# AudiTBlock

ContractsAudit.com

# AssetFi (ASF)

v0.8.4+commit.c7e474f2

| Low-Risk | Medium-Risk | High-Risk |
| --- | --- | --- |
| low-risk code | medium-risk code | high-risk code |

**Contract Address**

AssetFi Token Deployed On Bscscan.com

0xcd50fcc52a0051757695eef3230244eebb5567d8

# Disclaimer

AudiTBlock is not responsible if a project turns out to be a scam, rug-pull or honeypot. We only provide a detailed analysis for your own research.

AudiTBlock is not responsible for any financial losses. Nothing in this contract audit is financial advice, please do your own research.

The information provided in this audit is for informational purposes only and should not be considered investment advice. Coinsult does not endorse, recommend, support or suggest to invest in any project.

AudiTBlock can not be held responsible for when a project turns out to be a rug-pull, honeypot or scam.

## Tokenomics

- BNB Chain Network

## Source Code

- AudiTBlock was comissioned by ContrcatsAudit to perform an audit based on the following smart contract:

- https://bscscan.com/address/0xcd50fcc52a0051757695eef3230244eebb5567d8#code

# Gas Cost

```
IUniswapV2Router02 public immutable uniswapV2Router;
 address public immutable uniswapV2Pair;
 function name() public view returns (string memory) {
      return _name;
  }
  function transfer(address recipient, uint256 amount)
      public
      override
      returns (bool)
  {
      _transfer(_msgSender(), recipient, amount);
      return true;
  }
function excludeFromReward(address account) public onlyOwner {
      require(!_isExcluded[account], "Account is already excluded");
      if (_rOwned[account] > 0) {
          _tOwned[account] = tokenFromReflection(_rOwned[account]);
      }
      _isExcluded[account] = true;
      _excluded.push(account);
  }

   function setSwapAndLiquifyEnabled(bool _enabled) public
onlyOwner {
       swapAndLiquifyEnabled = _enabled;
       emit SwapAndLiquifyEnabledUpdated(_enabled);
   }
```

Gas requirement of  infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

# Auto

```
function includeInReward(address account) external onlyOwner {
    require(_isExcluded[account], "Account is already included");
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (_excluded[i] == account) {
            _excluded[i] = _excluded[_excluded.length - 1];
            _tOwned[account] = 0;
            _isExcluded[account] = false;
            _excluded.pop();
            break;
        }
    }
}
```

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas.

**Constant/View/Pure**
functions:IUniswapV2Router02.swapExactTokensForETHSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static

**Similar variable names**
AssetFi.deliver(uint256) : Variables have very similar names "_rOwned" and "_tOwned". Note: Modifiers are currently not considered by this static analysis.

**No return**
IUniswapV2Router02.removeLiquidityETHWithPermitSupportingFeeOnTransferTokens(address,uint256,uint256,uint256,address,uint256,bool,uint8,bytes32,bytes32): Defines a return type but never explicitly returns a value.

**Guard conditions**
Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

**Tested Contract Files**

The following are the MD5 hashes of the reviewed files. A file with a different MD5 hash has been modified, intentionally or otherwise,
after the security review. You are cautioned that a different MD5 hash could be (but is not necessarily) an indication of a changed
condition or potential vulnerability that was not within the scope of the review

| File | Fingerprint (MD5 |
| --- | --- |
| Contract/AssetFi.sol | 21c9383b7c65805a9d08fa0f2d20e2c3 |

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

| Dependency / Import Path | Source Sha1 Hash |
| --- | --- |
| Contracts /Context, IERC20, Ownable | 553fbd01798d95ce18de846b295071e7cb77f45d |

## 0.2 SOLIDITY UNIT TESTING

Progress: 2 finished (of 2)
PASS
**Tested (AssetFi.sol)**

✓ Check winning proposal
✓ Check winnin proposal with return value
✓ Before all
✓ Check success
✓ Check success2
✓ Check sender and value

**Result for tests** Passed:

0Time Taken: 0.18s

# 0.3 TESTING

```
Reentrancy in AssetFi._transfer(address,address,uint256) (contracts/audit.sol#1070-1099):
	External calls:
	- swapAndLiquify(contractTokenBalance) (contracts/audit.sol#1092)
		- uniswapV2Router.addLiquidityETH{value:
ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (contracts/audit.sol#1127-1134)
		-
uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),bloc
k.timestamp) (contracts/audit.sol#1116-1122)
	External calls sending eth:
	- swapAndLiquify(contractTokenBalance) (contracts/audit.sol#1092)
		- uniswapV2Router.addLiquidityETH{value:
ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (contracts/audit.sol#1127-1134)
	State variables written after the call(s):
	- _tokenTransfer(from,to,amount,takeFee) (contracts/audit.sol#1098)
		- _rOwned[address(this)] = _rOwned[address(this)].add(rLiquidity) (contracts/audit.sol#1011)
		- _rOwned[_developmentWalletAddress] = _rOwned[_developmentWalletAddress].add(rDevelopment)
(contracts/audit.sol#1019-1020)
		- _rOwned[sender] = _rOwned[sender].sub(rAmount) (contracts/audit.sol#1194)
		- _rOwned[sender] = _rOwned[sender].sub(rAmount) (contracts/audit.sol#1172)
		- _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (contracts/audit.sol#1173)
		- _rOwned[sender] = _rOwned[sender].sub(rAmount) (contracts/audit.sol#881)
		- _rOwned[sender] = _rOwned[sender].sub(rAmount) (contracts/audit.sol#1218)
		- _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (contracts/audit.sol#1219)
		- _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (contracts/audit.sol#1196)
		- _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (contracts/audit.sol#883)
	AssetFi._rOwned (contracts/audit.sol#657) can be used in cross function reentrancies:
	- AssetFi._getCurrentSupply() (contracts/audit.sol#993-1006)
	- AssetFi._takeDevelopment(uint256) (contracts/audit.sol#1016-1025)
	- AssetFi._takeLiquidity(uint256) (contracts/audit.sol#1008-1014)
	- AssetFi._transferBothExcluded(address,address,uint256) (contracts/audit.sol#866-888)
	- AssetFi._transferFromExcluded(address,address,uint256) (contracts/audit.sol#1203-1224)
	- AssetFi._transferStandard(address,address,uint256) (contracts/audit.sol#1158-1178)
	- AssetFi._transferToExcluded(address,address,uint256) (contracts/audit.sol#1180-1201)
	- AssetFi.balanceOf(address) (contracts/audit.sol#727-730)
	- AssetFi.constructor() (contracts/audit.sol#698-709)
	- AssetFi.deliver(uint256) (contracts/audit.sol#807-817)
	- AssetFi.excludeFromReward(address) (contracts/audit.sol#844-851)
	- _tokenTransfer(from,to,amount,takeFee) (contracts/audit.sol#1098)
		- _rTotal = _rTotal.sub(rFee) (contracts/audit.sol#924)
	AssetFi._rTotal (contracts/audit.sol#667) can be used in cross function reentrancies:
	- AssetFi._getCurrentSupply() (contracts/audit.sol#993-1006)
	- AssetFi._reflectFee(uint256,uint256) (contracts/audit.sol#923-926)
	- AssetFi.constructor() (contracts/audit.sol#698-709)
	- AssetFi.deliver(uint256) (contracts/audit.sol#807-817)
	- AssetFi.tokenFromReflection(uint256) (contracts/audit.sol#833-842)
	- _tokenTransfer(from,to,amount,takeFee) (contracts/audit.sol#1098)
		- _tOwned[address(this)] = _tOwned[address(this)].add(tLiquidity) (contracts/audit.sol#1013)
		- _tOwned[_developmentWalletAddress] = _tOwned[_developmentWalletAddress].add(tDevelopment)
(contracts/audit.sol#1022-1024)
		- _tOwned[sender] = _tOwned[sender].sub(tAmount) (contracts/audit.sol#880)
		- _tOwned[sender] = _tOwned[sender].sub(tAmount) (contracts/audit.sol#1217)
		- _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount) (contracts/audit.sol#1195)
		- _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount) (contracts/audit.sol#882)
	AssetFi._tOwned (contracts/audit.sol#658) can be used in cross function reentrancies:
	- AssetFi._getCurrentSupply() (contracts/audit.sol#993-1006)
	- AssetFi._takeDevelopment(uint256) (contracts/audit.sol#1016-1025)
	- AssetFi._takeLiquidity(uint256) (contracts/audit.sol#1008-1014)
	- AssetFi._transferBothExcluded(address,address,uint256) (contracts/audit.sol#866-888)
	- AssetFi._transferFromExcluded(address,address,uint256) (contracts/audit.sol#1203-1224)
	- AssetFi._transferToExcluded(address,address,uint256) (contracts/audit.sol#1180-1201)
	- AssetFi.balanceOf(address) (contracts/audit.sol#727-730)
	- AssetFi.excludeFromReward(address) (contracts/audit.sol#844-851)
	- AssetFi.includeInReward(address) (contracts/audit.sol#853-864)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities
```

# Contract Gas Snapshot

Gas costs:
Gas requirement of function that many  is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

For loop over dynamic array: Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to

```
AssetFi.slitherConstructorVariables() (contracts/audit.sol#654-1225)
uses literals with too many digits:
        - _tTotal = 100000000 * 10 ** 18 (contracts/audit.sol#666)
AssetFi.slitherConstructorVariables() (contracts/audit.sol#654-1225)
uses literals with too many digits:
        - _maxTxAmount = 100000000 * 10 ** 18 (contracts/audit.sol#683)
AssetFi.slitherConstructorVariables() (contracts/audit.sol#654-1225)
uses literals with too many digits:
        - numTokensSellToAddToLiquidity = 100000000 * 10 ** 18
(contracts/audit.sol#684)
Reference: https://github.com/crytic/slither/wiki/Detector-
Documentation#too-many-digits

AssetFi._decimals (contracts/audit.sol#671) should be constant
AssetFi._developmentWalletAddress (contracts/audit.sol#663-664) should
be constant
AssetFi._name (contracts/audit.sol#669) should be constant
AssetFi._symbol (contracts/audit.sol#670) should be constant
AssetFi._tTotal (contracts/audit.sol#666) should be constant
AssetFi.numTokensSellToAddToLiquidity (contracts/audit.sol#684) should
be constant
Reference: https://github.com/crytic/slither/wiki/Detector-
Documentation#state-variables-that-could-be-declared-constant
```

# 0.1 Auto Debugging

No data available
**Storage**
**[Completely Loaded]**
No data available
**Return Value**
•0:Object
**Global Variables**
•block.chainid:0xd05
•block.coinbase:0x0000000000000000000000000000000000000000
•block.difficulty:70762765929000
•block.gaslimit:83124
•block.number:31
•block.timestamp:1679071907
•msg.sender:0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
•msg.sig:0x60566050
•msg.value:0 Wei
•tx.origin:0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
•block.basefee:1 Wei (1)
**Full Storage Changes**
•(Contract Creation - Step 0):Object
**Call Data**
•0:

"vm trace step": 0, "execution step": 0, "add memory": "", "gas": 3, "remaining gas": "28140", "loaded address": "(Contract Creation - Step 0)" }

"0x60566050600b82828239805160001a6073146043577f4e487b710000000000000000000000000000000000000000000000000000000000000000600052600060045260246000fd5b306000526073815382 81f3fe730000000000000000000000000000000000000000301460806040526000 80fdfea2646970667358221220bfdd2b9d194bc8b1d38c8342442670ea5350552ca1f971c7f8b967edb7aa0bd764736f6c63430008120033" ]

# Manual and Automated Vulnerability Test

## CRITICAL ISSUES

During the audit, AudiTBlock experts found **0 big Critical issues** in the code of the smart contract.

## HIGH ISSUES

During the audit, AudiTBlock experts found **0 High issues** in the code of the smart contract.

## MEDIUM ISSUES

During the audit, AudiTBlock experts found **2 Medium issues** in the code of the smart contract.

## LOW ISSUES

During the audit, AudiTBlock experts found **1 Low issues** in the code of the smart contract.

## INFORMATIONAL ISSUES

During the audit, AuditBlock experts found **no Informational issues** in the code of the smart contract.

# SWC Attacks

| ID | Title | | Test Result |
|---|---|---|---|
| SWC-131 | Presence of unused variables | CWE-1164: Irrelevant Code | |
| SWC-130 | Right-To-Left-Override control character (U+202E) | CWE-451: User Interface (UI) Misrepresentation of Critical Information | |
| SWC-129 | Typographical Error | CWE-480: Use of Incorrect Operator | |
| SWC-128 | DoS With Block Gas Limit | CWE-400: Uncontrolled Resource Consumption | |
| SWC-127 | Arbitrary Jump with Function Type Variable | CWE-695: Use of Low-Level Functionality | |
| SWC-125 | Incorrect Inheritance Order | CWE-696: Incorrect Behavior Order | |
| SWC-124 | Write to Arbitrary Storage Location | CWE-123: Write-what-where Condition | |
| SWC-123 | Requirement Violation | CWE-573: Improper Following of Specification by Caller | |

| ID | Title | | Test Result |
|---|---|---|---|
| SWC-113 | DoS with Failed Call | CWE-703: Improper Check or Handling of Exceptional Conditions | |
| SWC-112 | Delegatecall to Untrusted Callee | CWE-829: Inclusion of Functionality from Untrusted Control Sphere | |
| SWC-111 | Use of Deprecated Solidity Functions | CWE-477: Use of Obsolete Function | |
| SWC-110 | Assert Violation | CWE-670: Always-Incorrect Control Flow Implementation | |
| SWC-109 | Uninitialized Storage Pointer | CWE-824: Access of Uninitialized Pointer | |
| SWC-108 | State Variable Default Visibility | CWE-710: Improper Adherence to Coding Standards | |
| SWC-107 | Reentrancy | CWE-841: Improper Enforcement of Behavioral Workflow | |
| SWC-106 | Unprotected SELFDESTRUCT Instruction | CWE-284: Improper Access Control | |
| SWC-105 | Unprotected Ether Withdrawal | CWE-284: Improper Access Control | |
| SWC-104 | Unchecked Call Return Value | CWE-252: Unchecked Return Value | |

# Owner privileges

- Verify Claims

- The contract block difficulty 70762765929000
  Status: tested and verified

- UniswapV2Pair
  Status: tested and verified

- UniswapV2Router
  Status: tested and verified

- Fee
  Status: tested and verified

- Reward
  Status: tested and verified

- Owner/Deployer
  Status: tested and verified

## Executive Summary

Two (2) independent AuditBlock experts performed an unbiased and isolated audit of the smart contract codebase. The final debriefs
The overall code quality is good and not overloaded with unnecessary functions, these is greatly

benefiting the security of the contract. It correctly implemented widely used and reviewed contracts from OpenZeppelin. he main goal of the audit was to verify the claims regarding the security of the smart contract and the claims inside the scope of work.

During the audit, no issues were found after the manual and automated security testing.

## Deployed On BNB Mainnet Binance Chain

VERIFIED ✔

https://bscscan.com/address/0xcd50fcc52a0051757695eef3230244eebb5567d8