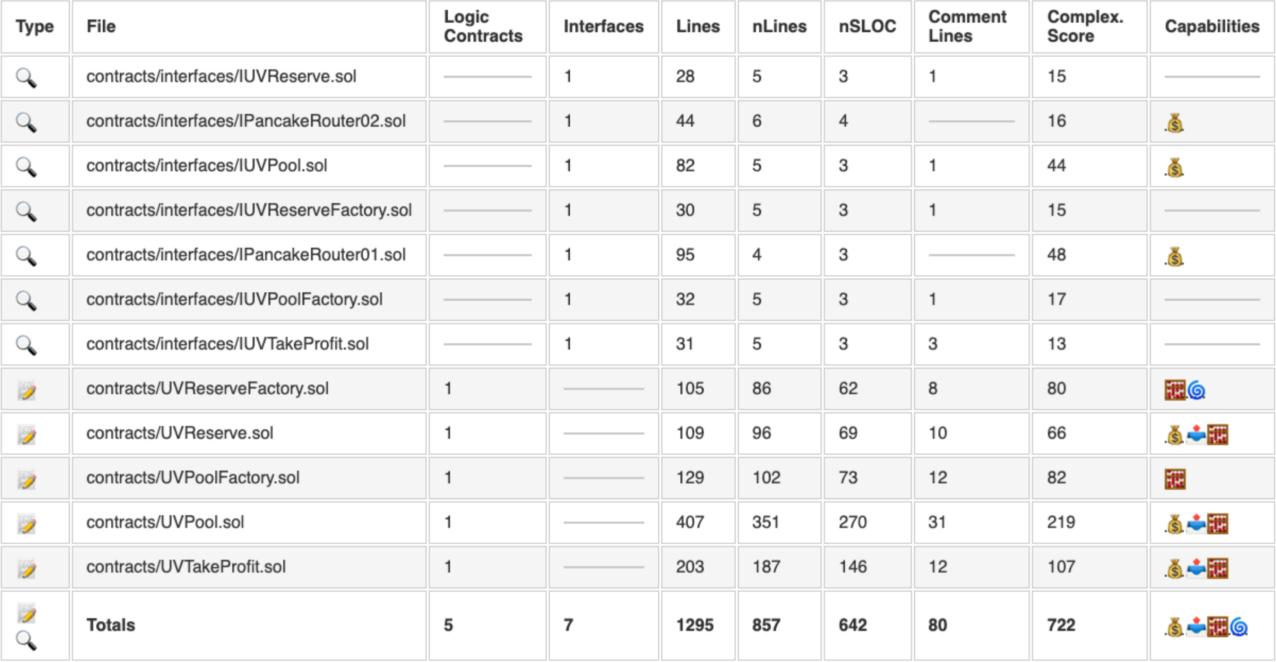
Please check every type of uint max type and the variable whether it is possible to set to it or not:

- uint8 => max: 2^8-1 = 255
- Uint16 => max: 2^16-1 = 65.535
- Uint64 => max: 2^64-1 = 1,84467441e19
- And so on

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/IUVReserve.sol	—	1	28	5	3	1	15	—
🔍	contracts/interfaces/IPancakeRouter02.sol	—	1	44	6	4	—	16	💰
🔍	contracts/interfaces/IUVPool.sol	—	1	82	5	3	1	44	💰
🔍	contracts/interfaces/IUVReserveFactory.sol	—	1	30	5	3	1	15	—
🔍	contracts/interfaces/IPancakeRouter01.sol	—	1	95	4	3	—	48	💰
🔍	contracts/interfaces/IUVPoolFactory.sol	—	1	32	5	3	1	17	—
🔍	contracts/interfaces/IUVTakeProfit.sol	—	1	31	5	3	3	13	—
📝	contracts/UVReserveFactory.sol	1	—	105	86	62	8	80	🧮🌀
📝	contracts/UVReserve.sol	1	—	109	96	69	10	66	💰📤🧮
📝	contracts/UVPoolFactory.sol	1	—	129	102	73	12	82	🧮
📝	contracts/UVPool.sol	1	—	407	351	270	31	219	💰📤🧮
📝	contracts/UVTakeProfit.sol	1	—	203	187	146	12	107	💰📤🧮
📝 🔍	**Totals**	**5**	**7**	**1295**	**857**	**642**	**80**	**722**	💰📤🧮🌀

Legend

Attribute	Description
Lines	total lines of the source unit
nLines	normalised lines of the source unit (e.g. normalises functions spanning multiple lines)
nSLOC	normalised 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	UVPool	400 is too large for type	128	The max value of type of uint8 is 2^8-1 = 255. You cannot set it higher than 255.

Low issues

Issue	File	Type	Line	Description
#1	All	A floating pragma is set	See description	See pragma versions, especially the one that started with "^" sign
#2	UVReserve	Missing Zero Address Validation (missing-zero-check)	38	Check that the address is not zero
#3	UVTake Profit	Missing Zero Address Validation (missing-zero-check)	63, 64	Check that the address is not zero
#4	UVPool	Missing Zero Address Validation (missing-zero-check)	75, 76, 93	Check that the address is not zero
#5	UVPool Factory	Missing Zero Address Validation (missing-zero-check)	103, 109	Check that the address is not zero

#6	UVReserveFactory	Missing Zero Address Validation (missing-zero-check)	31-33, 52, 62, 73, 78	Check that the address is not zero
#7	UVPool	State variable visibility is not set	55	It is best practice to set the visibility of state variables explicitly
#8	UVPoolFactory	State variable visibility is not set	15	It is best practice to set the visibility of state variables explicitly
#9	UVReserveFactory	State variable visibility is not set	20	It is best practice to set the visibility of state variables explicitly
#10	UVTakeProfit	Missing Events Arithmetic	62	Emit an event for critical parameter changes
#11	UVPool	Missing Events Arithmetic	189, 249, 257, 265, 273, 278, 283, 293	Emit an event for critical parameter changes
#12	UVPool	Wrong error message or statement	353	Check the isClosed require statement in L353. Change error message or statement

Informational issues

Issue	File	Type	Line	Description
#1	UVReserve	State variables that could be declared constant (constable-states)	24	Add the `constant` attributes to state variables that never change
#2	UVTakeProfit	State variables that could be declared constant (constable-states)	24	Add the `constant` attributes to state variables that never change
#3	UVReserve	Unused state variables	20, 24	Remove unused state variables
#4	UVTakeProfit	Unused state variables	41	Remove unused state variables
#5	UVReserve	Error message is missing	89, 90	Provide an error message for require statement
#6	UVTakeProfit	Error message is missing	121, 126	Provide an error message for require statement

#7	All	NatSpec documentation missing	-	If you started to comment your code, also comment all other functions, variables etc.
#8	UVPool	Unnecessary check	See description	The Manager_Role was checked in the “if condition” in L323 but you are checking the role again in the “else” condition in L331. If the “else” should be called, the msg.sender doesn’t have the role
#9	UVPool	Wrong error message	398	Replace “This vote is closed” with “This Vote is still active”
#10	UVRese rve	Mainnet pancake address	17, 20	Don’t forget to change the testate router/busd address while you are deploying
#11	UVTake Profit	Mainnet pancake address	38, 41	Don’t forget to change the testate router/busd address while you are deploying
#12	IUVPool	Wrong named parameters	See description	Interface started in the following order; • uint256 _amountOut, • uint256 _amountInMax, • address[] calldata _path, • uint256 _deadline But the function itself has the following order: • uint256 _amountIn, • uint256 _amountOutMin, • address[] calldata _path, • uint256 _deadline
#13	IUVRese rveFact ory	Wrong named parameters	See description	Interface return parameters started in the following order; • address _poolReserve, • address _poolStake, • address _manager, • address _fundWallet But the function itself has the following order: • address _poolReserve, • address _poolTP, • address _manager, • address _fundWallet

#14	IUVTake Profit	Wrong named parameters	See description	Interface started in the following order; • uint8 _stakeId, • address _tokenAddress, • uint256 _amount, • uint64 _openTime, • uint64 _closeTime, • uint8 _feePercent But the function itself has the following order: • uint8 _id, • address _tokenAddress, • uint256 _amount, • uint64 _openTime, • uint64 _closeTime, • uint8 _feePercent
#15	IPancak eRouter 01	SPDX License is missing	See description	Provide a SPDX License at the top of the file.
#16	IPancak eRouter 02	SPDX License is missing	See description	Provide a SPDX License at the top of the file.

Commented Code exist

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

File	Line	Comment
IUVTakeProfit	20	// function stakeToken(uint8 _stakeId) external;
	24	// function unstakeToken(uint8 _stakeId) external;

Recommendation

Remove the commented code, or address them properly.

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. August 2022:

- Please check max type of uint’s
- Whitepaper was not provided from customer
- 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	NOT 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