Revoluzion Audit

Audit Report

Name	: **ShibConnect**
Symbol	: **ShibConnect**
Decimals	: **9**
Address	: **0x341C7dD7ac0C501C21D87a68Fcd30Bd50C314230**
Owner	: **0x13331eE13580D9cC961DCa5De3D8d50e332c18Da**
Network	: **Binance Smart Chain (Mainnet)**
Type	: **BEP20**
Audited on	: **4 March 2023**
Updated on	: **10 March 2023**

Revoluzion Audit

Contents

Project Overview

Name	ShibConnect
Symbol	ShibConnect
Decimals	9
Total Supply	1,000,000,000
Tax	Early Tax — Refer notable information Buy 0% | Sell 0% — (Variable Tax: Max Buy starting from 25% | Max Sell starting from 25%) Normal — Refer notable information Buy 9% | Sell 9% — (Variable Tax: Max Buy starting from 17% | Max Sell starting from 17%) Referred — Refer notable information Buy 7% | Sell 7% — (Variable Tax: Max Buy starting from 22% | Max Sell starting from 22%)
Compiler Version	v0.8.17+commit.8df45f5f
Optimization	Yes with 5 runs
License Type	Unlicensed
Explorer Link	https://bscscan.com/address/0x341c7dd7ac0c501c21d87a68fcd30bd50c314230
Create Tx	0x6289a137689d1fccd3388c4339f5229fadff340693bb6267a0a7c2e18017e0d4
Creator	0x13331eE13580D9cC961DCa5De3D8d50e332c18Da

Featured Wallet	Marketing Wallet — 0x5dEBAC6Dcda515C67f430FF7627b3867E999C468 Liquidity Wallet — 0x13331eE13580D9cC961DCa5De3D8d50e332c18Da
Website	https://shibconnect.com

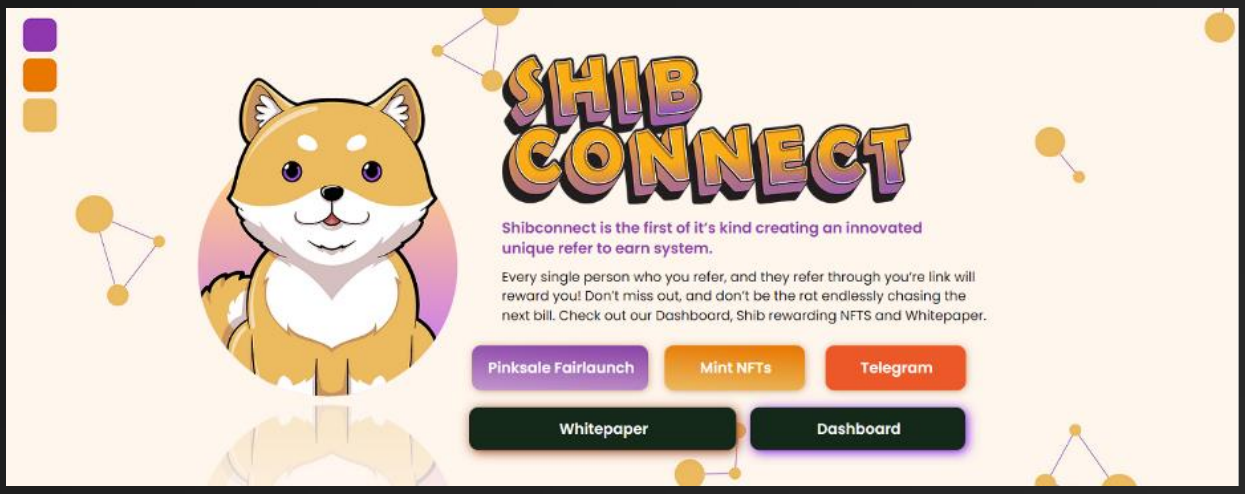
Project Description

According to their website

Shibconnect is the first of its kind creating an innovated unique refer to earn system. Every single person who you refer, and they refer through your link will reward you! Don't miss out, and don't be the rat endlessly chasing the next bill. Check out our Dashboard, Shib rewarding NFTS and Whitepaper.

Release Date : TBA

Category : Refer to Earn System



Online Presence

About Website

Registrar : https://www. namecheap.com

Domain Expiration : 2024-02-23

SSL Certificate : Issued by Sectigo Limited

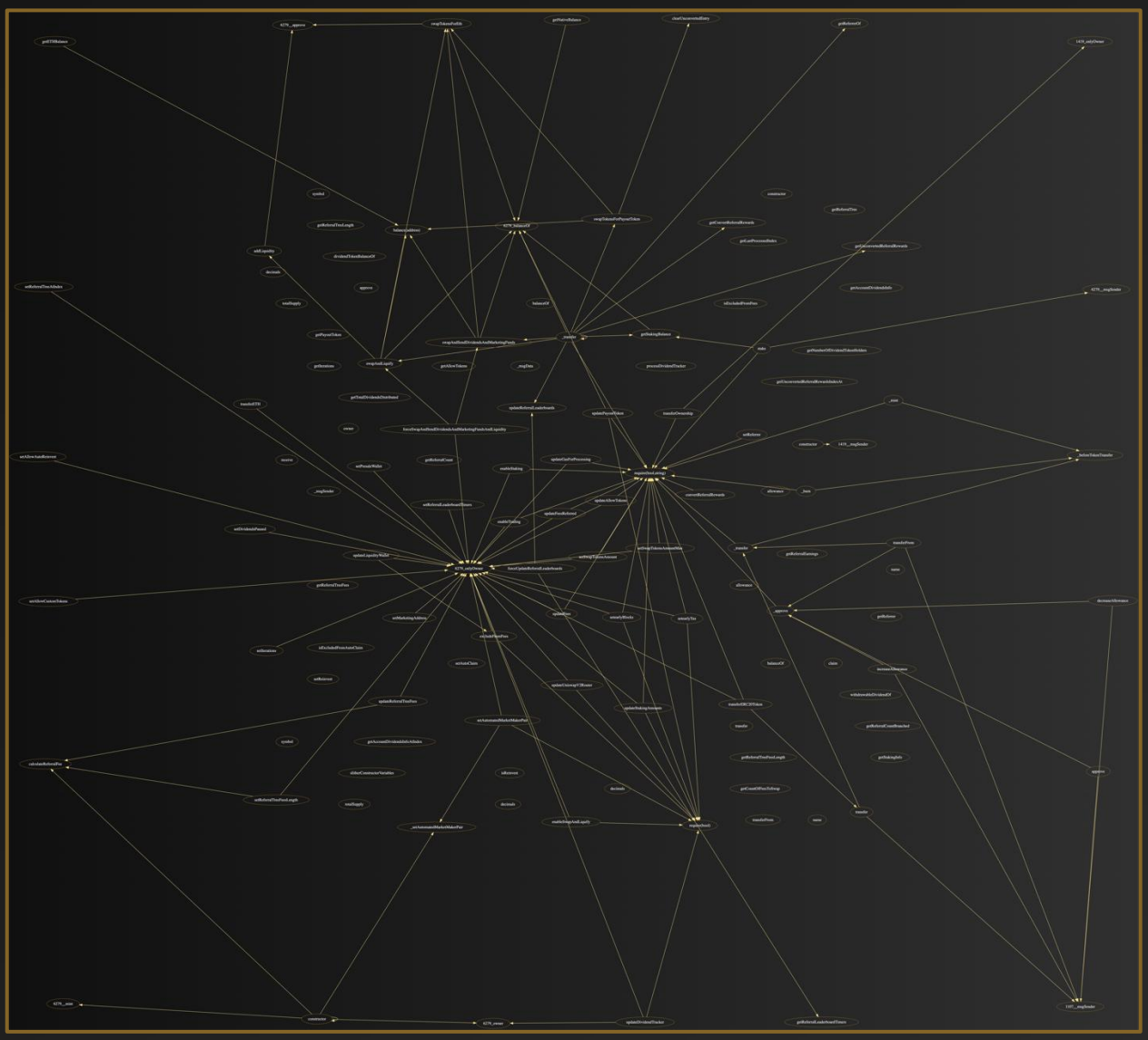
Official Links

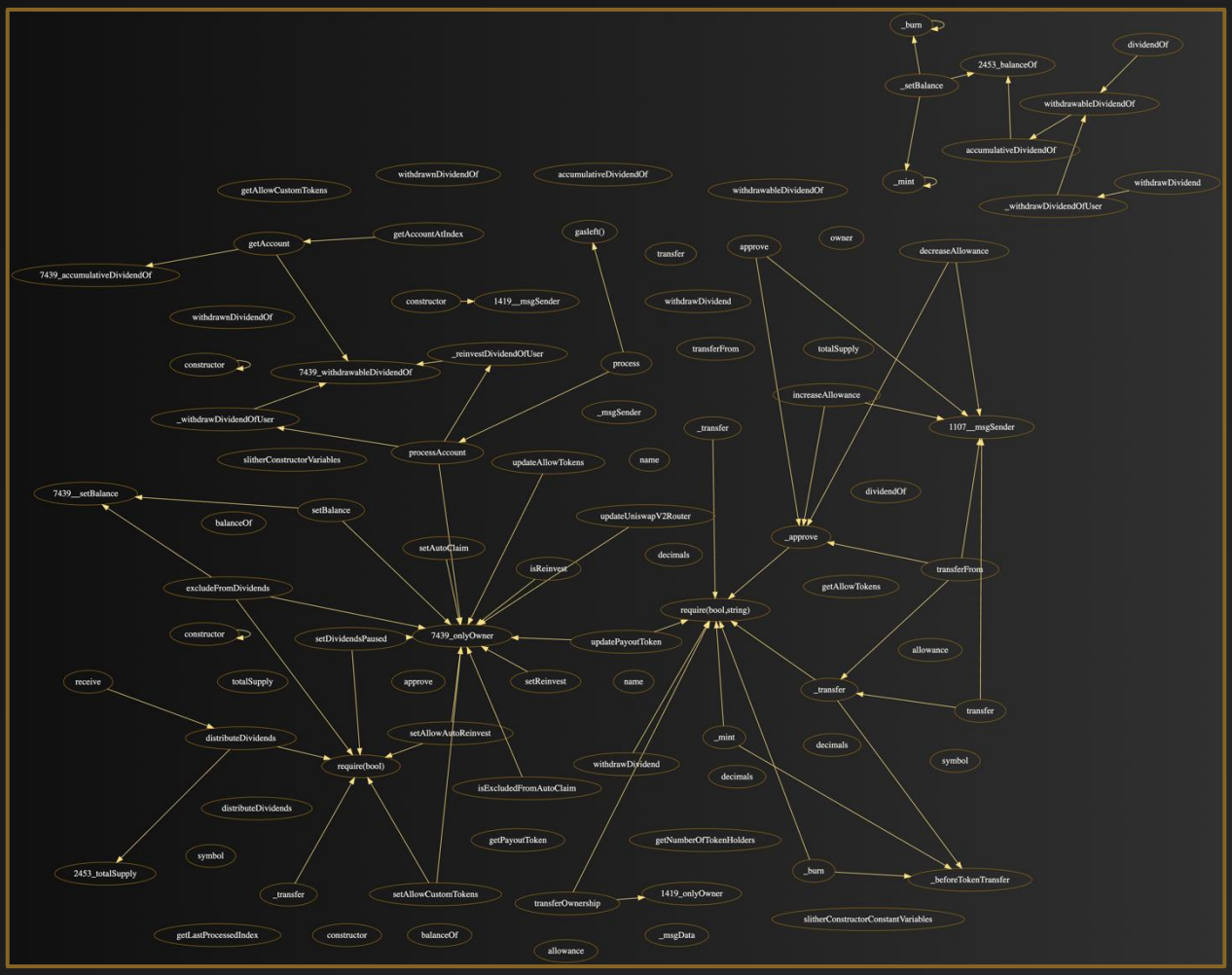
Website	**https://shibconnect.com**
Twitter	**https://twitter.com/shibconnectbsc**
Telegram	**https://t.me/shibconnect**
Facebook	**https://www.facebook.com/shibconnect**
GitBook	**https://shibconnect.gitbook.io/shibconnect.com**
GitHub	**https://github.com/ShibConnect**

The Team

About	We only interacted with the owner for the audit. However, there are no KYC procedure being conducted by Revoluzion on any of ShibConnect's team members.
KYC Issuer	N/A
Member's KYC'd	N/A
KYC Date	N/A
Certificate Link	N/A
Task Completed	N/A

Contract Functions Interaction




_burn
dividendOf
2453_balanceOf
_setBalance
withdrawableDividendOf
accumulativeDividendOf
withdrawnDividendOf
accumulativeDividendOf
getAllowCustomTokens
withdrawableDividendOf
_mint
withdrawDividend
_withdrawDividendOfUser
getAccount
getAccountAtIndex
gasleft()
transfer
approve
owner
decreaseAllowance
7439_accumulativeDividendOf
constructor
1419__msgSender
withdrawDividend
withdrawnDividendOf
transferFrom
totalSupply
constructor
_reinvestDividendOfUser
7439_withdrawableDividendOf
process
increaseAllowance
_withdrawDividendOfUser
_msgSender
_transfer
1107__msgSender
slitherConstructorVariables
processAccount
updateAllowTokens
name
7439__setBalance
dividendOf
setBalance
updateUniswapV2Router
balanceOf
_approve
setAutoClaim
decimals
isReinvest
transferFrom
excludeFromDividends
getAllowTokens
require(bool,string)
constructor
setDividendsPaused
7439_onlyOwner
updatePayoutToken
allowance
receive
totalSupply
approve
setReinvest
name
_transfer
transfer
distributeDividends
setAllowAutoReinvest
_mint
decimals
symbol
require(bool)
withdrawDividend
decimals
isExcludedFromAutoClaim
distributeDividends
getPayoutToken
getNumberOfTokenHolders
symbol
_burn
_beforeTokenTransfer
2453_totalSupply
_transfer
setAllowCustomTokens
transferOwnership
1419_onlyOwner
slitherConstructorConstantVariables
getLastProcessedIndex
constructor
balanceOf
_msgData
allowance

Audit Overview

Threat Level

When conducting audit on smart contract(s), we first look for known vulnerabilities and issues within the code because any exploitation on such vulnerabilities and issues by malicious actors could potentially result in serious financial damage to the projects. All the issues and vulnerabilities will be categorized into the categories as provided below.

Critical

This category provides issues and vulnerabilities that are critical to the performance/functionality of the smart contract and should be fixed by project creator before moving to a live environment.

Medium

This category provides issues and vulnerabilities that are not that critical to the performance/functionality of the smart contract but is recommended to be fixed by project creator before moving to a live environment.

Minor

This category provides issues and vulnerabilities that are minor to the performance/functionality of the smart contract and can remain unfixed by project creator before moving to a live environment.

Informational

This category provides issues and vulnerability that have insignificant effect on the performance/functionality of the smart contract and can remain unfixed by project creator before moving to a live environment. However, fixing them can further improve the efficacy or security for features with a risk-free factor.

Notable Information

- Contract Owner cannot stop or pause transactions.
- Contract Owner cannot transfer tokens from specific address.
- Contract Owner cannot mint new tokens after deploying smart contract.
- Contract Owner cannot burn tokens from specific wallet.
- Both buy and sell fees are set to be a total of 9% for normal transaction.
- Both buy and sell fees are set to be a total of 7% for referred transaction.
- Both buy and sell fees are set to be a total of 4% for early transaction.
- Both buy and sell fees can be change to a maximum total that starts from 17% for normal transaction, please refer caution section for the explanation.
- Both buy and sell fees can be change to a maximum total that starts from 22% for normal transaction, please refer caution section for the explanation.
- Both buy and sell fees can be change to a maximum total that starts from 25% for early transaction, please refer caution section for the explanation.
- There no limit for sell transaction since the max sell transaction amount was set to be the amount of the total supply.
- Contract Owner cannot blacklist wallets from selling.
- There are no compiler warnings when compiling the smart contracts.
- Contract is using interface from safe Zeppelin modules.

Caution

- The project owner must manually enable trading. Without this action, transactions will be prohibited except for the owner's account and those wallets that have been approved during the presale whitelist.

- Currently, there is no limit on the maximum amount for sell transactions as it is set to the total supply. Nonetheless, the project owner holds the authority to adjust this figure, reducing it to as low as 1% of the total supply.

- It has been identified that calling the updateReferralTreeFees() function could lead to a potential bug in the total fee calculation. This is because the function does not impose any restrictions on the total referral fee that can be set by the project owner, which should be a maximum of 3% as defined in the update fee function. As a result, in certain scenarios, the total fee imposed could exceed 100% if multiple tiers are added. However, if the project owner updates the referral tree for tiers 1-3, the total referral fee will not exceed 0.09% as each tier is currently limited to a maximum of 0.03%, would not be an issue If tax and index remain constant.

Bugs and Optimizations Detection

This table is based on the result obtained from running the smart contract through Slither's Solidity static analysis.

What it detects	Impact	Confidence	Status
Storage abiencoderv2 array	High	High	Passed
transferFrom uses arbitrary from	High	High	Passed
Modifying storage array by value	High	High	Passed
The order of parameters in a shift instruction is incorrect.	High	High	Passed
Multiple constructor schemes	High	High	Passed
Contract's name reused	High	High	Passed
Detected unprotected variables	High	High	Passed
Public mappings with nested variables	High	High	Passed
Right-To-Left-Override control character is used	High	High	Passed
State variables shadowing	High	High	Passed
Functions allowing anyone to destruct the contract	High	High	Passed
Uninitialized state variables	High	High	Passed
Uninitialized storage variables	High	High	Passed

Unprotected upgradeable contract	High	High	Passed
transferFrom uses arbitrary from with permit	High	Medium	Passed
Functions that send Ether to arbitrary destinations	High	Medium	Moderated
Tainted array length assignment	High	Medium	Passed
Controlled delegatecall destination	High	Medium	Passed
Payable functions using delegatecall inside a loop	High	Medium	Passed
msg.value inside a loop	High	Medium	Passed
Reentrancy vulnerabilities (theft of ethers)	High	Medium	Moderated
Signed storage integer array compiler bug	High	Medium	Passed
Unchecked tokens transfer	High	Medium	Moderated
Weak PRNG	High	Medium	Passed
Detects ERC20 tokens that have a function whose signature collides with EIP-2612's DOMAIN_SEPARATOR()	Medium	High	Passed
Detect dangerous enum conversion	Medium	High	Passed
Incorrect ERC20 interfaces	Medium	High	Passed
Incorrect ERC721 interfaces	Medium	High	Passed
Dangerous strict equalities	Medium	High	Passed

Contracts that lock ether	Medium	High	Passed
Deletion on mapping containing a structure	Medium	High	Passed
State variables shadowing from abstract contracts	Medium	High	Passed
Tautology or contradiction	Medium	High	Passed
Unused write	Medium	High	Passed
Misuse of Boolean constant	Medium	Medium	Passed
Constant functions using assembly code	Medium	Medium	Passed
Constant functions changing the state	Medium	Medium	Passed
Imprecise arithmetic operations order	Medium	Medium	Passed
Reentrancy vulnerabilities (no theft of ethers)	Medium	Medium	Moderated
Reused base constructor	Medium	Medium	Passed
Dangerous usage of tx.origin	Medium	Medium	Passed
Unchecked low-level calls	Medium	Medium	Passed
Unchecked send	Medium	Medium	Passed
Uninitialized local variables	Medium	Medium	Moderated
Unused return values	Medium	Medium	Passed
Modifiers that can return the default value	Low	High	Passed

Built-in symbol shadowing	Low	High	Passed
Local variables shadowing	Low	High	Moderated
Uninitialized function pointer calls in constructors	Low	High	Passed
Local variables used prior their declaration	Low	High	Moderated
Constructor called not implemented	Low	High	Passed
Multiple calls in a loop	Low	Medium	Moderated
Missing Events Access Control	Low	Medium	Passed
Missing Events Arithmetic	Low	Medium	Moderated
Dangerous unary expressions	Low	Medium	Passed
Missing Zero Address Validation	Low	Medium	Moderated
Benign reentrancy vulnerabilities	Low	Medium	Moderated
Reentrancy vulnerabilities leading to out-of-order Events	Low	Medium	Moderated
Dangerous usage of block.timestamp	Low	Medium	Moderated
Assembly usage	Informational	High	Passed
Assert state change	Informational	High	Passed
Comparison to boolean constant	Informational	High	Passed
Deprecated Solidity Standards	Informational	High	Passed
Un-indexed ERC20 event parameters	Informational	High	Passed

Function initializing state variables	Informational	High	Passed
Low level calls	Informational	High	Moderated
Missing inheritance	Informational	High	Passed
Conformity to Solidity naming conventions	Informational	High	Moderated
If different pragma directives are used	Informational	High	Passed
Redundant statements	Informational	High	Moderated
Incorrect Solidity version	Informational	High	Moderated
Unimplemented functions	Informational	High	Passed
Unused state variables	Informational	High	Moderated
Costly operations in a loop	Informational	Medium	Passed
Functions that are not used	Informational	Medium	Moderated
Reentrancy vulnerabilities through send and transfer	Informational	Medium	Passed
Variable names are too similar	Informational	Medium	Moderated
Conformance to numeric notation best practices	Informational	Medium	Moderated
State variables that could be declared constant	Optimization	High	Moderated
Public function that could be declared external	Optimization	High	Passed

Contract Diagnostic

Link for initial smart contract being audited on Binance Smart Chain (Testnet):

https://testnet.bscscan.com/address/0xc9db25a72060de4e6a774bb8da96843a8f5daee3

CODE	SEVERITY	DESCRIPTION
SWC-101	**Unknown**	Compiler-rewritable "<uint> - 1" discovered.
SWC-110	**Unknown**	Out of bounds array access.
SWC-115	**Minor**	Use of "tx.origin" as a part of authorization control.
SWC-120	**Minor**	Potential use of "block.number" as source of randomness.
CaL	**Minor**	Loops with multiple calls.
UL	**Informational**	Uninitialized local variables.
UR	**Informational**	Unused return.
SL	**Informational**	Shadowing local.
DC	**Informational**	Dead code.
LLC	**Informational**	Low level calls.
NC	**Informational**	Naming convention.
SN	**Informational**	Similar name.
TMD	**Informational**	Too many digits.
US	**Informational**	Unused state variable.

CS	**Informational**	State variable that can be declared as constant.

SWC-101 — Compiler-rewritable "<uint> - 1" discovered

SEVERITY	Unknown
LOCATION(S)	ShibConnect.sol#L219, 2064, 2070, 2076

```
210     function remove(Map storage map, address key) internal {
211         if (!map.inserted[key]) {
212             return;
213         }
214
215         delete map.inserted[key];
216         delete map.values[key];
217
218         uint256 index = map.indexOf[key];
219         uint256 lastIndex = map.keys.length - 1;
220         address lastKey = map.keys[lastIndex];
221
222         map.indexOf[lastKey] = index;
223         delete map.indexOf[key];
224
225         map.keys[index] = lastKey;
226         map.keys.pop();
227     }
228
```

```
2058  function updateReferralLeaderboards() private {
2059      // check if the daily/weekly/monthly leaderboards should be reset
2060
2061      if(block.timestamp >= dailyTimer){
2062          iterationDaily++;
2063          dailyTimer = block.timestamp + 8600;
2064          emit LeaderboardCompletion(1, iterationDaily - 1);
2065      }
2066
2067      if(block.timestamp >= weeklyTimer){
2068          iterationWeekly++;
2069          weeklyTimer = block.timestamp + 604800;
2070          emit LeaderboardCompletion(2, iterationWeekly - 1);
2071      }
2072
2073      if(block.timestamp >= monthlyTimer){
2074          iterationMonthly++;
2075          monthlyTimer = block.timestamp + 2629743;
2076          emit LeaderboardCompletion(3, iterationMonthly - 1);
2077      }
2078  }
```

DESCRIPTION	Compiler-rewritable "<uint> - 1" is discovered within the code, which could potentially result in underflow/negative values.
RECOMMENDATIONS	Solidity 0.8.0 onwards has an underflow check, so project creator doesn't have to worry about any invalid/underflow value and this should not produce any issue. No specific actions needed to be taken by project creator.
STATUS	**N/A**

SWC-110 — Out of bounds array access

SEVERITY	Unknown
LOCATION(S)	ShibConnect.sol#L188, 220, 225, 1852-1853, 1923-1924, 2455, 2551-2552, 2621-2622

```
183  function getKeyAtIndex(Map storage map, uint256 index)
184      internal
185      view
186      returns (address)
187  {
188      return map.keys[index];
189  }
```

```
210  function remove(Map storage map, address key) internal {
211      if (!map.inserted[key]) {
212          return;
213      }
214
215      delete map.inserted[key];
216      delete map.values[key];
217
218      uint256 index = map.indexOf[key];
219      uint256 lastIndex = map.keys.length - 1;
220      address lastKey = map.keys[lastIndex];
221
222      map.indexOf[lastKey] = index;
223      delete map.indexOf[key];
224
225      map.keys[index] = lastKey;
226      map.keys.pop();
227  }
```

```
1842  function swapTokensForEth(uint256 tokenAmount, address payable account) private {
1843      if(tokenAmount <= 0){
1844          return;
1845      }
1846      if(balanceOf(address(this)) < tokenAmount){
1847          tokenAmount = balanceOf(address(this));
1848      }
1849
1850      // generate the uniswap pair path of token -> weth
1851      address[] memory path = new address[](2);
1852      path[0] = address(this);
1853      path[1] = uniswapV2Router.WETH();
1854
1855      _approve(address(this), address(uniswapV2Router), tokenAmount);
1856
1857      // make the swap
1858      uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
1859          tokenAmount,
1860          0, // accept any amount of ETH
1861          path,
1862          account,
1863          block.timestamp
1864      );
1865  }
```

```
1921
1922    address[] memory path = new address[](2);
1923    path[0] = uniswapV2Router.WETH();
1924    path[1] = dividendTracker.getPayoutToken(upcoming);
1925
```

```
2549
2550    address[] memory path = new address[](2);
2551    path[0] = uniswapV2Router.WETH();
2552    path[1] = address(shibConnect);
2553
```

```
2448  while (gasUsed < gas && iterations < numberOfTokenHolders) {
2449      _lastProcessedIndex++;
2450
2451      if (_lastProcessedIndex >= numberOfTokenHolders) {
2452          _lastProcessedIndex = 0;
2453      }
2454
2455      address account = tokenHoldersMap.keys[_lastProcessedIndex];
2456
2457      if (!excludedFromAutoClaim[account]) {
2458          if (processAccount(payable(account), true)) {
2459              claims++;
2460          }
2461      }
2462
2463      iterations++;
2464
2465      uint256 newGasLeft = gasleft();
2466
2467      if (gasLeft > newGasLeft) {
2468          gasUsed = gasUsed.add(gasLeft.sub(newGasLeft));
2469      }
2470
2471      gasLeft = newGasLeft;
2472  }
```

```
2619  //investor wants to be payed out in a custom token
2620  address[] memory path = new address[](2);
2621  path[0] = uniswapV2Router.WETH();
2622  path[1] = tokenAddress;
2623
```

DESCRIPTION	The index access expression can cause an exception in case of use of invalid array index value.
RECOMMENDATIONS	As long as project creator is careful with the index access expression to prevent an exception in case of use of invalid array index value, this should not produce any

	issue. No specific actions needed to be taken by project creator.
STATUS	**N/A**

SWC-115 — Use of "tx.origin" as a part of authorization control

SEVERITY	Minor
LOCATION(S)	ShibConnect.sol#L1510, 1760

```
1498  function processDividendTracker(uint256 gas) external {
1499      (
1500          uint256 iterations,
1501          uint256 claims,
1502          uint256 lastProcessedIndex
1503      ) = dividendTracker.process(gas);
1504      emit ProcessedDividendTracker(
1505          iterations,
1506          claims,
1507          lastProcessedIndex,
1508          false,
1509          gas,
1510          tx.origin
1511      );
1512  }
```

```
1749  try dividendTracker.process(gas) returns (
1750      uint256 iterations,
1751      uint256 claims,
1752      uint256 lastProcessedIndex
1753  ) {
1754      emit ProcessedDividendTracker(
1755          iterations,
1756          claims,
1757          lastProcessedIndex,
1758          true,
1759          gas,
1760          tx.origin
1761      );
1762  } catch { }
```

DESCRIPTION	Using "tx.origin" as a security control can lead to authorization bypass vulnerabilities.
RECOMMENDATIONS	We would recommend project creator to consider using "msg.sender" unless you really know what you are doing. Other than that, no specific actions needed to be taken by project creator.
STATUS	**N/A**

SWC-120 — Potential use of "block.number" as source of randomness

SEVERITY	Minor
LOCATION(S)	ShibConnect.sol#L1241, 1636

```
1237  function enableTrading() external onlyOwner {
1238      require(!tradingEnabled, "trading already enabled");
1239      tradingEnabled = true;
1240      enableConvertingReferralRewards = true;
1241      blockNumEnabled = block.number;
1242      emit TradingEnabled();
1243  }
```

```
1635
1636   if(block.number < blockNumEnabled + earlyBlocks){
1637       devFees = earlyTax;
1638       liquidityFee = 0;
1639       BNBRewardsBuyFee = 0;
1640       BNBRewardsSellFee = 0;
1641   }
1642
```

DESCRIPTION	The environment variable "block.number" looks like it might be used as a source of randomness to trigger a function.
RECOMMENDATIONS	We would recommend project owner to not use any of the environment variables like coinbase, gaslimit, block number and timestamp as sources of randomness since they are predictable and be aware that such usage could introduces a certain level of trust into miners. Keep in

	mind that malicious miner can manipulate the value of those variables and that any attackers could also predetermine the hashes of earlier blocks. However, based on our analysis, there's nothing to be done by project owner since in each of the "block.number" value was used as a means to keep track of time/epoch that relates to the trigger of a specific function.
STATUS	**N/A**

CaL — Loops with multiple calls

SEVERITY	Minor
LOCATION(S)	ShibConnect.sol#L2538-2592, 2594-2654

```
2537
2538  function _reinvestDividendOfUser(address account)
2539      private
2540      returns (uint256)
2541  {
2542      uint256 _withdrawableDividend = withdrawableDividendOf(account);
2543      if (_withdrawableDividend > 0) {
2544          bool success;
2545
2546          withdrawnDividends[account] = withdrawnDividends[account].add(
2547              _withdrawableDividend
2548          );
2549
2550          address[] memory path = new address[](2);
2551          path[0] = uniswapV2Router.WETH();
2552          path[1] = address(shibConnect);
2553
2554          uint256 prevBalance = shibConnect.balanceOf(address(this));
2555
2556          // make the swap
2557          try
2558              uniswapV2Router
2559                  .swapExactETHForTokensSupportingFeeOnTransferTokens{
2560                  value: _withdrawableDividend
2561              }(
2562                  0, // accept any amount of Tokens
2563                  path,
2564                  address(this),
2565                  block.timestamp
2566              )
2567          {
2568              uint256 received = shibConnect.balanceOf(address(this)).sub(
2569                  prevBalance
2570              );
2571              if (received > 0) {
2572                  success = true;
2573                  shibConnect.transfer(account, received);
2574              } else {
2575                  success = false;
2576              }
2577          } catch {
2578              success = false;
2579          }
2580
2581          if (!success) {
2582              withdrawnDividends[account] = withdrawnDividends[account].sub(
2583                  _withdrawableDividend
2584              );
2585              return 0;
2586          }
2587
2588          return _withdrawableDividend;
2589      }
2590
2591      return 0;
2592  }
2593
```

```
2594  function _withdrawDividendOfUser(address payable user)
2595      internal
2596      override
2597      returns (uint256)
2598  {
2599      uint256 _withdrawableDividend = withdrawableDividendOf(user);
2600      if (_withdrawableDividend > 0) {
2601          withdrawnDividends[user] = withdrawnDividends[user].add(
2602              _withdrawableDividend
2603          );
2604
2605          address tokenAddress = payoutToken[user];
2606          bool success;
2607
2608          // if no tokenAddress assume bnb payout
2609          if (
2610              !allowCustomTokens ||
2611              tokenAddress == address(0) ||
2612              !allowTokens[tokenAddress]
2613          ) {
2614              (success, ) = user.call{
2615                  value: _withdrawableDividend,
2616                  gas: 3000
2617              }("");
2618          } else {
2619              //investor wants to be payed out in a custom token
2620              address[] memory path = new address[](2);
2621              path[0] = uniswapV2Router.WETH();
2622              path[1] = tokenAddress;
2623
2624              try
2625                  uniswapV2Router
2626                      .swapExactETHForTokensSupportingFeeOnTransferTokens{
2627                      value: _withdrawableDividend
2628                  }(
2629                      0, // accept any amount of Tokens
2630                      path,
2631                      user,
2632                      block.timestamp
2633                  )
2634              {
2635                  success = true;
2636              } catch {
2637                  success = false;
2638              }
2639          }
2640
2641          if (!success) {
2642              withdrawnDividends[user] = withdrawnDividends[user].sub(
2643                  _withdrawableDividend
2644              );
2645              return 0;
2646          } else {
2647              emit DividendWithdrawn(user, _withdrawableDividend);
2648          }
2649
2650          return _withdrawableDividend;
2651      }
2652
2653      return 0;
2654  }
```

DESCRIPTION	[ShibConnectDividendTracker._reinvestDividendOfUser] (#L2538-2592) has external calls inside a loop at line #L2551, 2554, 2557-2579 [ShibConnectDividendTracker._withdrawDividendOfUser] (#L2594-2654) has external calls inside a loop at line #L2614-2617, 2621, 2624-2638
RECOMMENDATIONS	Project creator can choose to either make use of pull over push strategy for external calls or ignore the issues since the logic does require such function(s).
STATUS	**N/A**

UL — Uninitialized local variables

SEVERITY	Informational — Medium
LOCATION(S)	ShibConnect.sol#L1750 – 1752, 1887, 2544, 2606

```
1748
1749  try dividendTracker.process(gas) returns (
1750      uint256 iterations,
1751      uint256 claims,
1752      uint256 lastProcessedIndex
1753  ) {
1754      emit ProcessedDividendTracker(
1755          iterations,
1756          claims,
1757          lastProcessedIndex,
1758          true,
1759          gas,
1760          tx.origin
1761      );
1762  } catch { }
1763
```

```
1885
1886    uint256 initialBalance;
1887    uint256 newBalance;
1888
```

```
2543  if (_withdrawableDividend > 0) {
2544      bool success;
2545
2546      withdrawnDividends[account] = withdrawnDividends[account].add(
2547          _withdrawableDividend
2548      );
2549
2550      address[] memory path = new address[](2);
2551      path[0] = uniswapV2Router.WETH();
2552      path[1] = address(shibConnect);
2553
2554      uint256 prevBalance = shibConnect.balanceOf(address(this));
2555
2556      // make the swap
2557      try
2558          uniswapV2Router
2559              .swapExactETHForTokensSupportingFeeOnTransferTokens{
2560              value: _withdrawableDividend
2561          }(
2562              0, // accept any amount of Tokens
2563              path,
2564              address(this),
2565              block.timestamp
2566          )
2567      {
2568          uint256 received = shibConnect.balanceOf(address(this)).sub(
2569              prevBalance
2570          );
2571          if (received > 0) {
2572              success = true;
2573              shibConnect.transfer(account, received);
2574          } else {
2575              success = false;
2576          }
2577      } catch {
2578          success = false;
2579      }
2580
2581      if (!success) {
2582          withdrawnDividends[account] = withdrawnDividends[account].sub(
2583              _withdrawableDividend
2584          );
2585          return 0;
2586      }
2587
2588      return _withdrawableDividend;
2589  }
2590
```

```
2604
2605  address tokenAddress = payoutToken[user];
2606  bool success;
2607
```

DESCRIPTION	[ShibConnect._transfer().iterations] (#L1750) is a local variable that is never initialized. [ShibConnect._transfer().claims] (#L1751) is a local variable that is never initialized. [ShibConnect._transfer().lastProcessedIndex] (#L1752) is a local variable that is never initialized. [ShibConnect.swapTokensForPayoutToken().newBalance] (#L1887) is a local variable that is never initialized. [ShibConnectDividendTracker._reinvestDividendOfUser().success] (#L2544) is a local variable that is never initialized. [ShibConnectDividendTracker._withdrawDividendOfUser().success] (#L2606) is a local variable that is never initialized.
RECOMMENDATIONS	Project creator is recommended to initialize all the variables even if a variable is meant to be initialized to zero, it is better to explicitly set it to zero for the purpose of improving the readability of the code.
STATUS	**N/A**

UR — Unused return

SEVERITY	Informational — Medium
LOCATION(S)	ShibConnect.sol#L1193, 1515, 1749-1762, 2111-2118

```
1186
1187  // exclude from receiving dividends
1188  dividendTracker.excludeFromDividends(address(dividendTracker));
1189  dividendTracker.excludeFromDividends(address(this));
1190  dividendTracker.excludeFromDividends(
1191      0x000000000000000000000000000000000000dEaD
1192  );
1193  dividendTracker.excludedFromDividends(address(0));
1194  dividendTracker.excludeFromDividends(owner());
1195  dividendTracker.excludeFromDividends(address(uniswapV2Router));
1196
```

```
1513
1514  function claim() external {
1515      dividendTracker.processAccount(payable(msg.sender), false);
1516  }
1517
```

```
1748
1749    try dividendTracker.process(gas) returns (
1750        uint256 iterations,
1751        uint256 claims,
1752        uint256 lastProcessedIndex
1753    ) {
1754        emit ProcessedDividendTracker(
1755            iterations,
1756            claims,
1757            lastProcessedIndex,
1758            true,
1759            gas,
1760            tx.origin
1761        );
1762    } catch { }
1763
```

```
2105
2106  function addLiquidity(uint256 tokenAmount, uint256 ethAmount) private {
2107      // approve token transfer to cover all possible scenarios
2108      _approve(address(this), address(uniswapV2Router), tokenAmount);
2109
2110      // add the liquidity
2111      uniswapV2Router.addLiquidityETH{value: ethAmount}(
2112          address(this),
2113          tokenAmount,
2114          0, // slippage is unavoidable
2115          0, // slippage is unavoidable
2116          liquidityWallet,
2117          block.timestamp
2118      );
2119  }
2120
```

DESCRIPTION	[ShibConnect.constructor()] (#L1169-1217) ignores return value by [dividendTracker.excludedFromDividends()] (#1193) [ShibConnect.claim()] (#L1514-1516) ignores return value by [dividendTracker.processAccount()] (#1515)

	[ShibConnect._transfer()] (#L1584-1779) ignores return value by [dividendTracker.process()] (#L1749-1762) [ShibConnect.addLiquidity()] (#L2106-2119) ignores return value by [uniswapV2Router.addLiquidityETH()] (#L2111-2118)
RECOMMENDATIONS	We recommend project creator to ensure that all the return values of the function calls within the smart contract are used.
STATUS	**N/A**

SL — Shadowing local

SEVERITY	Informational — Minor
LOCATION(S)	ShibConnect.sol#L282-283, 877

```
281
282   string private _name;
283   string private _symbol;
284
```

```
876
877   constructor(string memory _name, string memory _symbol)
878       ERC20(_name, _symbol)
879   {}
880
```

DESCRIPTION	[DividendPayingToken.constructor()._name] (#L877) shadows [ERC20._name] (#L282) [DividendPayingToken.constructor()._symbol] (#L877) shadows [ERC20._symbol] (#L283)
RECOMMENDATIONS	Based on our analysis, this is a direct use of Zeppelin modules. Hence, project owner is not required to actually rename the local variables that shadow another component.
STATUS	**N/A**

DC —Dead code

SEVERITY	Informational — Medium
LOCATION(S)	ShibConnect.sol#L14-17, 168-170, 549-551, 554-561, 606-612, 614-619, 634-637, 904-931, 969-984

```
14  function _msgData() internal view virtual returns (bytes memory) {
15      this;
16      return msg.data;
17  }
```

```
168  function get(Map storage map, address key) internal view returns (uint256) {
169      return map.values[key];
170  }
```

```
549   function mod(uint256 a, uint256 b) internal pure returns (uint256) {
550       return mod(a, b, "SafeMath: modulo by zero");
551   }
```

```
606  function mul(int256 a, int256 b) internal pure returns (int256) {
607      int256 c = a * b;
608
609      require(c != MIN_INT256 || (a & MIN_INT256) != (b & MIN_INT256));
610      require((b == 0) || (c / b == a));
611      return c;
612  }
```

```
554   function mod(
555       uint256 a,
556       uint256 b,
557       string memory errorMessage
558   ) internal pure returns (uint256) {
559       require(b != 0, errorMessage);
560       return a % b;
561   }
```

```
614   function div(int256 a, int256 b) internal pure returns (int256) {
615       // Prevent overflow when dividing MIN_INT256 by -1
616       require(b != -1 || a != MIN_INT256);
617
618       return a / b;
619   }
```

```
634   function abs(int256 a) internal pure returns (int256) {
635       require(a != MIN_INT256);
636       return a < 0 ? -a : a;
637   }
```

```
904  function _withdrawDividendOfUser(address payable user)
905      internal
906      virtual
907      returns (uint256)
908  {
909      uint256 _withdrawableDividend = withdrawableDividendOf(user);
910      if (_withdrawableDividend > 0) {
911          withdrawnDividends[user] = withdrawnDividends[user].add(
912              _withdrawableDividend
913          );
914          emit DividendWithdrawn(user, _withdrawableDividend);
915          (bool success, ) = user.call{
916              value: _withdrawableDividend,
917              gas: 3000
918          }("");
919
920          if (!success) {
921              withdrawnDividends[user] = withdrawnDividends[user].sub(
922                  _withdrawableDividend
923              );
924              return 0;
925          }
926
927          return _withdrawableDividend;
928      }
929
930      return 0;
931  }
```

```
969  function _transfer(
970      address from,
971      address to,
972      uint256 value
973  ) internal virtual override {
974      require(false);
975
976      int256 _magCorrection = magnifiedDividendPerShare
977          .mul(value)
978          .toInt256Safe();
979      magnifiedDividendCorrections[from] = magnifiedDividendCorrections[from]
980          .add(_magCorrection);
981      magnifiedDividendCorrections[to] = magnifiedDividendCorrections[to].sub(
982          _magCorrection
983      );
984  }
```

DESCRIPTION	[Context._msgData()] (#L14-17) is never used and should be removed. [IterableMapping.get()] (#L168-170) is never used and should be removed.

	[SafeMath.mod()] (#L549-551) is never used and should be removed. [SafeMathInt.mod()] (#L554-561) is never used and should be removed. [SafeMathInt.mul()] (#L606-612) is never used and should be removed. [SafeMathInt.div()] (#L614-619) is never used and should be removed. [SafeMathInt.abs()] (#L634-637) is never used and should be removed. [DividendPayingToken._withdrawDividendOfUser()] (#L904-931) is never used and should be removed. [DividendPayingToken._transfer()] (#L969-984) is never used and should be removed.
RECOMMENDATIONS	Based on our analysis, the Address, Context and SafeMath smart contracts is the standard that is a direct fork from Open Zeppelin while IterableMapping and SafeMathInt smart contract is a derivative from the standard used by many and the DividendPayingToken is a custom contract made by project owner. While other functions within these smart contracts were used in some part throughout the smart contract, these functions are not being used anywhere at all. It is recommended for project creator to remove those functions to further optimize the smart contract since they are not used anywhere at all. Doing so will reduce the amount of gas required when deploying the smart contract.
STATUS	**N/A**

LLC — Low level calls

SEVERITY	Informational — Medium
LOCATION(S)	ShibConnect.sol#L915-918, 2151, 2165, 2614-2617

```
904  function _withdrawDividendOfUser(address payable user)
905      internal
906      virtual
907      returns (uint256)
908  {
909      uint256 _withdrawableDividend = withdrawableDividendOf(user);
910      if (_withdrawableDividend > 0) {
911          withdrawnDividends[user] = withdrawnDividends[user].add(
912              _withdrawableDividend
913          );
914          emit DividendWithdrawn(user, _withdrawableDividend);
915          (bool success, ) = user.call{
916              value: _withdrawableDividend,
917              gas: 3000
918          }("");
919
920          if (!success) {
921              withdrawnDividends[user] = withdrawnDividends[user].sub(
922                  _withdrawableDividend
923              );
924              return 0;
925          }
926
927          return _withdrawableDividend;
928      }
929
930      return 0;
931  }
```

```
2150
2151  (bool success, ) = address(dividendTracker).call{value: BNBRewardsFeeBNB}("");
2152
```

```
2164
2165  (bool successMarketing, ) = address(marketingAddress).call{value: devFeesBNB}("");
2166
```

```
2614                    (success, ) = user.call{
2615                        value: _withdrawableDividend,
2616                        gas: 3000
2617                    }("");
2618
```

DESCRIPTION	[DividendPayingToken._withdrawDividendOfUser()] (#L904-931) is using low level call at #L915-918. [ShibConnect.swapAndSendDividendsAndMarketingFunds ()] (#L2126-2172) is using low level call at #L2151 and #L2165. [ShibConnectDividendTracker._withdrawDividendOfUser()] (#L2594-2654) is using low level call at #L2614-2617.
RECOMMENDATIONS	Project creator should avoid using low-level calls. Make sure to check the call success or for code existence if the call is meant for a contract. The use of low-level calls is usually error-prone since they do not check for code existence or call success.
STATUS	**N/A**

NC — Naming convention

SEVERITY	Informational — Minor
LOCATION(S)	ShibConnect.sol#L53, 55, 86, 656, 868, 933, 937, 946, 955, 1040-1043, 1372-1373, 1393-1394, 1575-1576, 1981, 2332

```
52
53  function DOMAIN_SEPARATOR() external view returns (bytes32);
54
55  function PERMIT_TYPEHASH() external pure returns (bytes32);
56
```

```
85
86  function MINIMUM_LIQUIDITY() external pure returns (uint256);
87
```

```
655
656  function WETH() external pure returns (address);
657
```

```
867
868  uint256 internal constant magnitude = 2**128;
869
```

```
932
933  function dividendOf(address _owner) public view override returns (uint256) {
934      return withdrawableDividendOf(_owner);
935  }
936
937  function withdrawableDividendOf(address _owner)
938      public
939      view
940      override
941      returns (uint256)
942  {
943      return accumulativeDividendOf(_owner).sub(withdrawnDividends[_owner]);
944  }
945
946  function withdrawnDividendOf(address _owner)
947      public
948      view
949      override
950      returns (uint256)
951  {
952      return withdrawnDividends[_owner];
953  }
954
```

```
954
955   function accumulativeDividendOf(address _owner)
956       public
957       view
958       override
959       returns (uint256)
960   {
961       return
962           magnifiedDividendPerShare
963               .mul(balanceOf(_owner))
964               .toInt256Safe()
965               .add(magnifiedDividendCorrections[_owner])
966               .toUint256Safe() / magnitude;
967   }
968
```

```
1040   uint256 public BNBRewardsBuyFee = 0;
1041   uint256 public BNBRewardsBuyFeeReferred = 0;
1042   uint256 public BNBRewardsSellFee = 0;
1043   uint256 public BNBRewardsSellFeeReferred = 0;
```

```
1368
1369  function updateFees(
1370      uint256 dev,
1371      uint256 liquidity,
1372      uint256 BNBRewardsBuy,
1373      uint256 BNBRewardsSell,
1374      uint256 referral
1375  ) public onlyOwner {
1376      require(referral <= 5,"Cannot set referral fee over 5%");
1377      require(dev <= 5,"Cannot set dev fee over 5%");
1378      require(BNBRewardsBuy <= 2,"Cannot set BNBreward fee over 2%");
1379      require(BNBRewardsSell <= 2,"Cannot set BNBreward fee over 2%");
1380      require(liquidity <= 5,"Cannot set liquidity fee over 5%");
1381      devFees = dev;
1382      liquidityFee = liquidity;
1383      BNBRewardsBuyFee = BNBRewardsBuy;
1384      BNBRewardsSellFee = BNBRewardsSell;
1385      referralFee = referral;
1386
1387      emit UpdateFees(dev, liquidity, BNBRewardsBuy, BNBRewardsSell, referralFee);
1388  }
1389
```

```
1389
1390  function updateFeesReferred(
1391      uint256 devReferred,
1392      uint256 liquidityReferred,
1393      uint256 BNBRewardsBuyReferred,
1394      uint256 BNBRewardsSellReferred
1395  ) public onlyOwner {
1396      require(BNBRewardsBuyReferred <= 2,"Cannot set BNBreward fee over 2%");
1397      require(BNBRewardsSellReferred <= 2,"Cannot set BNBreward fee over 2%");
1398      require(devReferred <= 10,"Cannot set dev fee over 10%");
1399      require(liquidityReferred <= 10,"Cannot set liquidity fee over 10%");
1400      devFeesReferred = devReferred;
1401      liquidityFeeReferred = liquidityReferred;
1402      BNBRewardsBuyFeeReferred = BNBRewardsBuyReferred;
1403      BNBRewardsSellFeeReferred = BNBRewardsSellReferred;
1404
1405      emit UpdateFeesReferred(devReferred, liquidityReferred, BNBRewardsBuyReferred, BNBRewardsSellReferred);
1406  }
1407
```

```
1575  uint256 private BNBRewardsBuyFeeActual;
1576  uint256 private BNBRewardsSellFeeActual;
```

```
1980
1981  function setReferrer(address _referrer) public {
1982      require(_referrer != address(0),"Not a valid referrer");
1983      require(_referrer != msg.sender, "You cannot refer yourself");
1984      require(referrerTree[msg.sender][0] == address(0), "Referrer cannot be changed!");
1985
1986      // add the direct referrer to the user's payout tree
1987      referrerTree[msg.sender][0] = _referrer;
1988      referralCount[_referrer] = referralCount[_referrer] + 1;
1989
1990      // check if the referrer was referred through a tree of their own;
1991      // set payout tree accordingly if so
1992      for(int i = 0; i < referralTreeFeesLength - 1; i++){
1993          address treeAddress = referrerTree[_referrer][i];
1994
1995          if(treeAddress == address(0x0000000000000000000000000000000000000000)){
1996              break;
1997          }
1998
1999          referrerTree[msg.sender][i + 1] = treeAddress;
2000          referralCountBranched[treeAddress] = referralCountBranched[treeAddress] + 1;
2001      }
2002
2003      emit ReferredBy(_referrer, msg.sender, iterationDaily, iterationWeekly, iterationMonthly);
2004  }
2005
```

```
2331
2332  function getAccount(address _account)
2333      public
2334      view
2335      returns (
2336          address account,
2337          int256 index,
2338          int256 iterationsUntilProcessed,
2339          uint256 withdrawableDividends,
2340          uint256 totalDividends,
2341          uint256 lastClaimTime
2342      )
2343  {
2344      account = _account;
2345
2346      index = tokenHoldersMap.getIndexOfKey(account);
2347
2348      iterationsUntilProcessed = -1;
2349
2350      if (index >= 0) {
2351          if (uint256(index) > lastProcessedIndex) {
2352              iterationsUntilProcessed = index.sub(
2353                  int256(lastProcessedIndex)
2354              );
2355          } else {
2356              uint256 processesUntilEndOfArray = tokenHoldersMap.keys.length >
2357                  lastProcessedIndex
2358                  ? tokenHoldersMap.keys.length.sub(lastProcessedIndex)
2359                  : 0;
2360
2361              iterationsUntilProcessed = index.add(
2362                  int256(processesUntilEndOfArray)
2363              );
2364          }
2365      }
2366
2367      withdrawableDividends = withdrawableDividendOf(account);
2368      totalDividends = accumulativeDividendOf(account);
2369
2370      lastClaimTime = lastClaimTimes[account];
2371  }
2372
```

DESCRIPTION	[IUniswapV2Pair.DOMAIN_SEPARATOR] (#L53) is not in mixedCase. [IUniswapV2Pair.PERMIT_TYPEHASH] (#L55) is not in mixedCase. [IUniswapV2Pair.MINIMUM_LIQUIDITY] (#L86) is not in mixedCase. [IUniswapV2Router01.WETH] (#L656) is not in mixedCase. [DividendPayingToken.dividendOf()._owner] (#L933) is not in mixedCase. [DividendPayingToken.withdrawableDividendOf()._owner] (#L937) is not in mixedCase. [DividendPayingToken.withdrawnDividendOf()._owner] (#L946) is not in mixedCase. [DividendPayingToken.accumulativeDividendOf()._owner] (#L955) is not in mixedCase. [ShibConnect.BNBRewardsBuyFee] (#L1040) is not in mixedCase. [ShibConnect.BNBRewardsBuyFeeReferred] (#L1041) is not in mixedCase. [ShibConnect.BNBRewardsSellFee] (#L1042) is not in mixedCase. [ShibConnect.BNBRewardsSellFeeReferred] (#L1043) is not in mixedCase. [ShibConnect.updateFees().BNBRewardsBuy] (#L1372) is not in mixedCase. [ShibConnect.updateFees().BNBRewardsSell] (#L1373) is not in mixedCase. [ShibConnect.updateFeesReferred().BNBRewardsBuyReferred] (#L1393) is not in mixedCase. [ShibConnect.updateFeesReferred().BNBRewardsSellReferred] (#L1394) is not in mixedCase. [ShibConnect.BNBRewardsBuyFeeActual] (#L1575) is not in mixedCase.

	[ShibConnect. BNBRewardsSellFeeActual] (#L1576) is not in mixedCase. [ShibConnect.setReferrer()._referrer] (#L1981) is not in mixedCase. [ShibConnectDividendTracker.getAccount()._account] (#L2332) is not in mixedCase. [DividendPayingToken.magnitude] (#L868) is not in UPPER_CASE_WITH_UNDERSCORES.
RECOMMENDATIONS	Based on our analysis, the IUniswapV2Pair and IUniswapV2Router01 smart contract are direct forks from Uniswap. Although the name doesn't conform to the standard convention, it's still okay to leave it be to avoid from potentially breaking any external function. However, for ShibConnect and DividendPayingToken smart contracts, it is okay for project creator to update the name of the parameters in those functions so that they conform to the standard naming convention.
STATUS	**N/A**

SN — Similar name

SEVERITY	Informational — Minor
LOCATION(S)	ShibConnect.sol#L661-662, 909, 2339, 2542, 2599

```
658  function addLiquidity(
659      address tokenA,
660      address tokenB,
661      uint256 amountADesired,
662      uint256 amountBDesired,
663      uint256 amountAMin,
664      uint256 amountBMin,
665      address to,
666      uint256 deadline
667  )
668      external
669      returns (
670          uint256 amountA,
671          uint256 amountB,
672          uint256 liquidity
673      );
```

```
2537
2538  function _reinvestDividendOfUser(address account)
2539      private
2540      returns (uint256)
2541  {
2542      uint256 _withdrawableDividend = withdrawableDividendOf(account);
2543      if (_withdrawableDividend > 0) {
2544          bool success;
2545
2546          withdrawnDividends[account] = withdrawnDividends[account].add(
2547              _withdrawableDividend
2548          );
2549
2550          address[] memory path = new address[](2);
2551          path[0] = uniswapV2Router.WETH();
2552          path[1] = address(shibConnect);
2553
2554          uint256 prevBalance = shibConnect.balanceOf(address(this));
2555
2556          // make the swap
2557          try
2558              uniswapV2Router
2559                  .swapExactETHForTokensSupportingFeeOnTransferTokens{
2560                  value: _withdrawableDividend
2561              }(
2562                  0, // accept any amount of Tokens
2563                  path,
2564                  address(this),
2565                  block.timestamp
2566              )
2567          {
2568              uint256 received = shibConnect.balanceOf(address(this)).sub(
2569                  prevBalance
2570              );
2571              if (received > 0) {
2572                  success = true;
2573                  shibConnect.transfer(account, received);
2574              } else {
2575                  success = false;
2576              }
2577          } catch {
2578              success = false;
2579          }
2580
2581          if (!success) {
2582              withdrawnDividends[account] = withdrawnDividends[account].sub(
2583                  _withdrawableDividend
2584              );
2585              return 0;
2586          }
2587
2588          return _withdrawableDividend;
2589      }
2590
2591      return 0;
2592  }
2593
```

```
2331
2332  function getAccount(address _account)
2333      public
2334      view
2335      returns (
2336          address account,
2337          int256 index,
2338          int256 iterationsUntilProcessed,
2339          uint256 withdrawableDividends,
2340          uint256 totalDividends,
2341          uint256 lastClaimTime
2342      )
2343  {
2344      account = _account;
2345
2346      index = tokenHoldersMap.getIndexOfKey(account);
2347
2348      iterationsUntilProcessed = -1;
2349
2350      if (index >= 0) {
2351          if (uint256(index) > lastProcessedIndex) {
2352              iterationsUntilProcessed = index.sub(
2353                  int256(lastProcessedIndex)
2354              );
2355          } else {
2356              uint256 processesUntilEndOfArray = tokenHoldersMap.keys.length >
2357                  lastProcessedIndex
2358                  ? tokenHoldersMap.keys.length.sub(lastProcessedIndex)
2359                  : 0;
2360
2361              iterationsUntilProcessed = index.add(
2362                  int256(processesUntilEndOfArray)
2363              );
2364          }
2365      }
2366
2367      withdrawableDividends = withdrawableDividendOf(account);
2368      totalDividends = accumulativeDividendOf(account);
2369
2370      lastClaimTime = lastClaimTimes[account];
2371  }
2372
```

```
903
904  function _withdrawDividendOfUser(address payable user)
905      internal
906      virtual
907      returns (uint256)
908  {
909      uint256 _withdrawableDividend = withdrawableDividendOf(user);
910      if (_withdrawableDividend > 0) {
911          withdrawnDividends[user] = withdrawnDividends[user].add(
912              _withdrawableDividend
913          );
914          emit DividendWithdrawn(user, _withdrawableDividend);
915          (bool success, ) = user.call{
916              value: _withdrawableDividend,
917              gas: 3000
918          }("");
919
920          if (!success) {
921              withdrawnDividends[user] = withdrawnDividends[user].sub(
922                  _withdrawableDividend
923              );
924              return 0;
925          }
926
927          return _withdrawableDividend;
928      }
929
930      return 0;
931  }
932
```

```
2594  function _withdrawDividendOfUser(address payable user)
2595      internal
2596      override
2597      returns (uint256)
2598  {
2599      uint256 _withdrawableDividend = withdrawableDividendOf(user);
2600      if (_withdrawableDividend > 0) {
2601          withdrawnDividends[user] = withdrawnDividends[user].add(
2602              _withdrawableDividend
2603          );
2604
2605          address tokenAddress = payoutToken[user];
2606          bool success;
2607
2608          // if no tokenAddress assume bnb payout
2609          if (
2610              !allowCustomTokens ||
2611              tokenAddress == address(0) ||
2612              !allowTokens[tokenAddress]
2613          ) {
2614              (success, ) = user.call{
2615                  value: _withdrawableDividend,
2616                  gas: 3000
2617              }("");
2618          } else {
2619              //investor wants to be payed out in a custom token
2620              address[] memory path = new address[](2);
2621              path[0] = uniswapV2Router.WETH();
2622              path[1] = tokenAddress;
2623
2624              try
2625                  uniswapV2Router
2626                      .swapExactETHForTokensSupportingFeeOnTransferTokens{
2627                      value: _withdrawableDividend
2628                  }(
2629                      0, // accept any amount of Tokens
2630                      path,
2631                      user,
2632                      block.timestamp
2633                  )
2634              {
2635                  success = true;
2636              } catch {
2637                  success = false;
2638              }
2639          }
2640
2641          if (!success) {
2642              withdrawnDividends[user] = withdrawnDividends[user].sub(
2643                  _withdrawableDividend
2644              );
2645              return 0;
2646          } else {
2647              emit DividendWithdrawn(user, _withdrawableDividend);
2648          }
2649
2650          return _withdrawableDividend;
2651      }
2652
2653      return 0;
2654  }
```

DESCRIPTION	[IUniswapV2Router01.addLiquidity()] (#L661-662) has two parameters names that are too similar. [DividendPayingToken._withdrawDividendOfUser()._withdrawableDividend] (#L909) is too similar with [ShibConnectDividendTracker.getAccount().withdrawableDividends] (#L2339). [ShibConnectDividendTracker._reinvestDividendOfUser().withdrawableDividend] (#L2542) is too similar with [ShibConnectDividendTracker.getAccount().withdrawableDividends] (#L2339). [ShibConnectDividendTracker._withdrawDividendOfUser()._withdrawableDividend] (#L2599) is too similar with [ShibConnectDividendTracker.getAccount().withdrawableDividends] (#L2339).
RECOMMENDATIONS	Based on our analysis, the IUniswapV2Router01 smart contract is a direct fork from Uniswap and it is just an interface. Although their names are too similar, it's still okay to leave them be for the purpose of following the standard parameter declaration that is widely used as reference. However, for DividendPayingToken and ShibConnectDividendTracker smart contract, it is recommended for project owner to make the changes to the name and prevent variables from having similar names.
STATUS	**N/A**

TMD — Too many digits

SEVERITY	Informational — Medium
LOCATION(S)	ShibConnect.sol#L1032-1034, 1080, 1216, 1364, 1832, 1838

```
1032  uint256 public maxSellTransactionAmount = 1000000000 * (10**9); // No max sell
1033  uint256 public swapTokensAtAmount = 200000 * (10**9);
1034  uint256 public swapTokensAtAmountMax = 5000000 * (10**9);
```

```
1080  uint256 public gasForProcessing = 300000;
```

```
1362
1363  function updateGasForProcessing(uint256 newValue) public onlyOwner {
1364      require(newValue >= 200000 && newValue <= 500000, "new gas value must be between 200000 and 500000");
1365      emit GasForProcessingUpdated(newValue, gasForProcessing);
1366      gasForProcessing = newValue;
1367  }
1368
```

```
1830
1831  function setSwapTokensAmount(uint256 amount) public onlyOwner {
1832      require (amount <= 20000000, "cannot set swap amount greater than .2%");
1833      swapTokensAtAmount = amount;
1834  }
1835
1836  function setSwapTokensAmountMax(uint256 amount) public onlyOwner {
1837      require(amount > swapTokensAtAmount, "Max amount must be greater than minimum");
1838      require (amount <= 20000000, "cannot set swap amount greater than .2%");
1839      swapTokensAtAmountMax = amount;
1840  }
1841
```

```
1168
1169  constructor() ERC20("ShibConnect", "ShibConnect") {
1170      dividendTracker = new ShibConnectDividendTracker(payable(this));
1171
1172      liquidityWallet = owner();
1173
1174      uniswapV2Router = IUniswapV2Router02(
1175          // 0x10ED43C718714eb63d5aA57B78B54704E256024E //mainnet
1176          0xD99D1c33F9fC3444f8101754aBC46c52416550D1 //testnet
1177      );
1178
1179      // Create a uniswap pair for this new token
1180      uniswapV2Pair = IUniswapV2Factory(uniswapV2Router.factory()).createPair(
1181              address(this),
1182              uniswapV2Router.WETH()
1183          );
1184
1185      _setAutomatedMarketMakerPair(uniswapV2Pair, true);
1186
1187      // exclude from receiving dividends
1188      dividendTracker.excludeFromDividends(address(dividendTracker));
1189      dividendTracker.excludeFromDividends(address(this));
1190      dividendTracker.excludeFromDividends(
1191          0x000000000000000000000000000000000000dEaD
1192      );
1193      dividendTracker.excludedFromDividends(address(0));
1194      dividendTracker.excludeFromDividends(owner());
1195      dividendTracker.excludeFromDividends(address(uniswapV2Router));
1196
1197      // exclude from paying fees or having max transaction amount
1198      _isExcludedFromFees[liquidityWallet] = true;
1199      _isExcludedFromFees[address(this)] = true;
1200      _isExcludedFromFees[owner()] = true;
1201      _isExcludedFromFees[address(dividendTracker)] = true;
1202
1203      referralTreeFees[0] = 300; // 3% to primary referrer
1204      referralTreeFees[1] = 60; // 0.6% to secondary referrer
1205      referralTreeFees[2] = 40; // 0.4% to tertiary referrer
1206      referralTreeFeesLength = 3;
1207
1208      calculateReferralFee();
1209
1210      canTransferBeforeTradingIsEnabled[owner()] = true;
1211      /*
1212          _mint is an internal function in ERC20.sol that is only called here,
1213          and CANNOT be called ever again
1214      */
1215
1216      _mint(owner(), 1000000000 * (10**9));
1217  }
1218
```

DESCRIPTION	[ShibConnect.maxSellTransactionAmount] (#L1032) uses literals with too many digits. [ShibConnect.swapTokensAtAmount] (#L1033) uses literals with too many digits. [ShibConnect.swapTokensAtAmountMax] (#L1034) uses literals with too many digits. [ShibConnect.gasForProcessing] (#L1080) uses literals with too many digits.

	[ShibConnect.constructor()] (#L1216) uses literals with too many digits. [ShibConnect.updateGasForProcessing()] (#L1364) uses literals with too many digits. [ShibConnect.setSwapTokensAmount()] (#L1832) uses literals with too many digits. [ShibConnect.setSwapTokensAmount()] (#L1838) uses literals with too many digits.
RECOMMENDATIONS	Literals that use too many digits are usually difficult to read and review, which makes them likely to be used incorrectly. Project creator can try to use ether suffix.
STATUS	**N/A**

US — Unused state variable

SEVERITY	Informational — Minor
LOCATION(S)	ShibConnect.sol#L603, 1577-1578

```
603   int256 private constant MAX_INT256 = ~(int256(1) << 255);
```

```
1577   uint256 private totalBuyFeesActual;
1578   uint256 private totalSellFeesActual;
```

DESCRIPTION	[SafeMathInt.MAX_INT256] (#L603) is never used. [ShibConnect.totalBuyFeesActual] (#L1577) is never used. [ShibConnect.totalSellFeesActual] (#L1578) is never used.
RECOMMENDATIONS	Based on our analysis, project creator can either create a function to call the value for preview or completely remove it from the code. Project creator can also opt to keep them as it doesn't give any issue other than increasing the gas fee when deploying the smart contract.
STATUS	**N/A**

CS — State variable that can be declared as constant

SEVERITY	Informational — Minor
LOCATION(S)	ShibConnect.sol#L1032, 1577-1578

```
1032  uint256 public maxSellTransactionAmount = 1000000000 * (10**9); // No max sell
```

```
1577  uint256 private totalBuyFeesActual;
1578  uint256 private totalSellFeesActual;
```

DESCRIPTION	[ShibConnect.maxSellTransactionAmount] (#L1032) should be constant. [ShibConnect.totalBuyFeesActual] (#L1577) should be constant. [ShibConnect.totalSellFeesActual] (#L1578) should be constant.
RECOMMENDATIONS	Based on our analysis, these variables should be declared as constant since they don't change throughout smart contract.
STATUS	**N/A**

Disclaimer

This report only shows findings based on our limited project analysis according to the good industry practice from the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, overall online presence and team transparency details of which are set out in this report. To get a full view of our analysis, **it is important for you to read the full report.** Under no circumstances did Revoluzion Audit receive a payment to manipulate those results or change the awarding badge that we will be adding in our website. **Our team provides no guarantees against the sale of team tokens or the removal of liquidity by the project** audited in this document.

While **we have done our best to conduct thorough analysis to produce this report**, it is crucial to note that you should not rely solely on this report and use the content provided in this document as financial advice or a reason to buy any investment. The Our team disclaims any liability against us for the resulting losses based on the you decision made by relying on the content of this report. **You must conduct your own independent investigations before making any decisions** to protect yourselves from being scammed. We go into more detail on this in the disclaimer clause in the next page — please make sure to read it in full.

Full Disclaimer Clause

By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy all copies of this report downloaded and/or printed by you. This report is provided for information purposes only, on a non-reliance basis and does not constitute to any investment advice.

No one shall have any right to rely on the report or its contents, and Revoluzion Audit and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers, and other representatives) (collectively known as Revoluzion) owe no duty of care towards you or any other person, nor do we make any warranty or representation to any person on the accuracy or completeness of the report.

The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and Revoluzion hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report.

Except and only to the extent that it is prohibited by law, Revoluzion hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against Revoluzion, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report.

The analysis of the security is purely based on the smart contracts, website, social media, and team.