SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY

Shib Connect

Audit

Security Assessment
01. March, 2023

For

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	28. February 2023	• Layout project • Automated- /Manual-Security Testing • Summary
1.1	01. March 2023	• Reaudit

Network
Binance Smart Chain (BEP20)

Website
https://www.shibconnect.com/

Telegram
https://t.me/shibconnect

Twitter
https://twitter.com/Shibconnectbsc

Description

ShibConnect was developed with the intent to start a new trend within crypto. This idea was created the idea after realizing just how effective referral systems are in real-world businesses. But the blockchain allows for something that hasn't even been done before in these businesses. It allows for fully transparent and instant payouts of referral rewards, and the potential to scale to sizes that are unheard of when compared to multi-level-marketing companies that sell physical products.

ShibConnect is the first-ever token with built-in instant referral rewards and a unique referral train system.

The core features of the Shibconnect is the referral system integrated into the contract and dashboard and also it's NFT ecosystem paired along with it which will be explained in more detail.

Project Engagement

During the 25th of February 2023, **Shib Connect 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

v1.1

- Provided 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.

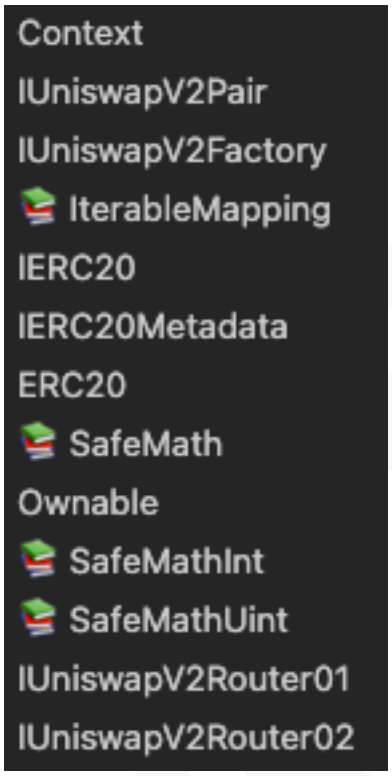
2. Testing and automated analysis that includes the following:
    i) Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
    ii) Symbolic execution, which is analysing a program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



Context
IUniswapV2Pair
IUniswapV2Factory
IterableMapping
IERC20
IERC20Metadata
ERC20
SafeMath
Ownable
SafeMathInt
SafeMathUint
IUniswapV2Router01
IUniswapV2Router02

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.

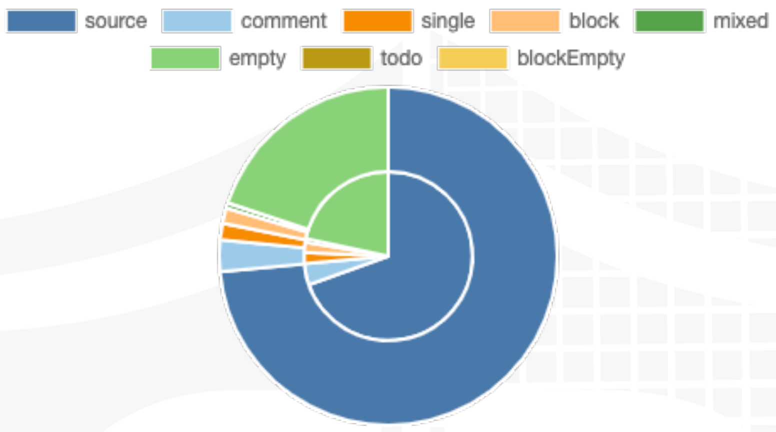
v1.0

File Name	SHA-1 Hash
contracts/shibconnect.sol	27147127a78eeea727e3c23f3a80af18a430e0cd

Metrics

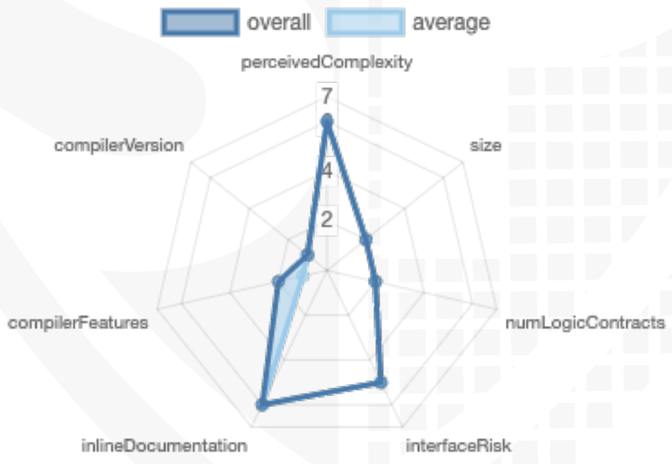
Source Lines

v1.0



Risk Level

v1.0



Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	3	4	6	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	133	5

Version	External	Internal	Private	Pure	View
1.0	77	115	10	25	59

State Variables

Version	Total	Public
1.0	71	37

Capabilities

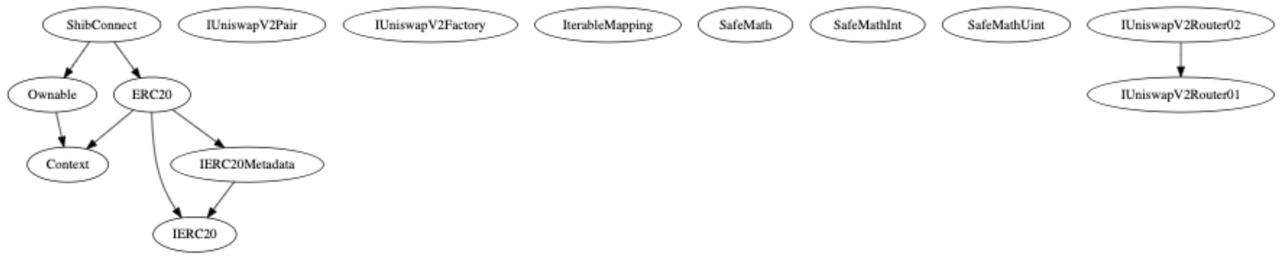
Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	`^0.8.17`		`yes`		

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

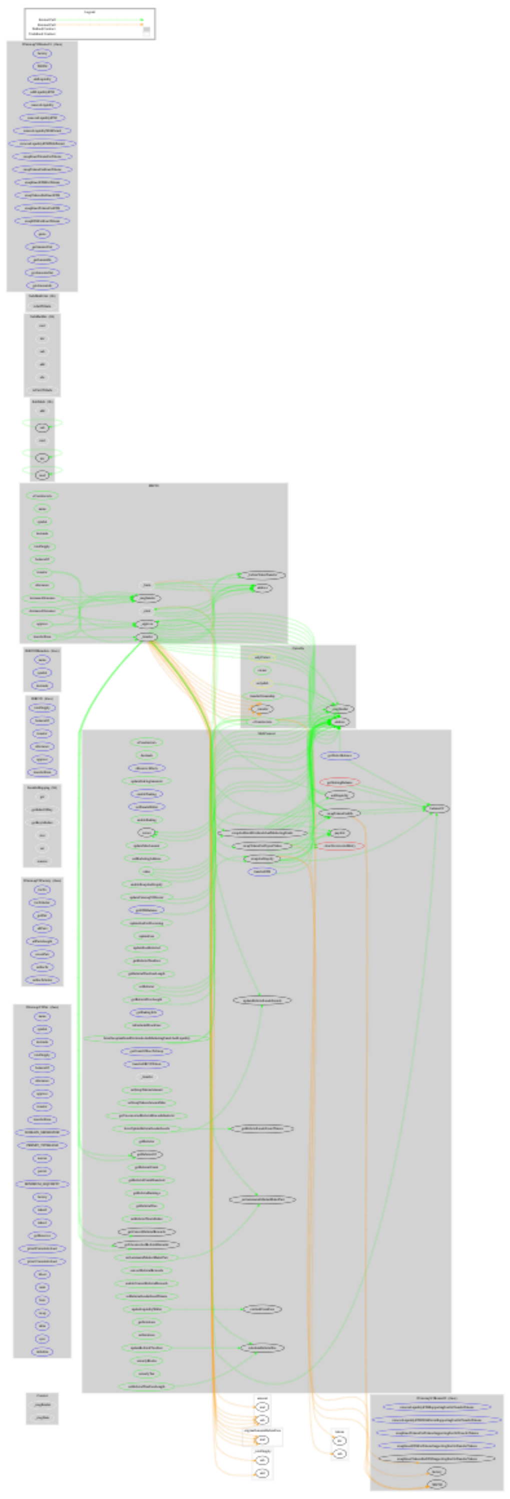
1.0	yes					

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. Correct implementation of Token standard
3. Deployer cannot mint any new tokens
4. Deployer cannot burn or lock user funds
5. Deployer cannot pause the contract
6. Deployer cannot set fees
7. Deployer cannot blacklist/antisnipe addresses
8. Overall checkup (Smart Contract Security)

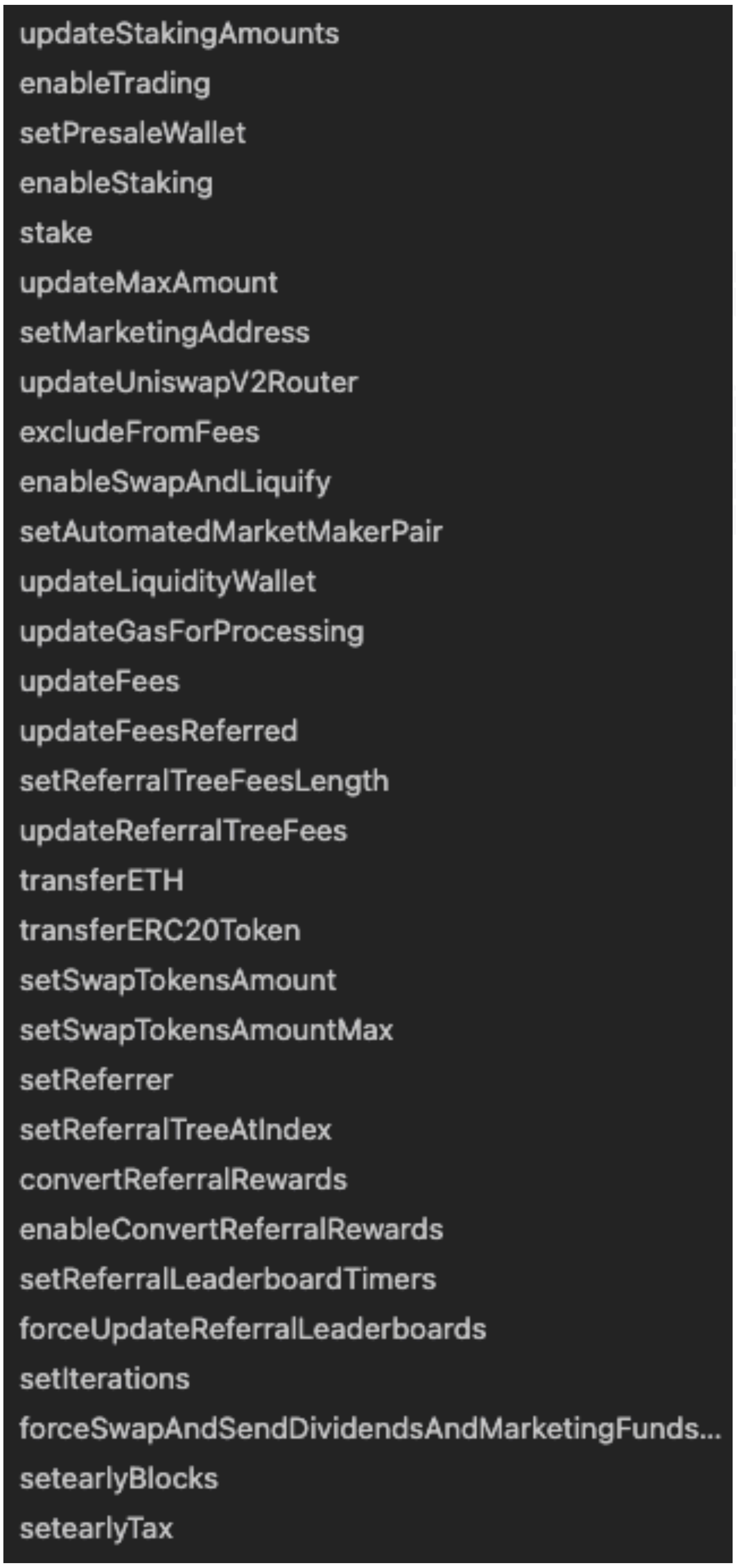
Is contract an upgradeable

Name	
Is contract an upgradeable?	No

Correct implementation of Token standard

ERC20				
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 v1.0



updateStakingAmounts
enableTrading
setPresaleWallet
enableStaking
stake
updateMaxAmount
setMarketingAddress
updateUniswapV2Router
excludeFromFees
enableSwapAndLiquify
setAutomatedMarketMakerPair
updateLiquidityWallet
updateGasForProcessing
updateFees
updateFeesReferred
setReferralTreeFeesLength
updateReferralTreeFees
transferETH
transferERC20Token
setSwapTokensAmount
setSwapTokensAmountMax
setReferrer
setReferralTreeAtIndex
convertReferralRewards
enableConvertReferralRewards
setReferralLeaderboardTimers
forceUpdateReferralLeaderboards
setIterations
forceSwapAndSendDividendsAndMarketingFunds...
setearlyBlocks
setearlyTax

Deployer cannot mint any new tokens

Name	Exist	Tested	Status
Deployer cannot mint	✓	✓	✓
Max / Total Supply	1000000000		

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 setting to high earlyTax. It is possible to set it above 100%
- referralFee can be set above 100%
- Owner can lock
    - staking by setting staking amounts to 0
    - By setting referralFee above 100

Deployer cannot pause the contract

Name	Exist	Tested	Status
Deployer cannot pause	–	–	–

Comments:

v1.0

- 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

- Fees can be set without any limitations. The owner is able to reset earlyBlock variable everytime he wants. According to this the owner can set the earlyTax to 100%
- referralFee can be set above 100%

Deployer can blacklist/antisnipe addresses

Name	Exist	Tested	Status
Deployer cannot blacklist/antisnipe addresses	✓	✓	✗

Comments:

v1.0

- There is no antisnipe function directly but the owner is able to set the Early blocks and early tax to any values. That means the early block can be reactivated by the owner at any time.

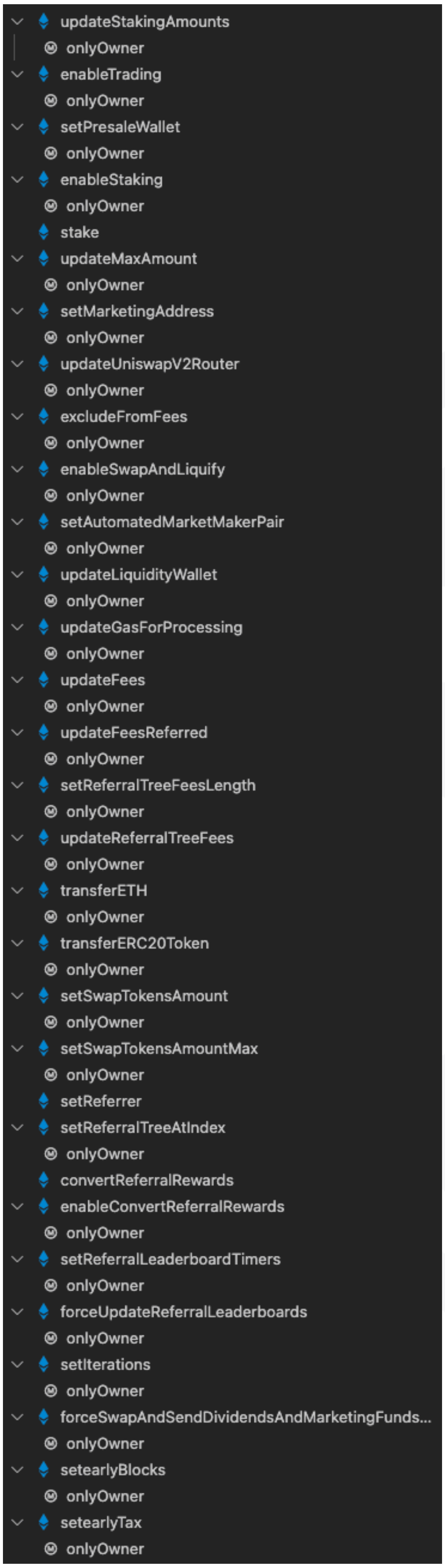
Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

Legend

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

Modifiers and public functions
v1.0


```
updateStakingAmounts
  onlyOwner
enableTrading
  onlyOwner
setPresaleWallet
  onlyOwner
enableStaking
  onlyOwner
stake
updateMaxAmount
  onlyOwner
setMarketingAddress
  onlyOwner
updateUniswapV2Router
  onlyOwner
excludeFromFees
  onlyOwner
enableSwapAndLiquify
  onlyOwner
setAutomatedMarketMakerPair
  onlyOwner
updateLiquidityWallet
  onlyOwner
updateGasForProcessing
  onlyOwner
updateFees
  onlyOwner
updateFeesReferred
  onlyOwner
setReferralTreeFeesLength
  onlyOwner
updateReferralTreeFees
  onlyOwner
transferETH
  onlyOwner
transferERC20Token
  onlyOwner
setSwapTokensAmount
  onlyOwner
setSwapTokensAmountMax
  onlyOwner
setReferrer
setReferralTreeAtIndex
  onlyOwner
convertReferralRewards
enableConvertReferralRewards
  onlyOwner
setReferralLeaderboardTimers
  onlyOwner
forceUpdateReferralLeaderboards
  onlyOwner
setIterations
  onlyOwner
forceSwapAndSendDividendsAndMarketingFunds...
  onlyOwner
setearlyBlocks
  onlyOwner
setearlyTax
  onlyOwner
```

Comments

- *Deployer can set following state variables without any limitations*
    - earlyTax
    - earlyBlocks
    - iteration
    - iterationDaily
    - iterationWeekly
    - iterationMonthly
    - dailyTimer
    - weeklyTimer
    - monthlyTimer
    - swapTokensAtAmountMax
    - swapTokensAtAmount
    - referralTreeFees
    - devFeesReferred
    - liquidityFeeReferred
    - devFees
    - liquidityFee
    - referralFee
    - maxSellTransactionAmount
    - stakingAmounts

- *Deployer can enable/disable following state variables*
    - enableConvertingReferralRewards
    - automatedMarketMakerPairs
    - swapAndLiquifyEnabled
    - _isExcludedFromFees
    - stakingEnabled
    - canTransferBeforeTradingIsEnabled
    - _isExcludedFromFees

- *Deployer can set following addresses*
    - referrerTree
    - liquidityWallet
    - uniswapV2Router
    - marketingAddress

- *Existing Modifiers*
    - onlyshib
    - onlyOwner

- Owner is able to
    - Manually swap
    - Force update leader boards
    - Drain out the contract tokens

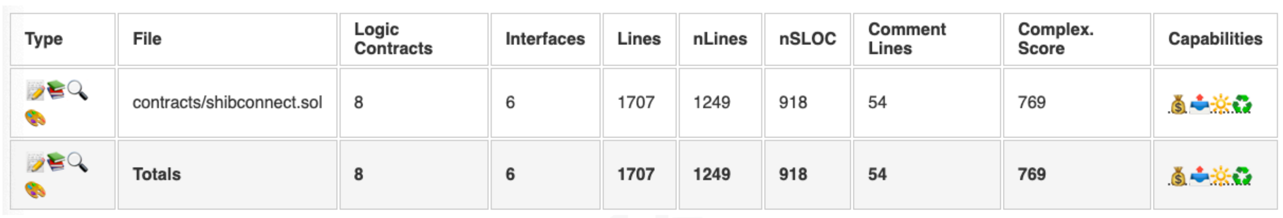
    - Take out contract address balance
    - Enable/disable liquidity adding

- We recommend you to change every
    - uint to uint256
    - Int256 to uin256

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/shibconnect.sol	8	6	1707	1249	918	54	769	💰📥☀️♻️
📝📚🔍🎨	**Totals**	**8**	**6**	**1707**	**1249**	**918**	**54**	**769**	💰📥☀️♻️

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 meium 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	Main	A floating pragma is set	3	The current pragma Solidity directive is „"^0.8.17"".
#3	Main	Missing Zero Address Validation (missing-zero-check)	1064, 1188, 1110	Check that the address is not zero
#4	Main	Missing Events Arithmetic	1645 1646 1647 1648 1630 1631 1632 1161 1419 1424 1705 1711 1057	Emit an event for critical parameter changes

#5	Main	Missing path	1491	path.length must be >= 2. Pools for each consecutive pair of addresses must exist and have liquidity.
#6	Main	AfterSwapDelta will be 0	1684, 1685	Since there are no actions between the beforeSwap variable and afterSwapDelta calculation the afterSwapDelta will be 0 because beforeSwap is only the balance of the contract and the value subtracted by itself will be 0. We recommend you to remove those lines because the swap will be executed from L1689-1691
#7	Main	Wrong comment or logic	1703, 1709	Modify the error message or adjust the logic. earlyBlocks can must be more then 5 here and earlyTax must be more than 25-

Informational issues

Issue	File	Type	Line	Description
#1	Main	State variables that could be declared constant (constable-states)	1208, 1209	Add the `constant` attributes to state variables that never change
#2	Main	Functions that are not used	1380	Remove unused functions. Before removing check the function, it could be possible, that you forget to implement it into the contract
#3	Main	Unused state variables	572	Remove unused state variables

#4	Main	Error message is missing	See description	Provide an error message for require statement Check all require statements whether they don't have an error message otherwise provide one
#5	Main	NatSpec documentation missing	-	If you started to comment your code, also comment all other functions, variables etc.
#6	Main	State variables that could be declared immutable	845	Add the `immutable` attribute to state variables that never change or are set only in the constructor

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.

01. March 2023:

- The payout will not work for the contract while swapping tokens for payout because 1 path is missing but 2 is require here. For details look low issue #5 missing path
- Read whole report carefully 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	**PASSED**
SWC-118	Incorrect Constructor Name	CWE-665: Improper Initialization	**PASSED**
SWC-117	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	**PASSED**

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

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

SolidProof_io

@solidproof_io

Solid Proofed

Blockchain Security | Smart Contract Audits | KYC
Development | Marketing

MADE IN GERMANY