SHARK (SHARK) TOKEN SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer:	Shark Team (SHARK)
Website:	https://shark.capital/
Prepared on:	15/05/2021
Platform:	Binance Smart Chain
Language:	Solidity
Audit Type:	Standard

audit@etherauthority.io

Table of contents

THIS IS SECURITY AUDIT REPORT DOCUMENT AND WHICH MAY CONTAIN INFORMATION WHICH IS CONFIDENTIAL. WHICH INCLUDES ANY POTENTIAL VULNERABILITIES AND MALICIOUS CODES WHICH CAN BE USED TO EXPLOIT THE SOFTWARE. THIS MUST BE REFERRED INTERNALLY AND ONLY SHOULD BE MADE AVAILABLE TO PUBLIC AFTER ISSUES ARE RESOLVED.

Email: audit@EtherAuthority.io

Project file

Name	Code Review and Security Analysis Report for SHARK (SHARK) Token Smart Contract
Platform	BSC / Solidity
File	SHARK.sol
File MD5 hash	57EF3C577EDA596678D8C6C290ED5212
Online Contract Code	https://testnet.bscscan.com/address/0x51bc7ad318a6b90d7d76f946914edfa1b34750c5

Introduction

We were contracted by the SHARK team to perform the Security audit of the SHARK Token smart contract code. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on 15/05/2021.

The Audit type was Standard Audit. Which means this audit is concluded based on Standard audit scope, which is one security engineer performing audit procedure for 2 days. This document outlines all the findings as well as an AS-IS overview of the smart contract codes.

Quick Stats:

Main Category	Subcategory	Result
Contract Programming	Solidity version not specified	Passed
	Solidity version too old	Moderated
	Integer overflow/underflow	Passed
	Function input parameters lack of check	Passed
	Function input parameters check bypass	Passed
	Function access control lacks management	Passed
	Critical operation lacks event log	Moderated
	Human/contract checks bypass	Passed
	Random number generation/use vulnerability	Passed
	Fallback function misuse	Passed
	Race condition	Passed
	Logical vulnerability	Passed
	Other programming issues	Passed
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Passed
	Other code specification issues	Passed
Gas Optimization	Assert() misuse	Passed
	High consumption ‘for/while’ loop	Moderated
	High consumption ‘storage’ storage	Passed
	“Out of Gas” Attack	Passed
Business Risk	The maximum limit for mintage not set	Passed
	“Short Address” Attack	Passed
	“Double Spend” Attack	Passed

Overall Audit Result: PASSED

Executive Summary

According to the **standard** audit assessment, Customer`s solidity smart contract is **Well secured**.

We used various tools like Mythril, Slither and Remix IDE. At the same time this finding is based on critical analysis of the manual audit.
All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the Audit overview section. General overview is presented in AS-IS section and all identified issues can be found in the Audit overview section.

We found 0 critical, 0 high, 0 medium and 5 low and some very low level issues.

Code Quality

SHARK Token smart contract has 1 smart contract. This smart contract also contains Libraries, Smart contract inherits and Interfaces. These are compact and well written contracts.

The libraries in the SHARK Token protocol are part of its logical algorithm. A library is a different type of smart contract that contains reusable code. Once deployed on the blockchain (only once), it is assigned a specific address and its properties / methods can be reused many times by other contracts in the SHARK Token protocol.

The SHARK team has **not** provided scenario and unit test scripts, which would have helped to determine the integrity of the code in an automated way.

Overall, code parts are **not well** commented on smart contracts.

Documentation

We were given SHARK token smart contracts code in the form of a BscScan web link. The hash of that code and that web link are mentioned above in the table.

As mentioned above, most code parts are **not well** commented. so it is difficult to quickly understand the programming flow as well as complex code logic. Comments are very helpful in understanding the overall architecture of the protocol.

Another source of information was its official website which provided rich information about the project architecture and tokenomics.

Use of Dependencies

As per our observation, the libraries are used in this smart contract infrastructure that are based on well known industry standard open source projects. And their core code blocks are written well.

Apart from libraries, its functions are used in smart contract calls.

AS-IS overview

SHARK (SHARK) is a community-focused, decentralized digital asset with instant rewards for holders as well as auto-burn liquidity.

SHARK.sol

(1) Interfaces

(a) IBEP20

(b) IUniswapV2Factory

(c) IUniswapV2Pair

(d) IUniswapV2Router01

(e) IUniswapV2Router02

(2) Inherited contracts

(a) Context: Context contract.

(b) Ownable: Ownership contract.

(c) IBEP20: IBEP20 contract.

(3) Usages

(a) using SafeMath for uint256;

(b) using Address for address;

(4) Events

(a) event Transfer(address indexed from, address indexed to, uint256 value);

(b) event Approval(address indexed owner, address indexed spender, uint256 value);

(c) event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);

(d) event PairCreated(address indexed token0, address indexed token1, address pair, uint);

(e) event Approval(address indexed owner, address indexed spender, uint value);

(f) event Transfer(address indexed from, address indexed to, uint value);

(g) event Mint(address indexed sender, uint amount0, uint amount1);

(h) event Burn(address indexed sender, uint amount0, uint amount1, address indexed to);

(i) event Swap(address indexed sender, uint amount0In,uint amount1In, uint amount0Out, uint amount1Out, address indexed to);

(j) event Sync(uint112 reserve0, uint112 reserve1);

(k) event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);

(l) event SwapAndLiquifyEnabledUpdated(bool enabled);

(m) event SwapAndLiquify(uint256 tokensSwapped,uint256 ethReceived,uint256 tokensIntoLiqudity);

(5) Functions

Sl.	Functions	Type	Observation	Conclusion
1	name	read	Passed	No Issue
2	symbol	read	Passed	No Issue
3	decimals	read	Passed	No Issue
4	totalSupply	read	Passed	No Issue
5	balanceOf	read	Passed	No Issue
6	transfer	write	Passed	No Issue
7	allowance	read	Passed	No Issue
8	approve	write	Passed	No Issue
9	transferFrom	write	Passed	No Issue
10	increaseAllowance	write	Passed	No Issue
11	decreaseAllowance	write	Passed	No Issue
12	isExcludedFromReward	read	Passed	No Issue
13	totalFees	write	Passed	No Issue
14	deliver	write	Passed	No Issue
15	reflectionFromToken	read	Passed	No Issue
16	tokenFromReflection	read	Passed	No Issue

17	excludeFromReward	write	access by only owner	No Issue
18	includeInReward	external	Infinite loop possibility	**Refer Audit Findings**
19	_transferBothExcluded	write	Infinite loop possibility	No Issue
20	excludeFromFee	write	access by only owner	No Issue
21	includeInFee	write	Missing events emitting	No Issue
22	setTaxFeePercent	external	Missing events emitting	No Issue
23	setLiquidityFeePercent	external	Missing events emitting	No Issue
24	setMaxTxPercent	external	Missing events emitting	No Issue
25	setSwapAndLiquifyEnabled	write	Missing events emitting	No Issue
26	_reflectFee	write	Passed	No Issue
27	_getValues	read	Passed	No Issue
28	_getTValues	read	Passed	No Issue
29	_getRValues	write	Passed	No Issue
30	_getRate	read	Passed	No Issue
31	_getCurrentSupply	read	Passed	No Issue
32	_takeLiquidity	write	Passed	No Issue
33	calculateTaxFee	read	Passed	No Issue
34	calculateLiquidityFee	read	Passed	No Issue
35	removeAllFee	write	Passed	No Issue
36	restoreAllFee	write	Passed	No Issue
37	isExcludedFromFee	read	Passed	No Issue
38	_approve	write	Passed	No Issue
39	_transfer	write	Passed	No Issue
40	swapAndLiquify	write	Passed	No Issue
41	swapTokensForEth	write	Passed	No Issue
42	_tokenTransfer	write	Passed	No Issue
43	addLiquidity	write	Ownership control	**Refer Audit Findings**

44	_transferStandard	write	Passed	No Issue
45	_transferFromExcluded	write	Passed	No Issue
46	_msgSender	read	Passed	No Issue
47	_msgData	read	Passed	No Issue
48	owner	read	Passed	No Issue
49	renounceOwnership	write	Passed	No Issue
50	transferOwnership	write	Passed	No Issue
51	geUnlockTime	read	Passed	No Issue
52	lock	write	Passed	No Issue
53	unlock	write	Passed	No Issue
54	_transferToExcluded	write	Passed	No Issue

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to tokens loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial functions
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to tokens lose
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have significant impact on execution
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can be ignored.

Audit Findings

Critical

No critical severity vulnerabilities were found.

High

No High severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

(1) Centralized risk in addLiquidity

```
function addLiquidity(uint256 tokenAmount, uint256 ethAmount) private {
    // approve token transfer to cover all possible scenarios
    _approve(address(this), address(uniswapV2Router), tokenAmount);
    // add the liquidity
    uniswapV2Router.addLiquidityETH{value: ethAmount}(
        address(this),
        tokenAmount,
        0, // slippage is unavoidable
        0, // slippage is unavoidable
        owner(),
        block.timestamp
    );
}
```

AddLiquidityETH function has to be addressed as owner() to get LP Tokens from Pool. At some time, The owner will accumulate significant LP tokens. If the _owner is an EOA(Externally Owned Account), mishandling of its private key can have devastating consequences to the project.

Solution: uniswapV2Router.addLiquidityETH function call to be replaced by address (this), and to restrict the management of the LP tokens. This will protect the LP tokens from being stolen if the _owner account is compromised.

(2) Missing Events emitting

There are some functions which change the state variable that should emit events.

Following Functions List:

- deliver
- excludeFromFee
- excludeFromReward
- includeInFee
- includeInReward
- setLiquidityFeePercent
- setMaxTxPercent
- setTaxFeePercent

Solution: For all the state variables which change runtime this needs to emit an event. Recommended to emit events for all these functions.

(3) Infinite loop possibility

```
function includeInReward(address account) external onlyOwner() {
    require(_isExcluded[account], "Account is already excluded");
    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;
        }
    }
}
```

If there are so many excluded wallets, then this logic will fail, as it might hit the block's gas limit. If there are very limited exceptions, then this will work, but will cost more gas.

Solution: Just use a mapping that will map wallet to bool and make excluded wallets to be true. This logic will not have any gas or scalability issues.

(4) Possible to gain ownership after renouncing the contract ownership Owner can renounce ownership and make contract without owner but here is a catch owner can misuse it by performing the following operations:

1) Owner calls the lock function in contract to set the current owner as _previousOwner.

2) Owner calls unlock to unlock contract and set _owner = _previousOwner.

3) Owner called renounceOwnership to leave the contract without the owner.

4) Owner calls unlock to regain ownership.

Solution: We advise updating/removing lock and unlock functions in the contract or call renounceOwnership function first before calling lock/unlock functions.

(5) Variable could be declared as constant

```
uint256 private constant MAX = ~uint256(0);
uint256 private _tTotal = 1000000 * 10**6 * 10**9;
uint256 private _rTotal = (MAX - (MAX % _tTotal));
uint256 private _tFeeTotal;
string private _name = "SHARK";
string private _symbol = "SHARK";
uint8 private _decimals = 9;

uint256 public _taxFee = 2;
uint256 private _previousTaxFee = _taxFee;

uint256 public _liquidityFee = 2;
uint256 private _previousLiquidityFee = _liquidityFee;
IUniswapV2Router02 public immutable uniswapV2Router;
address public immutable uniswapV2Pair;

bool inSwapAndLiquify;
bool public swapAndLiquifyEnabled = true;

uint256 public _maxTxAmount = 200000 * 10**6 * 10**9;
uint256 private numTokensSellToAddToLiquidity = 2000 * 10**6 * 10**9;
```

States variables that never change need to be declared as constants.

Variables Like:

- _name
- _symbol
- _decimals
- _tTotal
- numTokensSellToAddToLiquidity

It should be declared as constant.

Solution: This issue is acknowledged.

Very Low / Discussion / Best practices:

(1) Solidity version

```
pragma solidity ^0.6.12;
```

Use the latest solidity version while contract deployment to prevent any compiler version level bugs.

Solution: This issue is acknowledged.

(2) Redundant code

```
if (_isExcluded[sender] && !_isExcluded[recipient]) {
    _transferFromExcluded(sender, recipient, amount);
} else if (!_isExcluded[sender] && _isExcluded[recipient]) {
    _transferToExcluded(sender, recipient, amount);
} else if (!_isExcluded[sender] && !_isExcluded[recipient]) {
    _transferStandard(sender, recipient, amount);
} else if (_isExcluded[sender] && _isExcluded[recipient]) {
    _transferBothExcluded(sender, recipient, amount);
} else {
    _transferStandard(sender, recipient, amount);
}
```

The condition !_isExcluded[sender] && !_isExcluded[recipient] is not needed it can be included in else condition so no need this extra condition here

Solution: Line no : 991,992 need to removed

} else if (!_isExcluded[sender] && !_isExcluded[recipient]) {

_transferStandard(sender, recipient, amount);

(3) function and variable names do not match with bsc network.

```
uint256 half = contractTokenBalance.div(2);
uint256 otherHalf = contractTokenBalance.sub(half);
// capture the contract's current ETH balance.
// this is so that we can capture exactly the amount of ETH that the
// swap creates, and not make the liquidity event include any ETH that
// has been manually sent to the contract
uint256 initialBalance = address(this).balance;
// swap tokens for ETH
swapTokensForEth(half); // <- this breaks the ETH -> HATE swap when swap+liquify is triggered
// how much ETH did we just swap into?
uint256 newBalance = address(this).balance.sub(initialBalance);
```

This contract is for BSC. The comments and some functions have ETH text in it. But, it should be BNB. Some functions have used Uniswap but it should be Pancakeswap. This naming should be changed.

Solution:

- Change the ETH to BNB in comments.
- Change Uniswap to PancakeSwap to remove any confusion.
- Change IuniswapV2Router01 to IpancakeRouter01.
- Change IuniswapV2Router02 to IpancakeRouter02.
- Change uniswapV2Router to PancakeRouter.
- Change uniswapV2Pair to pancakePair.
- Change all uniswap to pancake and ETH to BNB.

(4) Typing mistake in contract.

```
event SwapAndLiquify(
    uint256 tokensSwapped,
    uint256 ethReceived,
    uint256 tokensIntoLiqudity
);
```

There are many typing mistakes in code and comments. TokensIntoLiqudity should be tokensIntoLiquidity, recieve should be receive, swapping should be swapping.

Solution: We recommend correcting all typing mistakes in the contract.

Centralization

This smart contract has some functions which can be executed by Admin (Ownable) only. If the admin wallet private key would be compromised, then it puts this smart contract in the hands of an attacker. Following are Admin functions:

- Owner can use the delivery function and send tokens to any wallet.
- Owner has permission to change the owner address and receive LP Tokens.
- Owner can lock the contract.
- Owner can enable/disable swapAndLiquifyEnabled.
- Owner can set Tax Fee Percent, Max Tx Percent, Liquidity fee percent ,etc.
- Owner can include/exclude any wallet from reward and fees.

Conclusion

We were given a contract code. And we have used all possible tests based on given objects as files. We observed some issues in the smart contract and those are fixed/acknowledged in the smart contract. **So it is good to go for the production.**

Since possible test cases can be unlimited for such extensive smart contract protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan everything.

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. Smart Contract's high level description of functionality was presented in As-is overview section of the report.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

Security state of the reviewed contract, based on standard audit procedure scope, is "**Well Secured**".

Our Methodology

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our security audits are to improve the quality of systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

Manual Code Review:

In manually reviewing all of the code, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

Vulnerability Analysis:

Our audit techniques included manual code analysis, user interface interaction, and whitebox penetration testing. We look at the project's web site to get a high level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. While we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation.

Documenting Results:

We follow a conservative, transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even though we have not yet verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this we analyze the feasibility of an attack in a live system.

Suggested Solutions:

We search for immediate mitigations that live deployments can take, and finally we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinized by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our report, and before the details are made public.

Disclaimers

EtherAuthority.io Disclaimer

EtherAuthority team has analyzed this smart contract in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment and functionality (performing the intended functions).

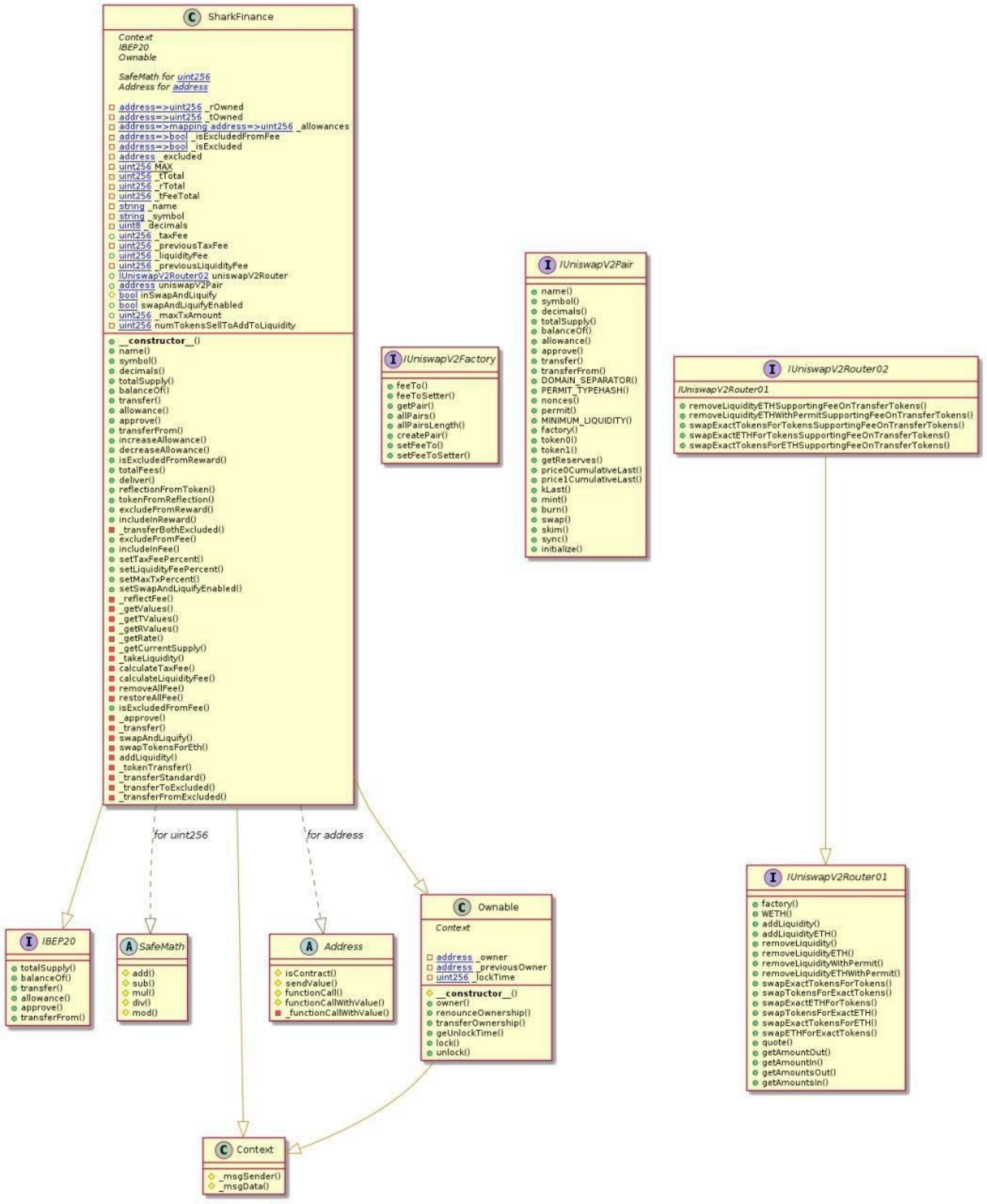
Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only. We also suggest conducting a bug bounty program to confirm the high level of security of this smart contract.

Technical Disclaimer

Smart contracts are deployed and executed on the blockchain platform. The platform, its programming language, and other software related to the smart contract can have their own vulnerabilities that can lead to hacks. Thus, the audit can’t guarantee explicit security of the audited smart contracts.

Appendix

Code Flow Diagram - SHARK Token



Slither Results Log

>>> SLITHER REPORT SHARK.sol

root@xcv-ThinkPad-T410:/home/xcv/tempSlither#
root@xcv-ThinkPad-T410:/home/xcv/tempSlither# **slither**
SHARK.sol INFO:Detectors:
Reentrancy in SHARK._transfer(address,address,uint256)
(SHARK.sol#914-956): External calls:
- swapAndLiquify(contractTokenBalance) (SHARK.sol#943)
- uniswapV2Router.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,owner(),block.timestamp)
(SHARK.sol#994-1001) -
uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmou
nt,0,path, address(this),block.timestamp) (SHARK.sol#982-988)
External calls sending eth:
- swapAndLiquify(contractTokenBalance) (SHARK.sol#943)
- uniswapV2Router.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,owner(),block.timestamp)
(SHARK.sol#994-1001) State variables written after the call(s):
- _tokenTransfer(from,to,amount,takeFee) (SHARK.sol#955)
- _rOwned[address(this)] = _rOwned[address(this)].add(rLiquidity)
(SHARK.sol#874)
- _rOwned[sender] = _rOwned[sender].sub(rAmount)
(SHARK.sol#1033) - _rOwned[sender] =
_rOwned[sender].sub(rAmount) (SHARK.sol#1025) -
_rOwned[recipient] = _rOwned[recipient].add(rTransferAmount)
(SHARK.sol#1026)
- _rOwned[sender] = _rOwned[sender].sub(rAmount)
(SHARK.sol#797) - _rOwned[sender] =
_rOwned[sender].sub(rAmount) (SHARK.sol#1043) -
_rOwned[recipient] = _rOwned[recipient].add(rTransferAmount)
(SHARK.sol#1035)
- _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount)
(SHARK.sol#1044)
- _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount)
(SHARK.sol#799)
- _tokenTransfer(from,to,amount,takeFee) (SHARK.sol#955)
- _rTotal = _rTotal.sub(rFee) (SHARK.sol#834)
- _tokenTransfer(from,to,amount,takeFee) (SHARK.sol#955)
- _tOwned[address(this)] = _tOwned[address(this)].add(tLiquidity)
(SHARK.sol#876)
- _tOwned[sender] = _tOwned[sender].sub(tAmount)
(SHARK.sol#796) - _tOwned[sender] =
_tOwned[sender].sub(tAmount) (SHARK.sol#1042) -
_tOwned[recipient] = _tOwned[recipient].add(tTransferAmount)
(SHARK.sol#1034)
- _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount)
(SHARK.sol#798)
Reference:

https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy vulnerabilities
INFO:Detectors:
SHARK.addLiquidity(uint256,uint256) (SHARK.sol#990-1002) ignores return value by uniswapV2Router.addLiquidityETH{value: ethAmount} (address(this),tokenAmount,0,0,owner(),block.timestamp) (SHARK.sol#994-1001) Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
SHARK.allowance(address,address).owner (SHARK.sol#724) shadows: - Ownable.owner() (SHARK.sol#406-408) (function)
SHARK._approve(address,address,uint256).owner (SHARK.sol#908) shadows: - Ownable.owner() (SHARK.sol#406-408) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Reentrancy in SHARK._transfer(address,address,uint256) (SHARK.sol#914-956): External calls:
- swapAndLiquify(contractTokenBalance) (SHARK.sol#943)
  - uniswapV2Router.addLiquidityETH{value: ethAmount} (address(this),tokenAmount,0,0,owner(),block.timestamp) (SHARK.sol#994-1001) - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path, address(this),block.timestamp) (SHARK.sol#982-988)
External calls sending eth:
- swapAndLiquify(contractTokenBalance) (SHARK.sol#943)
  - uniswapV2Router.addLiquidityETH{value: ethAmount} (address(this),tokenAmount,0,0,owner(),block.timestamp) (SHARK.sol#994-1001) State variables written after the call(s):
- _tokenTransfer(from,to,amount,takeFee) (SHARK.sol#955)
  - _liquidityFee = _previousLiquidityFee (SHARK.sol#902)
  - _liquidityFee = 0 (SHARK.sol#897)
- _tokenTransfer(from,to,amount,takeFee) (SHARK.sol#955)
  - _previousLiquidityFee = _liquidityFee (SHARK.sol#894)
- _tokenTransfer(from,to,amount,takeFee) (SHARK.sol#955)
- _previousTaxFee = _taxFee (SHARK.sol#893)
- _tokenTransfer(from,to,amount,takeFee) (SHARK.sol#955)
  - _tFeeTotal = _tFeeTotal.add(tFee) (SHARK.sol#835)
- _tokenTransfer(from,to,amount,takeFee) (SHARK.sol#955)
  - _taxFee = _previousTaxFee (SHARK.sol#901)
  - _taxFee = 0 (SHARK.sol#896)

Reentrancy in SHARK.constructor() (SHARK.sol#688-703): External calls:
- uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV2Router.WE TH()) (SHARK.sol#693-694)

State variables written after the call(s):
- _isExcludedFromFee[owner()] = true (SHARK.sol#699)
- _isExcludedFromFee[address(this)] = true (SHARK.sol#700)
- uniswapV2Router = _uniswapV2Router (SHARK.sol#696)

Reentrancy in SHARK.swapAndLiquify(uint256) (SHARK.sol#957-974): External calls:
- swapTokensForEth(half) (SHARK.sol#967)
  -

uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path, address(this),block.timestamp) (SHARK.sol#982-988)
- addLiquidity(otherHalf,newBalance) (SHARK.sol#971)
- uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (SHARK.sol#994-1001) External calls sending eth:
- addLiquidity(otherHalf,newBalance) (SHARK.sol#971)
- uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (SHARK.sol#994-1001) State variables written after the call(s):
- addLiquidity(otherHalf,newBalance) (SHARK.sol#971)
- _allowances[owner][spender] = amount (SHARK.sol#911)

Reentrancy in SHARK.transferFrom(address,address,uint256) (SHARK.sol#731-735): External calls:
- _transfer(sender,recipient,amount) (SHARK.sol#732)
- uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (SHARK.sol#994-1001) - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path, address(this),block.timestamp) (SHARK.sol#982-988) External calls sending eth:
- _transfer(sender,recipient,amount) (SHARK.sol#732)
- uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (SHARK.sol#994-1001) State variables written after the call(s):
- _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,BEP20: transfer amount exceeds allowance)) (SHARK.sol#733)
- _allowances[owner][spender] = amount (SHARK.sol#911)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy vulnerabilities-2

INFO:Detectors:

Reentrancy in SHARK._transfer(address,address,uint256) (SHARK.sol#914-956): External calls:
- swapAndLiquify(contractTokenBalance) (SHARK.sol#943)
- uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (SHARK.sol#994-1001) - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path, address(this),block.timestamp) (SHARK.sol#982-988) External calls sending eth:
- swapAndLiquify(contractTokenBalance) (SHARK.sol#943)
- uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (SHARK.sol#994-1001) Event emitted after the call(s):
- Transfer(sender,recipient,tTransferAmount) (SHARK.sol#1029)
- _tokenTransfer(from,to,amount,takeFee) (SHARK.sol#955)
- Transfer(sender,recipient,tTransferAmount) (SHARK.sol#1047)
- _tokenTransfer(from,to,amount,takeFee) (SHARK.sol#955)
- Transfer(sender,recipient,tTransferAmount) (SHARK.sol#1038)
- _tokenTransfer(from,to,amount,takeFee) (SHARK.sol#955)
- Transfer(sender,recipient,tTransferAmount) (SHARK.sol#802)

- _tokenTransfer(from,to,amount,takeFee) (SHARK.sol#955)
Reentrancy in SHARK.constructor() (SHARK.sol#688-703):
External calls:
- uniswapV2Pair =
IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV2Ro
uter.WE TH()) (SHARK.sol#693-694)
Event emitted after the call(s):
- Transfer(address(0),_msgSender(),_tTotal) (SHARK.sol#702)
Reentrancy in SHARK.swapAndLiquify(uint256)
(SHARK.sol#957-974): External calls:
- swapTokensForEth(half) (SHARK.sol#967)
-
uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmou
nt,0,path, address(this),block.timestamp) (SHARK.sol#982-988)
- addLiquidity(otherHalf,newBalance) (SHARK.sol#971)
- uniswapV2Router.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,owner(),block.timestamp)
(SHARK.sol#994-1001) External calls sending eth:
- addLiquidity(otherHalf,newBalance) (SHARK.sol#971)
- uniswapV2Router.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,owner(),block.timestamp)
(SHARK.sol#994-1001) Event emitted after the call(s):
- Approval(owner,spender,amount) (SHARK.sol#912)
- addLiquidity(otherHalf,newBalance) (SHARK.sol#971)
- SwapAndLiquify(half,newBalance,otherHalf) (SHARK.sol#973)
Reentrancy in SHARK.transferFrom(address,address,uint256)
(SHARK.sol#731-735): External calls:
- _transfer(sender,recipient,amount) (SHARK.sol#732)
- uniswapV2Router.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,owner(),block.timestamp)
(SHARK.sol#994-1001) -
uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmou
nt,0,path, address(this),block.timestamp) (SHARK.sol#982-988)
External calls sending eth:
- _transfer(sender,recipient,amount) (SHARK.sol#732)
- uniswapV2Router.addLiquidityETH{value: ethAmount}
(address(this),tokenAmount,0,0,owner(),block.timestamp)
(SHARK.sol#994-1001) Event emitted after the call(s):
- Approval(owner,spender,amount) (SHARK.sol#912)
- _approve(sender,_msgSender(),_allowances[sender]
[_msgSender()].sub(amount,BEP20: transfer amount exceeds allowance))
(SHARK.sol#733) Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy vulnerabilities-3
INFO:Detectors:
Ownable.unlock() (SHARK.sol#448-453) uses timestamp for
comparisons Dangerous comparisons:
- require(bool,string)(now > _lockTime,Contract is locked until 7 days)
(SHARK.sol#450)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (SHARK.sol#271-280) uses assembly
- INLINE ASM (SHARK.sol#278)

Address._functionCallWithValue(address,bytes,uint256,string)
(SHARK.sol#357-376) uses assembly
- INLINE ASM (SHARK.sol#368-371)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
SHARK._rTotal (SHARK.sol#654) is set pre-construction with a non-constant function or state variable:
- (MAX - (MAX % _tTotal))
SHARK._previousTaxFee (SHARK.sol#661) is set pre-construction with a non-constant function or state variable:
- _taxFee
SHARK._previousLiquidityFee (SHARK.sol#664) is set pre-construction with a non constant function or state variable:
- _liquidityFee
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing state-variables
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (SHARK.sol#297-302): - (success) = recipient.call{value: amount}() (SHARK.sol#300)
Low level call in
Address._functionCallWithValue(address,bytes,uint256,string)
(SHARK.sol#357-376):
- (success,returndata) = target.call{value: weiValue}(data)
(SHARK.sol#360) Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (SHARK.sol#480) is not in mixedCase Function IUniswapV2Pair.PERMIT_TYPEHASH() (SHARK.sol#481) is not in mixedCase Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (SHARK.sol#495) is not in mixedCase Function IUniswapV2Router01.WETH() (SHARK.sol#513) is not in mixedCase Parameter SHARK.setSwapAndLiquifyEnabled(bool)._enabled (SHARK.sol#826) is not in mixedCase
Parameter SHARK.calculateTaxFee(uint256)._amount
(SHARK.sol#879) is not in mixedCase
Parameter SHARK.calculateLiquidityFee(uint256)._amount (SHARK.sol#884) is not in mixedCase
Variable SHARK._taxFee (SHARK.sol#660) is not in mixedCase Variable SHARK._liquidityFee (SHARK.sol#663) is not in mixedCase Variable SHARK._maxTxAmount (SHARK.sol#671) is not in mixedCase Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity naming-conventions
INFO:Detectors:
Redundant expression "this (SHARK.sol#246)" inContext (SHARK.sol#241-249)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements INFO:Detectors:
Variable
IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint25 6).amountADesired (SHARK.sol#517) is too similar to

IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,addre
ss,uint25 6).amountBDesired (SHARK.sol#518)
Variable
SHARK._transferToExcluded(address,address,uint256).rTransferAmount
(SHARK.sol#1032) is too similar to
SHARK._transferBothExcluded(address,address,uint256).tTransfer Amount
(SHARK.sol#795)
Variable
SHARK._transferToExcluded(address,address,uint256).rTransferAmount
(SHARK.sol#1032) is too similar to
SHARK._getTValues(uint256).tTransferAmount (SHARK.sol#845)
Variable SHARK._getValues(uint256).rTransferAmount (SHARK.sol#839) is too similar to
SHARK._transferFromExcluded(address,address,uint256).tTransferAmount
(SHARK.sol#1041)
Variable
SHARK._transferBothExcluded(address,address,uint256).rTransferAmount
(SHARK.sol#795) is too similar to
SHARK._transferFromExcluded(address,address,uint256).tTransfer Amount
(SHARK.sol#1041)
Variable SHARK._transferFromExcluded(address,address,uint256).rTransferAmount
(SHARK.sol#1041) is too similar to
SHARK._transferBothExcluded(address,address,uint256).tTransfer
Amount (SHARK.sol#795) Variable
SHARK._transferFromExcluded(address,address,uint256).rTransferAmount
(SHARK.sol#1041) is too similar to SHARK._getTValues(uint256).tTransferAmount
(SHARK.sol#845) Variable
SHARK._transferBothExcluded(address,address,uint256).rTransferAmount
(SHARK.sol#795) is too similar to
SHARK._transferBothExcluded(address,address,uint256).tTransfer
Amount (SHARK.sol#795) Variable
SHARK._transferBothExcluded(address,address,uint256).rTransferAmount
(SHARK.sol#795) is too similar to SHARK._getTValues(uint256).tTransferAmount
(SHARK.sol#845) Variable
SHARK._transferFromExcluded(address,address,uint256).rTransferAmount
(SHARK.sol#1041) is too similar to
SHARK._transferFromExcluded(address,address,uint256).tTransfer Amount
(SHARK.sol#1041)
Variable SHARK._getRValues(uint256,uint256,uint256,uint256).rTransferAmount
(SHARK.sol#852) is too similar to
SHARK._transferFromExcluded(address,address,uint256).tTransfer Amount
(SHARK.sol#1041)
Variable
SHARK._getRValues(uint256,uint256,uint256,uint256).rTransferAmount
(SHARK.sol#852) is too similar to
SHARK._transferBothExcluded(address,address,uint256).tTransfer Amount
(SHARK.sol#795)

Variable
SHARK._getRValues(uint256,uint256,uint256,uint256).rTransferAmount
(SHARK.sol#852) is too similar to
SHARK._getTValues(uint256).tTransferAmount (SHARK.sol#845) Variable
SHARK._transferToExcluded(address,address,uint256).rTransferAmount
(SHARK.sol#1032) is too similar to
SHARK._getValues(uint256).tTransferAmount (SHARK.sol#838) Variable
SHARK._transferToExcluded(address,address,uint256).rTransferAmount
(SHARK.sol#1032) is too similar to
SHARK._transferFromExcluded(address,address,uint256).tTransfer Amount
(SHARK.sol#1041)
Variable SHARK._getValues(uint256).rTransferAmount (SHARK.sol#839) is too similar to
SHARK._transferBothExcluded(address,address,uint256).tTransferAmount
(SHARK.sol#795)
Variable SHARK._getValues(uint256).rTransferAmount (SHARK.sol#839) is too similar to
SHARK._getTValues(uint256).tTransferAmount (SHARK.sol#845) Variable
SHARK._transferToExcluded(address,address,uint256).rTransferAmount
(SHARK.sol#1032) is too similar to
SHARK._transferToExcluded(address,address,uint256).tTransfer
Amount (SHARK.sol#1032) Variable
SHARK._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (SHARK.sol#852) is too similar to SHARK._transferStandard(address,address,uint256).tTransfer
Amount (SHARK.sol#1024) Variable
SHARK._transferToExcluded(address,address,uint256).rTransferAmount (SHARK.sol#1032) is too similar to SHARK._transferStandard(address,address,uint256).tTransfer
Amount (SHARK.sol#1024)
Variable SHARK._getValues(uint256).rTransferAmount (SHARK.sol#839) is too similar to
SHARK._transferToExcluded(address,address,uint256).tTransferAmount
(SHARK.sol#1032)
Variable
SHARK._getRValues(uint256,uint256,uint256,uint256).rTransferAmount
(SHARK.sol#852) is too similar to
SHARK._transferToExcluded(address,address,uint256).tTransfer Amount
(SHARK.sol#1032)
Variable SHARK._getValues(uint256).rTransferAmount (SHARK.sol#839) is too similar to
SHARK._getValues(uint256).tTransferAmount (SHARK.sol#838) Variable
SHARK._transferStandard(address,address,uint256).rTransferAmount (SHARK.sol#1024) is too similar to SHARK._transferToExcluded(address,address,uint256).tTransfer
Amount (SHARK.sol#1032)
Variable SHARK.reflectionFromToken(uint256,bool).rTransferAmount
(SHARK.sol#764) is too similar to SHARK._getValues(uint256).tTransferAmount
(SHARK.sol#838) Variable

SHARK._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (SHARK.sol#852) is too similar to SHARK._getValues(uint256).tTransferAmount (SHARK.sol#838) Variable SHARK._transferBothExcluded(address,address,uint256).rTransferAmount (SHARK.sol#795) is too similar to SHARK._transferStandard(address,address,uint256).tTransfer Amount (SHARK.sol#1024) Variable SHARK.reflectionFromToken(uint256,bool).rTransferAmount (SHARK.sol#764) is too similar to SHARK._transferBothExcluded(address,address,uint256).tTransfer Amount (SHARK.sol#795)
Variable SHARK.reflectionFromToken(uint256,bool).rTransferAmount (SHARK.sol#764) is too similar to SHARK._getTValues(uint256).tTransferAmount (SHARK.sol#845) Variable SHARK._transferFromExcluded(address,address,uint256).rTransferAmount (SHARK.sol#1041) is too similar to SHARK._transferToExcluded(address,address,uint256).tTransfer Amount (SHARK.sol#1032) Variable SHARK.reflectionFromToken(uint256,bool).rTransferAmount (SHARK.sol#764) is too similar to SHARK._transferStandard(address,address,uint256).tTransfer Amount (SHARK.sol#1024)
Variable SHARK._transferFromExcluded(address,address,uint256).rTransferAmount (SHARK.sol#1041) is too similar to SHARK._transferStandard(address,address,uint256).tTransfer Amount (SHARK.sol#1024) Variable SHARK.reflectionFromToken(uint256,bool).rTransferAmount (SHARK.sol#764) is too similar to SHARK._transferToExcluded(address,address,uint256).tTransfer Amount (SHARK.sol#1032)
Variable SHARK._transferStandard(address,address,uint256).rTransferAmount (SHARK.sol#1024) is too similar to SHARK._getValues(uint256).tTransferAmount (SHARK.sol#838) Variable SHARK._transferBothExcluded(address,address,uint256).rTransferAmount (SHARK.sol#795) is too similar to SHARK._transferToExcluded(address,address,uint256).tTransfer Amount (SHARK.sol#1032) Variable SHARK._transferBothExcluded(address,address,uint256).rTransferAmount (SHARK.sol#795) is too similar to SHARK._getValues(uint256).tTransferAmount (SHARK.sol#838) Variable SHARK._transferStandard(address,address,uint256).rTransferAmount (SHARK.sol#1024) is too similar to SHARK._transferBothExcluded(address,address,uint256).tTransfer Amount (SHARK.sol#795)

Variable SHARK._transferStandard(address,address,uint256).rTransferAmount (SHARK.sol#1024) is too similar to SHARK._getTValues(uint256).tTransferAmount (SHARK.sol#845) Variable SHARK._transferFromExcluded(address,address,uint256).rTransferAmount (SHARK.sol#1041) is too similar to SHARK._getValues(uint256).tTransferAmount (SHARK.sol#838) Variable SHARK._transferStandard(address,address,uint256).rTransferAmount (SHARK.sol#1024) is too similar to SHARK._transferStandard(address,address,uint256).tTransfer Amount (SHARK.sol#1024)
Variable SHARK.reflectionFromToken(uint256,bool).rTransferAmount (SHARK.sol#764) is too similar to SHARK._transferFromExcluded(address,address,uint256).tTransfer Amount (SHARK.sol#1041)
Variable SHARK._transferStandard(address,address,uint256).rTransferAmount (SHARK.sol#1024) is too similar to SHARK._transferFromExcluded(address,address,uint256).tTransfer Amount (SHARK.sol#1041)
Variable SHARK._getValues(uint256).rTransferAmount (SHARK.sol#839) is too similar to SHARK._transferStandard(address,address,uint256).tTransferAmount (SHARK.sol#1024)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too similar
INFO:Detectors:
SHARK.slitherConstructorVariables() (SHARK.sol#642-1051) uses literals with too many digits:
- _tTotal = 1000000 * 10 ** 6 * 10 ** 9 (SHARK.sol#653)
SHARK.slitherConstructorVariables() (SHARK.sol#642-1051) uses literals with too many digits:
- _maxTxAmount = 200000 * 10 ** 6 * 10 ** 9 (SHARK.sol#671)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
SHARK._decimals (SHARK.sol#658) should be constant SHARK._name (SHARK.sol#656) should be constant SHARK._symbol (SHARK.sol#657) should be constant SHARK._tTotal (SHARK.sol#653) should be constant SHARK.numTokensSellToAddToLiquidity (SHARK.sol#672) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that could-be-declared-constant
INFO:Detectors:
renounceOwnership() should be declared external:
- Ownable.renounceOwnership() (SHARK.sol#423-426)
transferOwnership(address) should be declared external:
- Ownable.transferOwnership(address) (SHARK.sol#431-435)
geUnlockTime() should be declared external:

- Ownable.geUnlockTime() (SHARK.sol#436-438)
lock(uint256) should be declared external:
- Ownable.lock(uint256) (SHARK.sol#440-445)
unlock() should be declared external:
- Ownable.unlock() (SHARK.sol#448-453)
name() should be declared external:
- SHARK.name() (SHARK.sol#704-706) symbol()
should be declared external:
- SHARK.symbol() (SHARK.sol#707-709) decimals()
should be declared external:
- SHARK.decimals() (SHARK.sol#710-712)
totalSupply() should be declared external:
- SHARK.totalSupply() (SHARK.sol#713-715)
transfer(address,uint256) should be declared external:
- SHARK.transfer(address,uint256) (SHARK.sol#720-723)
allowance(address,address) should be declared external:
- SHARK.allowance(address,address) (SHARK.sol#724-
726) approve(address,uint256) should be declared external:
- SHARK.approve(address,uint256) (SHARK.sol#727-730)
transferFrom(address,address,uint256) should be declared external:
- SHARK.transferFrom(address,address,uint256) (SHARK.sol#731-
735) increaseAllowance(address,uint256) should be declared external:
- SHARK.increaseAllowance(address,uint256) (SHARK.sol#736-739)
decreaseAllowance(address,uint256) should be declared external:
- SHARK.decreaseAllowance(address,uint256) (SHARK.sol#740-
743) isExcludedFromReward(address) should be declared external:
- SHARK.isExcludedFromReward(address) (SHARK.sol#744-
746) totalFees() should be declared external:
- SHARK.totalFees() (SHARK.sol#747-749)
deliver(uint256) should be declared external:
- SHARK.deliver(uint256) (SHARK.sol#750-757)
reflectionFromToken(uint256,bool) should be declared external:
- SHARK.reflectionFromToken(uint256,bool) (SHARK.sol#758-767)
excludeFromReward(address) should be declared external:
- SHARK.excludeFromReward(address) (SHARK.sol#773-
781) excludeFromFee(address) should be declared external:
- SHARK.excludeFromFee(address) (SHARK.sol#805-807)
includeInFee(address) should be declared external:
- SHARK.includeInFee(address) (SHARK.sol#809-811)
setSwapAndLiquifyEnabled(bool) should be declared external:
- SHARK.setSwapAndLiquifyEnabled(bool) (SHARK.sol#826-829)
isExcludedFromFee(address) should be declared external:
- SHARK.isExcludedFromFee(address) (SHARK.sol#905-907) Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that
could-be-declared-external
INFO:Slither:SHARK.sol analyzed (10 contracts with 72 detectors), 105

result(s) found INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration root@xcv-ThinkPad-T410:/home/xcv/tempSlither#

Email: audit@EEherAuEhority.io