SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY

ZaynCore

Audit

Security Assessment
15. December, 2022

For

Zayn

Disclaimer

Version	Date	Description
1.0	23. November 2022	• Layout project • Automated- /Manual-Security Testing • Summary
1.1	15. December 2022	• Reaudit

Network
Binance Smart Chain (BEP20)

Website
https://zayn.fi/

Telegram
https://t.me/zaynfi

Twitter
https://twitter.com/ZaynFinance

Medium
https://medium.com/@zfadmin

Discord
https://discord.gg/zaynfi

Description
Decentralized Finance (DeFi) is the new frontier of money. A system that is transparent, fair and empowering. We believe that DeFi should be for all. For that to happen, it should be as simple as possible. That is where we come in.

Presenting, the easiest way to earn in DeFi.

Project Engagement
During the 22nd of November 2022, **ZaynCore 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/ZaynFi/zayn-core
  - Commit: b5e8f274efccc4026425f61bd985aad91849c65c

v1.1
- Github
  - https://github.com/ZaynFi/zayn-core
  - Commit: 54eac8cbc007103a2794dd4f0499bad149525950

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.

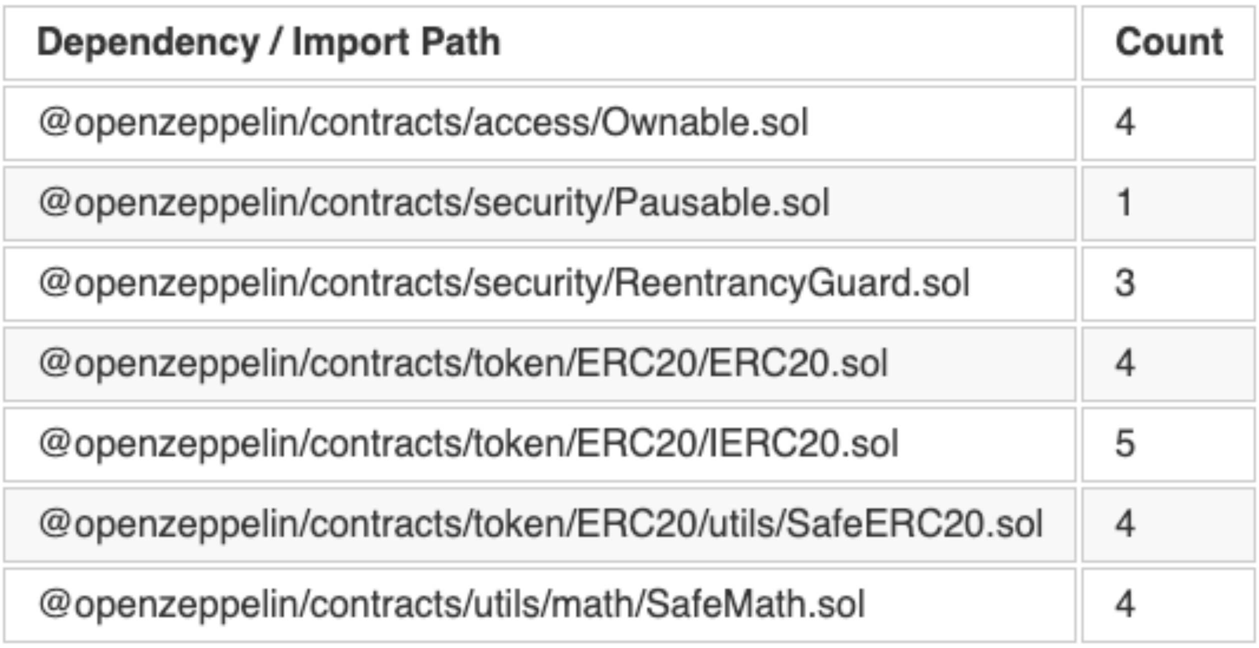
2. Testing and automated analysis that includes the following:
    i) Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
    ii) Symbolic execution, which is analysing a program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



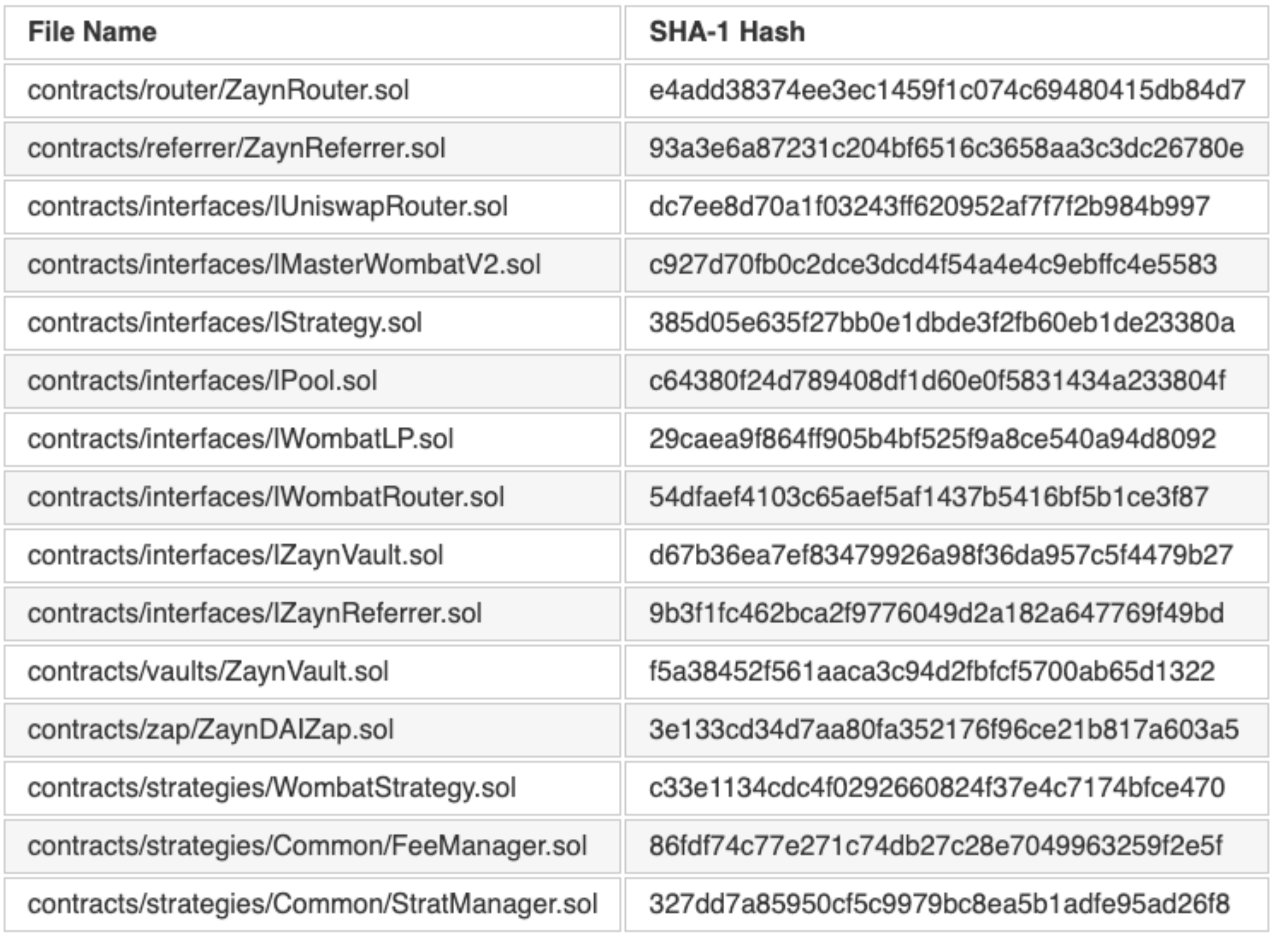
Dependency / Import Path	Count
@openzeppelin/contracts/access/Ownable.sol	4
@openzeppelin/contracts/security/Pausable.sol	1
@openzeppelin/contracts/security/ReentrancyGuard.sol	3
@openzeppelin/contracts/token/ERC20/ERC20.sol	4
@openzeppelin/contracts/token/ERC20/IERC20.sol	5
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol	4
@openzeppelin/contracts/utils/math/SafeMath.sol	4

Tested Contract Files

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

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

v1.0

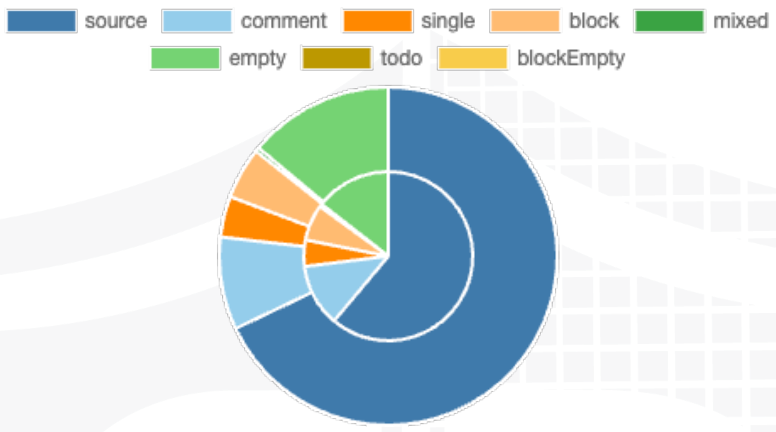


File Name	SHA-1 Hash
contracts/router/ZaynRouter.sol	e4add38374ee3ec1459f1c074c69480415db84d7
contracts/referrer/ZaynReferrer.sol	93a3e6a87231c204bf6516c3658aa3c3dc26780e
contracts/interfaces/IUniswapRouter.sol	dc7ee8d70a1f03243ff620952af7f7f2b984b997
contracts/interfaces/IMasterWombatV2.sol	c927d70fb0c2dce3dcd4f54a4e4c9ebffc4e5583
contracts/interfaces/IStrategy.sol	385d05e635f27bb0e1dbde3f2fb60eb1de23380a
contracts/interfaces/IPool.sol	c64380f24d789408df1d60e0f5831434a233804f
contracts/interfaces/IWombatLP.sol	29caea9f864ff905b4bf525f9a8ce540a94d8092
contracts/interfaces/IWombatRouter.sol	54dfaef4103c65aef5af1437b5416bf5b1ce3f87
contracts/interfaces/IZaynVault.sol	d67b36ea7ef83479926a98f36da957c5f4479b27
contracts/interfaces/IZaynReferrer.sol	9b3f1fc462bca2f9776049d2a182a647769f49bd
contracts/vaults/ZaynVault.sol	f5a38452f561aaca3c94d2fbfcf5700ab65d1322
contracts/zap/ZaynDAIZap.sol	3e133cd34d7aa80fa352176f96ce21b817a603a5
contracts/strategies/WombatStrategy.sol	c33e1134cdc4f0292660824f37e4c7174bfce470
contracts/strategies/Common/FeeManager.sol	86fdf74c77e271c74db27c28e7049963259f2e5f
contracts/strategies/Common/StratManager.sol	327dd7a85950cf5c9979bc8ea5b1adfe95ad26f8

Metrics

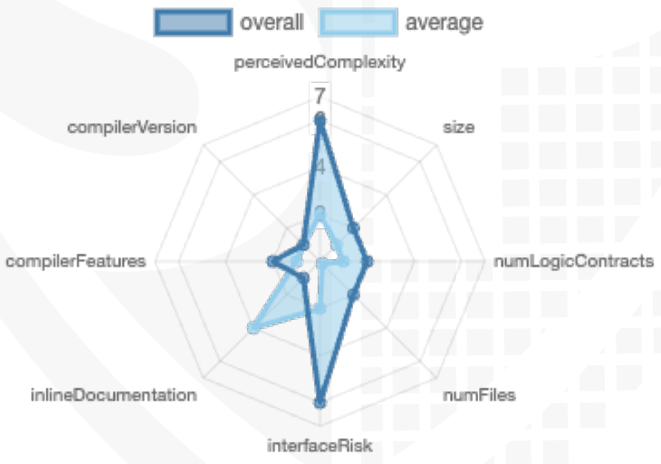
Source Lines

v1.0



Risk Level

v1.0



Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	6	3	14	1

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	232	12

Version	External	Internal	Private	Pure	View
1.0	193	173	1	24	68

State Variables

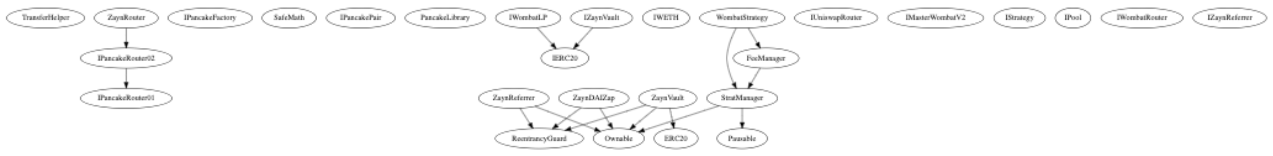
Version	Total	Public
1.0	47	46

Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	`>=0.6.0` `>=0.6.2` `>=0.5.0` `=0.6.6` `^0.8.0` `>=0.6.0` `<0.9.0` `^0.8.5`		`yes`		

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

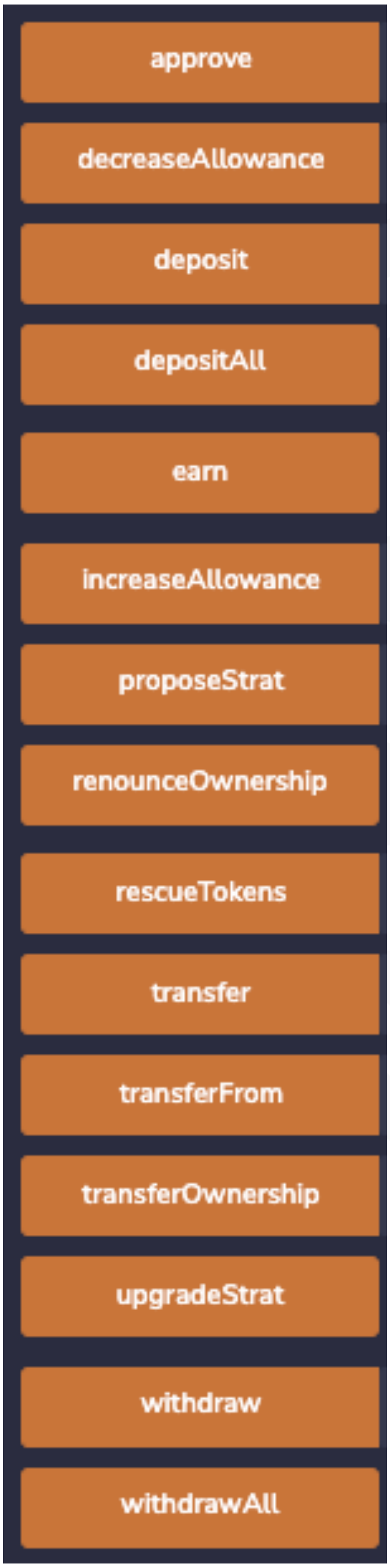
1. Is contract an upgradeable
2. Deployer cannot mint any new tokens
3. Deployer cannot burn or lock user funds
4. Deployer cannot pause the contract
5. Deployer cannot set fees
6. Deployer cannot blacklist/antisnipe addresses
7. Overall checkup (Smart Contract Security)

Is contract an upgradeable

Name	
Is contract an upgradeable?	No

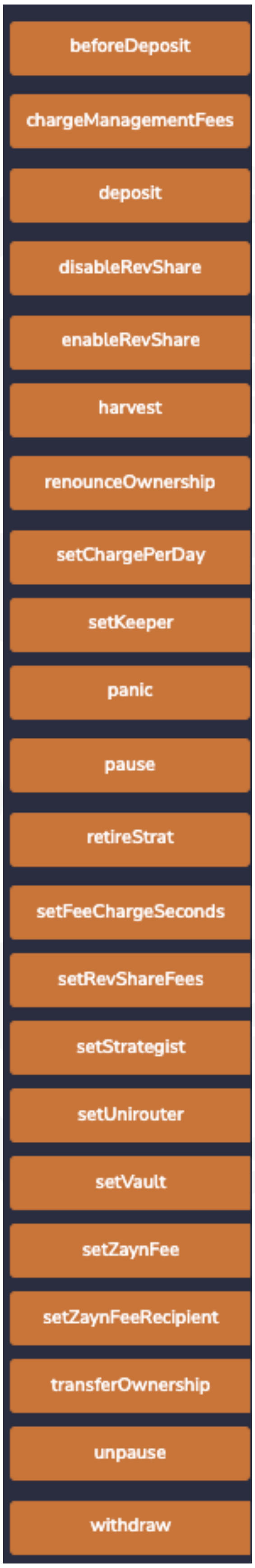
Write functions of contract v1.0

ZaynVault



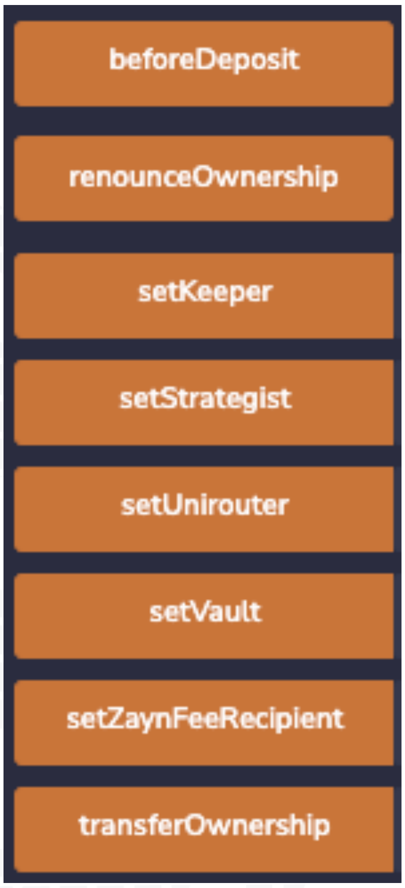
approve
decreaseAllowance
deposit
depositAll
earn
increaseAllowance
proposeStrat
renounceOwnership
rescueTokens
transfer
transferFrom
transferOwnership
upgradeStrat
withdraw
withdrawAll

Wombatstrategy



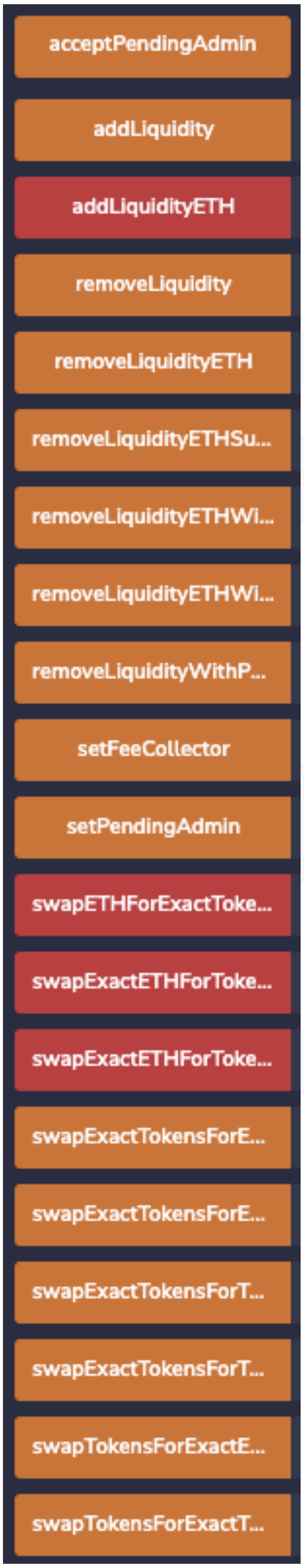
beforeDeposit
chargeManagementFees
deposit
disableRevShare
enableRevShare
harvest
renounceOwnership
setChargePerDay
setKeeper
panic
pause
retireStrat
setFeeChargeSeconds
setRevShareFees
setStrategist
setUnirouter
setVault
setZaynFee
setZaynFeeRecipient
transferOwnership
unpause
withdraw

StratManager



beforeDeposit
renounceOwnership
setKeeper
setStrategist
setUnirouter
setVault
setZaynFeeRecipient
transferOwnership

ZaynRouter


acceptPendingAdmin
addLiquidity
addLiquidityETH
removeLiquidity
removeLiquidityETH
removeLiquidityETHSu...
removeLiquidityETHWi...
removeLiquidityETHWi...
removeLiquidityWithP...
setFeeCollector
setPendingAdmin
swapETHForExactToke...
swapExactETHForToke...
swapExactETHForToke...
swapExactTokensForE...
swapExactTokensForE...
swapExactTokensForT...
swapExactTokensForT...
swapTokensForExactE...
swapTokensForExactT...

Deployer cannot mint any new tokens

Name	Exist	Tested	Status
Deployer cannot mint	✓	✓	✓

Comments:

v1.0

- Tokens will be minted while deposit in the vault contract

Deployer cannot burn or lock user funds

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

Comments:

v1.0

- ZaynDAIZap
  - Owner can lock zapIn function by setting paths to zero address because while swapping it tries to transferFrom this address. This cause a revert because address zero is not able to allow any token transfers
- ZaynReferrer
  - Owner is able to set the “delaySeconds” without any limitation. That causes that the block.timestamp must be higher than the last deposit time plus the delaySeconds otherwise you are not able to call the “claimBonusUser” function. The same applies to the “minAmountForBonus” variable which is also called in the claim function above. Additionally the owner is able to set the rewardToken address to zero/dead address that will also lock user funds because in the claimBonusUser function the Referrer contract is transferring the reward to the user of the set “rewardToken” which will not be possible. If the claimBonusUser function will be passed with the above conditions, the owner is still able to set the “rewardAmountUser” to 0 what means that the caller will get 0 tokens.
- Tokens
  - will be burned while withdrawing in the ZaynVault

Deployer cannot pause the contract

Name	Exist	Tested	Status
Deployer cannot pause	✓	✓	✗

Comments:

v1.0

- WombatStrategy
    - Owner can pause contract

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

- FeeManager
    - Fees can be set without any limitations

Deployer can blacklist/antisnipe addresses

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

Overall checkup (Smart Contract Security)

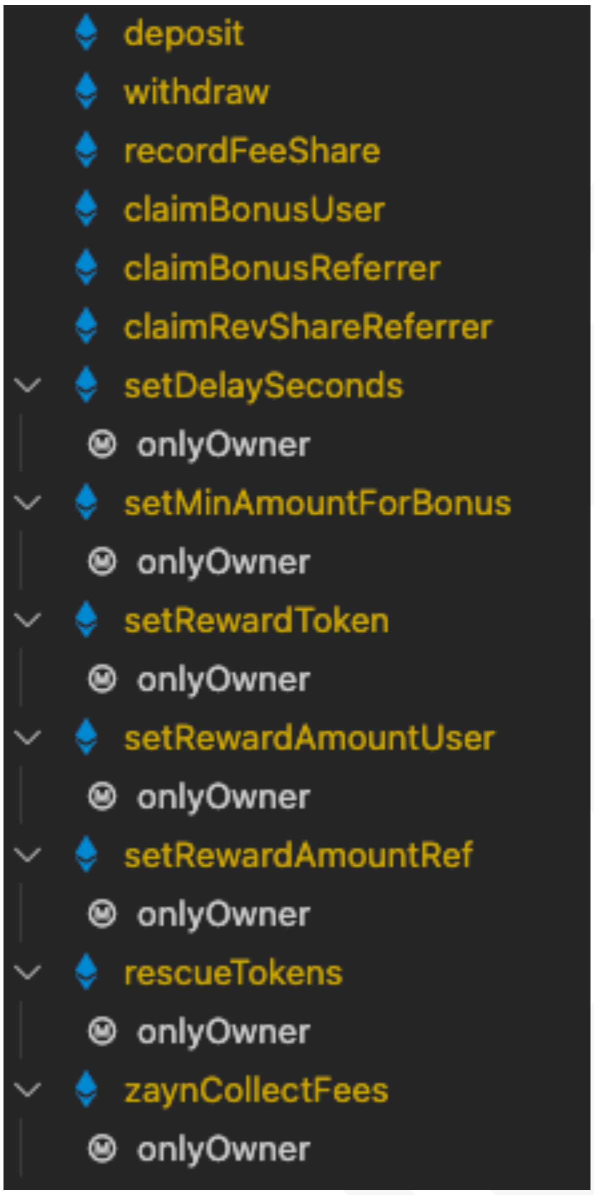
Tested	Verified
✓	✓

Legend

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

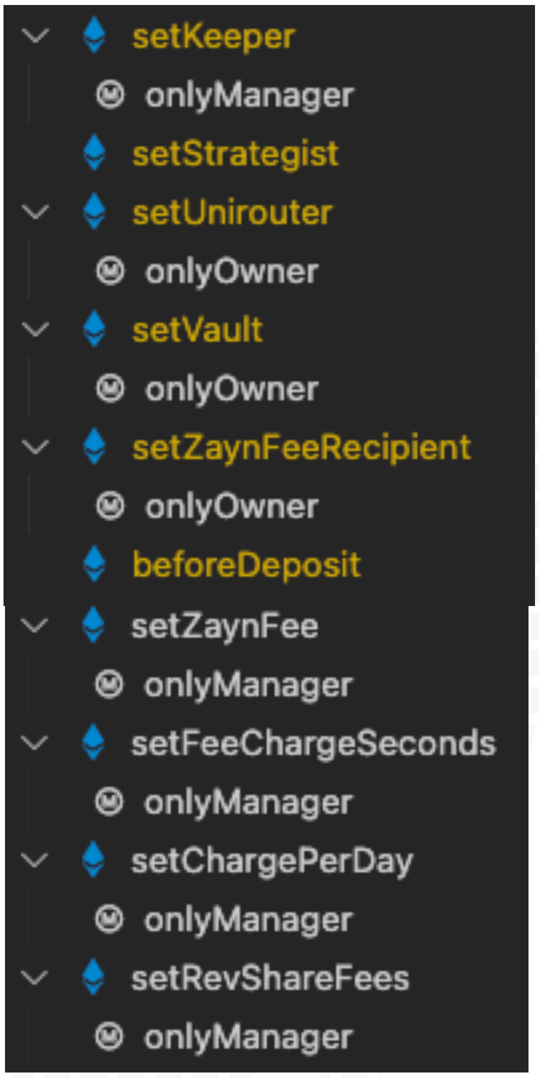
Modifiers and public functions
v1.0

ZaynReferrer



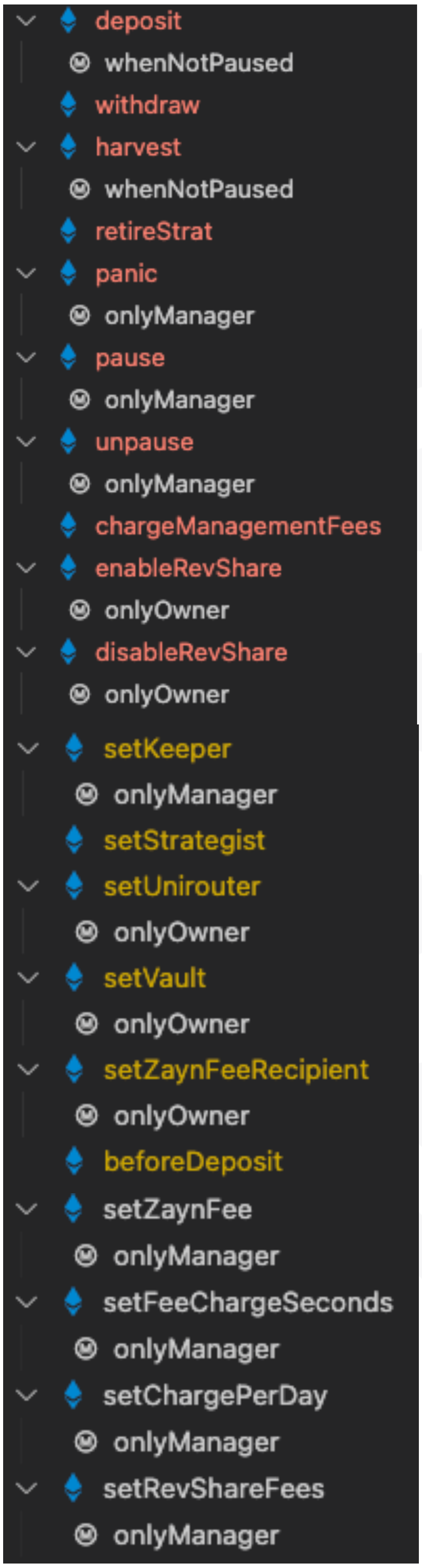
```
deposit
withdraw
recordFeeShare
claimBonusUser
claimBonusReferrer
claimRevShareReferrer
setDelaySeconds
    onlyOwner
setMinAmountForBonus
    onlyOwner
setRewardToken
    onlyOwner
setRewardAmountUser
    onlyOwner
setRewardAmountRef
    onlyOwner
rescueTokens
    onlyOwner
zaynCollectFees
    onlyOwner
```

StratManager



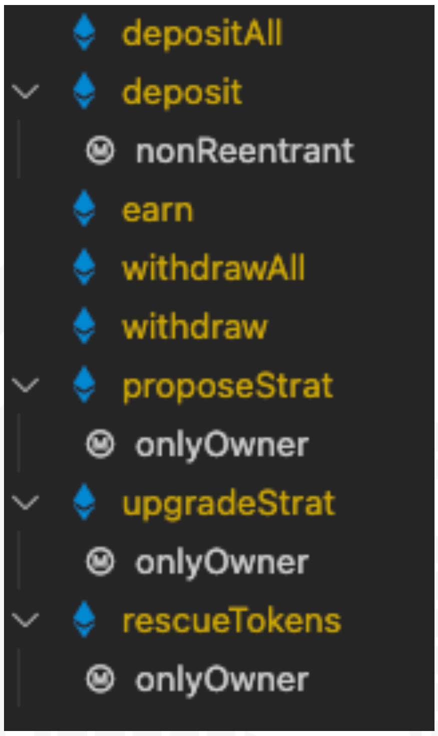
```
setKeeper
    onlyManager
setStrategist
setUnirouter
    onlyOwner
setVault
    onlyOwner
setZaynFeeRecipient
    onlyOwner
beforeDeposit
setZaynFee
    onlyManager
setFeeChargeSeconds
    onlyManager
setChargePerDay
    onlyManager
setRevShareFees
    onlyManager
```

WombatStrategy



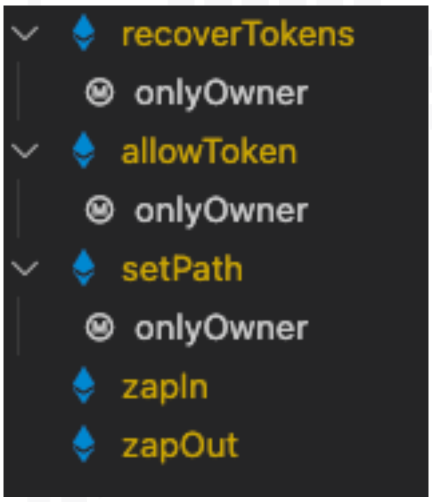
- deposit
  - whenNotPaused
- withdraw
- harvest
  - whenNotPaused
- retireStrat
- panic
  - onlyManager
- pause
  - onlyManager
- unpause
  - onlyManager
- chargeManagementFees
- enableRevShare
  - onlyOwner
- disableRevShare
  - onlyOwner
- setKeeper
  - onlyManager
- setStrategist
- setUnirouter
  - onlyOwner
- setVault
  - onlyOwner
- setZaynFeeRecipient
  - onlyOwner
- beforeDeposit
- setZaynFee
  - onlyManager
- setFeeChargeSeconds
  - onlyManager
- setChargePerDay
  - onlyManager
- setRevShareFees
  - onlyManager

ZaynVault



- depositAll
- deposit
  - nonReentrant
- earn
- withdrawAll
- withdraw
- proposeStrat
  - onlyOwner
- upgradeStrat
  - onlyOwner
- rescueTokens
  - onlyOwner

ZaynDAIZap



- recoverTokens
  - onlyOwner
- allowToken
  - onlyOwner
- setPath
  - onlyOwner
- zapIn
- zapOut

Note:

- Functions from imported libraries will not be listed here
- The ZaynRouter contract is the same as pancakeSwapRouter functions with the only difference that there will be taken fees while the following functions
    - swapExactTokensForTokens
    - swapTokensForExactTokens
    - swapExactETHForTokens
    - swapTokensForExactETH
    - swapExactTokensForETH
    - swapETHForExactTokens
    - swapExactTokensForTokensSupportingFeeOnTransferToken
    - swapExactETHForTokensSupportingFeeOnTransferTokens
    - swapExactTokensForETHSupportingFeeOnTransferTokens

Comments

- *Deployer can set following state variables without any limitations*
    - ZaynReferrer
        - rewardAmountReferrer
        - rewardAmountUser
        - minAmountForBonus
        - delaySeconds
    - FeeManager
        - zaynFee
            - Up to 100%
        - feeChargeSeconds
        - chargePerDay
        - revShareFees

- *Deployer can enable/disable following state variables*
    - Wombatstrategy
        - revShareEnabled
            - For enabling to transfer the rev share fees while charging fees
    - ZaynDAPZap
        - allowedTokens

- *Deployer can set following addresses*
    - ZaynReferrer
        - rewardToken
    - StratManager
        - onlyManager and owner can set
            - zaynFeeRecipient
            - vault
            - unirouter

            - keeper
    - Wombatstrategy
        - zaynReferrer
    - ZaynDAIZap
        - paths

- *Existing Modifiers*
    - ZaynRouter
        - ensure
    - StratManager
        - onlyManager

- ZaynReferrer
    - Owner is able to
        - Take out vault and revShareToken balance of the ZaynReferrer contract by calling rescueToken and passing the address of it.
    - While depositing any investor can set his/her own address from the wallet (not with the calling address) as referrer
- ZaynVault
    - Owner can
        - Propose new strategy
        - upgrade strategy

- There are several authorities which are authorized to call some functions, that means, if the owner is renounced, another address is still authorized to call functions
    - Be aware of this

v1.1
Comments
- Owner is able to enable/disable deposit function in the ZaynReferrer contract
- Added functions
    - setRevShareToken
        - to set the rev share token

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/router/ZaynRouter.sol	4	6	914	513	412	48	622	💰📤🧮
📝	contracts/referrer/ZaynReferrer.sol	1	—	171	171	135	2	104	📤
🔍	contracts/interfaces/IUniswapRouter.sol	—	1	63	6	3	1	23	💰
🔍	contracts/interfaces/IMasterWombatV2.sol	—	1	59	10	4	4	29	—
🔍	contracts/interfaces/IStrategy.sol	—	1	23	8	4	1	31	—
🔍	contracts/interfaces/IPool.sol	—	1	65	5	3	1	21	—
🔍	contracts/interfaces/IWombatLP.sol	—	1	36	7	4	1	33	—
🔍	contracts/interfaces/IWombatRouter.sol	—	1	19	5	3	1	5	—
🔍	contracts/interfaces/IZaynVault.sol	—	1	12	7	4	1	13	—
🔍	contracts/interfaces/IZaynReferrer.sol	—	1	10	7	4	1	7	—
📝	contracts/vaults/ZaynVault.sol	1	—	203	203	108	67	124	—
📝	contracts/zap/ZaynDAIZap.sol	1	—	122	121	88	7	122	—
📝	contracts/strategies/WombatStrategy.sol	1	—	207	207	143	19	174	📤
🎨	contracts/strategies/Common/FeeManager.sol	1	—	30	30	22	4	21	—
📝	contracts/strategies/Common/StratManager.sol	1	—	96	96	44	41	31	—
📝 📚 🔍 🎨	**Totals**	**10**	**14**	**2030**	**1396**	**981**	**199**	**1360**	💰📤🧮

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

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	See description	Choose a certain version of pragma instead of floating (usually started with "^", ">=" etc.
#3	WombatStrategy	Missing Zero Address Validation (missing-zero-check)	54	Check that the address is not zero
#4	StratManager	Missing Zero Address Validation (missing-zero-check)	37 38 39 40 41 55 64 72 80 88	Check that the address is not zero

#5	ZaynReferrer	Missing Zero Address Validation (missing-zero-check)	55 54 62 148	Check that the address is not zero
#6	ZaynRouter	Missing Zero Address Validation (missing-zero-check)	407 406 912 901	Check that the address is not zero
#7	ZaynReferrer	State variable visibility is not set	22	It is best practice to set the visibility of state variables explicitly
#8	WombatStrategy	Local variables shadowing	12 9	Rename the local variables that shadow another component
#9	ZaynReferrer	Local variables shadowing	34	Rename the local variables that shadow another component
#9	ZaynVault	Local variables shadowing	49, 50	Rename the local variables that shadow another component
#10	StratMamager	Missing Events Arithmetic	55	Emit an event for critical parameter changes
#11	FeeManager	Missing Events Arithmetic	24, 20	Emit an event for critical parameter changes
#12	ZaynReferrer	Missing Events Arithmetic	77 96 140 144 156 152 89	Emit an event for critical parameter changes

Informational issues

Issue	File	Type	Line	Description
#1	WombatStrategy	State variables that could be declared constant (constable-states)	28	Add the `constant` attributes to state variables that never change

#2	IMasterWombatV2	Misspelling	See description	Change following words: - transfered L38 Make sure to change it everywhere else as well.
#3	ZaynReferrer	Misspelling	See description	Change following words: - eligble L114, L102 Make sure to change it everywhere else as well.
#4	ZaynRouter	Change error messages	See description	Replace “PancakeRouter” with “ZaynRouter” for a better overlook
#5	ZaynReferrer	Unecessary visibility	53	Remove public visibility from constructor
#6	WombatStrategy	Unecessary visibility	53	Remove public visibility from constructor
#7	ZaynVault	Unecessary visibility	52	Remove public visibility from constructor
#8	ZaynReferrer	Visibility first	128, 132	Visibility modifier “public” should come before other modifiers
#9	Wombatstrategy/ FeeManager	Check zaynFee for 0/ revShareFees for 0	See description	If the zaynFee is zero and the revShare is enabled all zaynFees’s will be sent to the zaynFeeRecipient. We recommend you to check also the zaynFee is 0 in L109. Additionally the owner is able to set it to MAX_FEE (1000 = 100%) to send all fees to zaynReferrer in FeeManager contract L16. Also the owner is able to set the zaynReferrer to an arbitrary address in L201.

#10	Wombatstrategy/ FeeManager	feeChargeSeconds can lock charge management fees	See description	In L186 (Womberstrategy) the chargeManagementFees function can only be called when the block.timestamp is higher than the lastFeeCharge + the feeChargeSeconds. The owner is able to lock this function by setting a too high value for the “feeChargeSeconds” variable in FeeManager contract L20. If the seconds are set to 0 the fees in the chargeManagementFees function L191 will be 0 also.
#11	StratManager	Strategist has no functionality in the contract	64	Remove or use the state variable. Even the setStrategist function was not used from the outside of the contract.
#12	WombatStrategy	Same function call	83-88	Check the if/else condition. They are the same logic in the contract.

Alleviations

Medium issues

#1 ZaynReferrer
Type: Owner is able to drain out contracts

Description:
The owner is able to call the rescueTokens function and pass the rewardToken/revShareToken address to it to drain out these contracts.

We recommend you to prevent passing these addresses to the function.

Alleviation:
Given user’s capital is in vault tokens we safeguard against that rewardToken is topped up by us and it makes sense to have ability to rescue and revShareToken is sent extra by strategy and makes sense to have ability to rescue.

We will put these functions behind Multisig, so you can mark them as centralization risk and mitigation is to put behind msig

Commented Code exist

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

File	Line	Comment
Wombat Strategy	86	// uint256 withdrawalFeeAmount = wantBal.mul(withdrawalFee).div(WITHDRAWAL_MAX);

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/latest/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.

23. November 2022:

- Masterchef and Pools were not provided to solidproof. Please dyor here.
- 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	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 Development | Marketing

MADE IN GERMANY