QuillAudits

Audit Report
April, 2021

SALVATION

Contents

Overview

Salvation

Salvation aspires to drive blockchain adoption by introducing disruptive, trustworthy projects to investors that solve real-world issues and generate value.

Salvation enables VC Strategists to propose ecosystem upgrades to suit all market opportunities.

Salvation utilizes NFT's technology to verifiably create a whitelist and presale lists, rewarding long-term community members.

Salvation (SLVN) Tokenomics:

Untitled

Title	Tags	Address
TICKER	SLVN	
Max Supply	150,000,000,000	
Holders Tax	100 (1%)	
Burn Tax	300 (3%)	
Liquidity Fee	400 (4%)	
Max Transaction	150,000,000,000	
SLVN To Sell To Create LP	25,000	
Approval Delay	1 Day	
Strategist Address	TBC	
Team Tokens	30,000,000,000	Timelock contract
Marketing Tokens	12,000,000,000	
Presale Tokens	78,000,000,000	
Listing Tokens	30,000,000,000	
Domain	salvation.finance	

+ New

Contract: Salvation.sol
*The bugs have been fixed in newer releases/versions of the contract

Description Report: Salvation.md

Scope of Audit

The scope of this audit was to analyze Salvation.sol smart contract's codebase for quality, security, and correctness.

Checked Vulnerabilities

We have scanned the smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that we considered:

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- Exception Disorder
- Gasless Send
- Use of tx.origin
- Malicious libraries
- Compiler version not fixed
- Address hardcoded
- Divide before multiply
- Integer overflow/underflow
- ERC20 transfer() does not return boolean
- ERC20 approve() race
- Dangerous strict equalities
- Tautology or contradiction
- Return values of low-level
- calls
- Missing Zero Address
- Validation
- Private modifier
- Revert/require functions
- Using block.timestamp
- Multiple Sends
- Using SHA3
- Using suicide
- Using throw
- Using inline assembly

Techniques and Methods

Throughout the audit of smart contracts care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

Structural Analysis
In this step we have analyzed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.
SmartCheck.

Static Analysis
Static Analysis of Smart Contracts was done to identify contract vulnerabilities. In this step a series of automated tools are used to test security of smart contracts.

Code Review / Manual Analysis
Manual Analysis or review of code was done to identify new vulnerability or verify the vulnerabilities found during the static analysis. Contracts were completely manually analyzed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of automated analysis were manually verified.

Gas Consumption
In this step we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit
Mythril, Slither, SmartCheck, Surya, Solhint.

Issue Categories

Every issue in this report has been assigned with a severity level. There are four levels of severity and each of them has been explained below.

High severity issues

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality and we recommend these issues to be fixed before moving to a live environment.

Medium level severity issues

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems and they should still be fixed.

Low level severity issues

Low level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are severity four issues which indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Number of issues per severity

	High	Medium	Low	Informational
Open	0	0	5	2
Closed	0	1	2	9

Issues Found – Code Review / Manual Testing

High severity issues

None.

Medium severity issues

1. **[FIXED] No Range Defined for setting Burn/Tax/Liquidity Fees.**

   Inconsistent values can make the calculations/transactions revert on underflow

   Attack Scenario #1

   takeFee = true

1. Included the sender and recipient in fees,
   _isExcludedFromFee(sender) = False
   _isExcludedFromFee(recipient) = False
   Sender is the Owner and Recipient is another test account, which is also a strategist
2. This is done to make the condition at #L1333 false, and not to change the takeFee = false
3. Strategist proposes a New Burn Fee/Tax Fee/Liquidity Fee
4. Since a strategist has the power to approve its own proposals
5. Upgraded the Burn Tax after the approval Delay
6. Set Burn Fee to **5000000**
7. Tried transferring 5000 to test account from Owner account
8. As both the sender and receiver are excluded from fees, it will initiate a standard token transfer
9. _transferStandard function will fetch values _getValues(tAmount)
10. Which in return will call _getTValues(tAmount);
11. Now the values calculated will be:
    tFee = calculateTaxFee(tAmount)
    tFee = _amount.mul(_taxFee).div(10**(_feeDecimal + 2));
          = (5000*100)/10**4
          =50

    Similarly,
    tLiquidity   = (5000*400)/10**4

tBurn = (5000*5000000)/104**
=2500000

Now while calculating
[#L1205] **tTransferAmount = tAmount.sub(tFee).sub(tLiquidity).sub(tBurn);**
= 5000 - 50 - 200 - 2500000
= -2495250

Which will revert on Underflow

Set the value back to original, and transaction will happen

Attack Scenario #2

1. The same scenario is possible if the owner sets an Inconsistent value for any of the taxes

Low Severity Issues

1. [#L5] Old solidity compiler used. Use the latest compiler to avoid bugs found in old compilers

2. [#L5] Floating Pragma: Different Pragma Directives have been used

3. Reentrancy Checks as mentioned in the Slither Report(provided in the automated tests section)

   The best practices to avoid Reentrancy weaknesses are:

   -Make sure all internal state changes are performed before the call is executed. This is known as the Checks-Effects-Interactions pattern https://docs.soliditylang.org/en/latest/security-considerations.html#use-the-checks-effects-interactions-pattern

   -Use a reentrancy lock (ie. OpenZeppelin's ReentrancyGuard. https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/security/ReentrancyGuard.sol

4. [#L535, 1511, 1526, 1541] Comparison with now/block.timestamp. Avoid using them they can be manipulated by miners

5. **[Fixed]Local Variable Shadowing:**
   [#L1398]function _tokenTransfer(address sender, address recipient, uint256 amount, bool **takeFee**) shadows local variable **takeFee**[#L948]

6. **[FIXED]**[#L1586 - 1589] function **setLiquidityAddress()** missing zero address validation

7. **ERC20 approve() race**
   The standard ERC20 implementation contains a widely-known racing condition in its approve function, wherein a spender is able to witness the token owner broadcast a transaction altering their approval, and quickly sign and broadcast a transaction using transferFrom to move the current approved amount from the owner's balance to the spender. If the spender's transaction is validated before the owner's, the spender is able to spend their entire approval amount twice.

   Possible Solution: make sure to create user interfaces in such a way that they set the allowance first to 0 before setting it to another value for the same spender

   Reference:

- https://docs.google.com/document/d/1YLPtQxZu1UAvO9cZ1O2RPXBbT0mooh4DYKjA_jp-RLM/edit

- https://medium.com/mycrypto/bad-actors-abusing-erc20-approval-to-steal-your-tokens-c0407b7f7c7c

- https://eips.ethereum.org/EIPS/eip-20

Informational

1. **[FIXED]**[#L522-527] A check can be added to not allow a value for time passed to **function lock(),** that can lock the contract for a long time

2. **[FIXED]**[#L535]: require(now > _lockTime, "**Contract is locked until 7** days");
   The error statement will always mark 7 days, irrespective of whatever the locktime is set

3. **[FIXED]**[#L1005] approvalDelay = 60; It should be **1 day;**

4. [#L1129-1140] function **includeInReward()**, [#L1223-1236] function **_getCurrentSupply()** consuming Extra Gas: **.length** of non-memory array is used in the condition for loop. In this case, every iteration of the loop consumes extra gas. Holding its value in a local variable is more gas efficient. Also if array.length is large enough, the function can exceed the block gas limit. So, it is recommended to avoid loops with an unknown number of steps

5. **[FIXED]**[#L1130] It should be included instead of excluded

6. **[FIXED]**[#L1163, 1167] Already Excluded/Included checks can be added in the functions

7. **[FIXED]**[#L1172] Event can be added for fallback function

8. [#L1353] ETH --> HATE Swap. Did you mean ETH --> SLVTN?

9. **[FIXED]**[#L1571] **emit** keyword missing

10. **[FIXED]**[#L1601, 1605, 1609, 1613] Events can be added for the setter functions

11. **[FIXED]**A range can be defined for setter functions changing the Constant Values, to avoid setting an Inconsistent Value

Suggestions

- [#L1119-1127] function **excludeFromReward()** add checks to not exclude uniswap router, this contract's address, and ownerWallet
  Ref:
  require(account != 0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D, We can not exclude Uniswap router.');

  require(account != address(this), We can not exclude contract self.');

  require(account != ownerWallet, 'We can not exclude owner wallet.');

Gas Optimization- public functions that are never called by the contract could be declared external to save gas.

```
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (Salvation.sol#499-502)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (Salvation.sol#508-515)
getUnlockTime() should be declared external:
        - Ownable.getUnlockTime() (Salvation.sol#517-519)
lock(uint256) should be declared external:
        - Ownable.lock(uint256) (Salvation.sol#522-527)
unlock() should be declared external:
        - Ownable.unlock() (Salvation.sol#530-538)
name() should be declared external:
        - Salvation.name() (Salvation.sol#1016-1018)
symbol() should be declared external:
        - Salvation.symbol() (Salvation.sol#1020-1022)
decimals() should be declared external:
        - Salvation.decimals() (Salvation.sol#1024-1026)
totalSupply() should be declared external:
        - Salvation.totalSupply() (Salvation.sol#1028-1030)
transfer(address,uint256) should be declared external:
        - Salvation.transfer(address,uint256) (Salvation.sol#1037-1040)
allowance(address,address) should be declared external:
        - Salvation.allowance(address,address) (Salvation.sol#1042-1045)
approve(address,uint256) should be declared external:
        - Salvation.approve(address,uint256) (Salvation.sol#1047-1051)
transferFrom(address,address,uint256) should be declared external:
        - Salvation.transferFrom(address,address,uint256) (Salvation.sol#1053-1064)
increaseAllowance(address,uint256) should be declared external:
        - Salvation.increaseAllowance(address,uint256) (Salvation.sol#1066-1073)
decreaseAllowance(address,uint256) should be declared external:
        - Salvation.decreaseAllowance(address,uint256) (Salvation.sol#1075-1085)
isExcludedFromReward(address) should be declared external:
        - Salvation.isExcludedFromReward(address) (Salvation.sol#1087-1089)
totalFees() should be declared external:
        - Salvation.totalFees() (Salvation.sol#1091-1093)
totalBurn() should be declared external:
        - Salvation.totalBurn() (Salvation.sol#1095-1097)
reflectionFromToken(uint256,bool) should be declared external:
        - Salvation.reflectionFromToken(uint256,bool) (Salvation.sol#1099-1108)
excludeFromReward(address) should be declared external:
        - Salvation.excludeFromReward(address) (Salvation.sol#1119-1127)
excludeFromFee(address) should be declared external:
        - Salvation.excludeFromFee(address) (Salvation.sol#1163-1165)
includeInFee(address) should be declared external:
        - Salvation.includeInFee(address) (Salvation.sol#1167-1169)
proposeBurnTax(uint256) should be declared external:
        - Salvation.proposeBurnTax(uint256) (Salvation.sol#1480-1485)
proposeHoldersTax(uint256) should be declared external:
        - Salvation.proposeHoldersTax(uint256) (Salvation.sol#1487-1495)
proposeLiquidityTax(uint256) should be declared external:
        - Salvation.proposeLiquidityTax(uint256) (Salvation.sol#1497-1505)
upgradeBurnStrat() should be declared external:
        - Salvation.upgradeBurnStrat() (Salvation.sol#1507-1520)
upgradeHoldersStrat() should be declared external:
        - Salvation.upgradeHoldersStrat() (Salvation.sol#1522-1535)
upgradeLiquidityStrat() should be declared external:
        - Salvation.upgradeLiquidityStrat() (Salvation.sol#1537-1550)
setNumTokensSellToAddToLiquidity(uint256) should be declared external:
        - Salvation.setNumTokensSellToAddToLiquidity(uint256) (Salvation.sol#1554-1559)
setMaxTxPercent(uint256) should be declared external:
        - Salvation.setMaxTxPercent(uint256) (Salvation.sol#1561-1566)
setTradingHalted(bool) should be declared external:
        - Salvation.setTradingHalted(bool) (Salvation.sol#1568-1572)
setSwapAndLiquifyEnabled(bool) should be declared external:
        - Salvation.setSwapAndLiquifyEnabled(bool) (Salvation.sol#1576-1579)
setStrategist(address) should be declared external:
        - Salvation.setStrategist(address) (Salvation.sol#1581-1584)
setLiquidityAddress(address) should be declared external:
        - Salvation.setLiquidityAddress(address) (Salvation.sol#1586-1589)
setTakeFee(bool) should be declared external:
        - Salvation.setTakeFee(bool) (Salvation.sol#1591-1594)
```

Automated Testing

Slither

Severity: HIGH

```
Reentrancy in Salvation._transfer(address,address,uint256) (Salvation.sol#1293-1339):
	External calls:
	- swapAndLiquify(contractTokenBalance) (Salvation.sol#1329)
		- uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,liquidityAddress,block.timestamp) (Salvation.sol#1387-1394)
		- uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Salvation.sol#1373-1379)
	External calls sending eth:
	- swapAndLiquify(contractTokenBalance) (Salvation.sol#1329)
		- uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,liquidityAddress,block.timestamp) (Salvation.sol#1387-1394)
	State variables written after the call(s):
	- _tokenTransfer(from,to,amount,takeFee) (Salvation.sol#1338)
		- _rOwned[address(this)] = _rOwned[address(this)].add(rLiquidity) (Salvation.sol#1241)
		- _rOwned[sender] = _rOwned[sender].sub(rAmount) (Salvation.sol#1428)
		- _rOwned[sender] = _rOwned[sender].sub(rAmount) (Salvation.sol#1447)
		- _rOwned[sender] = _rOwned[sender].sub(rAmount) (Salvation.sol#1155)
		- _rOwned[sender] = _rOwned[sender].sub(rAmount) (Salvation.sol#1468)
		- _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (Salvation.sol#1429)
		- _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (Salvation.sol#1449)
		- _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (Salvation.sol#1469)
		- _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (Salvation.sol#1157)
	- _tokenTransfer(from,to,amount,takeFee) (Salvation.sol#1338)
		- _rTotal = _rTotal.sub(rFee).sub(rBurn) (Salvation.sol#1175)
	- _tokenTransfer(from,to,amount,takeFee) (Salvation.sol#1338)
		- _tOwned[address(this)] = _tOwned[address(this)].add(tLiquidity) (Salvation.sol#1243)
		- _tOwned[sender] = _tOwned[sender].sub(tAmount) (Salvation.sol#1467)
		- _tOwned[sender] = _tOwned[sender].sub(tAmount) (Salvation.sol#1154)
		- _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount) (Salvation.sol#1448)
		- _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount) (Salvation.sol#1156)
	- _tokenTransfer(from,to,amount,takeFee) (Salvation.sol#1338)
		- _tTotal = _tTotal.sub(tBurn) (Salvation.sol#1178)
```

Severity: LOW

```
Reentrancy in Salvation._transfer(address,address,uint256) (Salvation.sol#1293-1339):
	External calls:
	- swapAndLiquify(contractTokenBalance) (Salvation.sol#1329)
		- uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,liquidityAddress,block.timestamp) (Salvation.sol#1387-1394)
		- uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Salvation.sol#1373-1379)
	External calls sending eth:
	- swapAndLiquify(contractTokenBalance) (Salvation.sol#1329)
		- uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,liquidityAddress,block.timestamp) (Salvation.sol#1387-1394)
	State variables written after the call(s):
	- _tokenTransfer(from,to,amount,takeFee) (Salvation.sol#1338)
		- _burnFee = _previousBurnFee (Salvation.sol#1272)
		- _burnFee = 0 (Salvation.sol#1266)
	- _tokenTransfer(from,to,amount,takeFee) (Salvation.sol#1338)
		- _liquidityFee = _previousLiquidityFee (Salvation.sol#1273)
		- _liquidityFee = 0 (Salvation.sol#1267)
	- _tokenTransfer(from,to,amount,takeFee) (Salvation.sol#1338)
		- _previousBurnFee = _burnFee (Salvation.sol#1262)
	- _tokenTransfer(from,to,amount,takeFee) (Salvation.sol#1338)
		- _previousLiquidityFee = _liquidityFee (Salvation.sol#1263)
	- _tokenTransfer(from,to,amount,takeFee) (Salvation.sol#1338)
		- _previousTaxFee = _taxFee (Salvation.sol#1261)
	- _tokenTransfer(from,to,amount,takeFee) (Salvation.sol#1338)
		- _tBurnTotal = _tBurnTotal.add(tBurn) (Salvation.sol#1176)
	- _tokenTransfer(from,to,amount,takeFee) (Salvation.sol#1338)
		- _tFeeTotal = _tFeeTotal.add(tFee) (Salvation.sol#1177)
	- _tokenTransfer(from,to,amount,takeFee) (Salvation.sol#1338)
		- _taxFee = _previousTaxFee (Salvation.sol#1271)
		- _taxFee = 0 (Salvation.sol#1265)
	- takeFee = false (Salvation.sol#1334)
Reentrancy in Salvation.constructor() (Salvation.sol#988-1014):
	External calls:
	- uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV2Router.WETH()) (Salvation.sol#995-996)
	State variables written after the call(s):
	- _isExcludedFromFee[owner()] = true (Salvation.sol#1002)
	- _isExcludedFromFee[address(this)] = true (Salvation.sol#1003)
	- approvalDelay = 60 (Salvation.sol#1005)
	- liquidityAddress = owner() (Salvation.sol#1006)
	- ownerWallet = _msgSender() (Salvation.sol#1011)
	- sellingHalted = true (Salvation.sol#1010)
	- takeFee = true (Salvation.sol#1008)
	- tradingHalted = true (Salvation.sol#1009)
	- uniswapV2Router = _uniswapV2Router (Salvation.sol#999)
Reentrancy in Salvation.swapAndLiquify(uint256) (Salvation.sol#1341-1362):
	External calls:
	- swapTokensForEth(half) (Salvation.sol#1353)
		- uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Salvation.sol#1373-1379)
	- addLiquidity(otherHalf,newBalance) (Salvation.sol#1359)
		- uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,liquidityAddress,block.timestamp) (Salvation.sol#1387-1394)
	External calls sending eth:
	- addLiquidity(otherHalf,newBalance) (Salvation.sol#1359)
		- uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,liquidityAddress,block.timestamp) (Salvation.sol#1387-1394)
	State variables written after the call(s):
	- addLiquidity(otherHalf,newBalance) (Salvation.sol#1359)
		- _allowances[owner][spender] = amount (Salvation.sol#1289)
Reentrancy in Salvation.transferFrom(address,address,uint256) (Salvation.sol#1053-1064):
	External calls:
	- _transfer(sender,recipient,amount) (Salvation.sol#1054)
		- uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,liquidityAddress,block.timestamp) (Salvation.sol#1387-1394)
		- uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Salvation.sol#1373-1379)
	External calls sending eth:
	- _transfer(sender,recipient,amount) (Salvation.sol#1054)
		- uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,liquidityAddress,block.timestamp) (Salvation.sol#1387-1394)
	State variables written after the call(s):
	- _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20: transfer amount exceeds allowance)) (Salvation.sol#1055-1062)
		- _allowances[owner][spender] = amount (Salvation.sol#1289)
```

```
Reentrancy in Salvation._transfer(address,address,uint256) (Salvation.sol#1293-1339):
        External calls:
        - swapAndLiquify(contractTokenBalance) (Salvation.sol#1329)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,liquidityAddress,block.timestamp) (Salvation.sol#1387-1394)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Salvation.sol#1373-1379)
        External calls sending eth:
        - swapAndLiquify(contractTokenBalance) (Salvation.sol#1329)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,liquidityAddress,block.timestamp) (Salvation.sol#1387-1394)
        Event emitted after the call(s):
        - Transfer(sender,recipient,tTransferAmount) (Salvation.sol#1432)
                - _tokenTransfer(from,to,amount,takeFee) (Salvation.sol#1338)
        - Transfer(sender,recipient,tTransferAmount) (Salvation.sol#1452)
                - _tokenTransfer(from,to,amount,takeFee) (Salvation.sol#1338)
        - Transfer(sender,recipient,tTransferAmount) (Salvation.sol#1472)
                - _tokenTransfer(from,to,amount,takeFee) (Salvation.sol#1338)
        - Transfer(sender,recipient,tTransferAmount) (Salvation.sol#1160)
                - _tokenTransfer(from,to,amount,takeFee) (Salvation.sol#1338)
Reentrancy in Salvation.constructor() (Salvation.sol#988-1014):
        External calls:
        - uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV2Router.WETH()) (Salvation.sol#995-996)
        Event emitted after the call(s):
        - Transfer(address(0),_msgSender(),_tTotal) (Salvation.sol#1013)
Reentrancy in Salvation.swapAndLiquify(uint256) (Salvation.sol#1341-1362):
        External calls:
        - swapTokensForEth(half) (Salvation.sol#1353)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Salvation.sol#1373-1379)
        - addLiquidity(otherHalf,newBalance) (Salvation.sol#1359)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,liquidityAddress,block.timestamp) (Salvation.sol#1387-1394)
        External calls sending eth:
        - addLiquidity(otherHalf,newBalance) (Salvation.sol#1359)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,liquidityAddress,block.timestamp) (Salvation.sol#1387-1394)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (Salvation.sol#1290)
                - addLiquidity(otherHalf,newBalance) (Salvation.sol#1359)
        - SwapAndLiquify(half,newBalance,otherHalf) (Salvation.sol#1361)
Reentrancy in Salvation.transferFrom(address,address,uint256) (Salvation.sol#1053-1064):
        External calls:
        - _transfer(sender,recipient,amount) (Salvation.sol#1054)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,liquidityAddress,block.timestamp) (Salvation.sol#1387-1394)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Salvation.sol#1373-1379)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (Salvation.sol#1054)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,liquidityAddress,block.timestamp) (Salvation.sol#1387-1394)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (Salvation.sol#1290)
                - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20: transfer amount exceeds allowance)) (Salvation.sol#1055-1062)
```

Smartcheck

Smartcheck didn’t detect any high severity issues

Solhint

```
  263:2  error  Line length must be no more than 120 but current length is 132  max-line-length
  317:2  error  Line length must be no more than 120 but current length is 160  max-line-length
  342:2  error  Line length must be no more than 120 but current length is 156  max-line-length
 1181:2  error  Line length must be no more than 120 but current length is 127  max-line-length
 1209:2  error  Line length must be no more than 120 but current length is 162  max-line-length
 1301:2  error  Line length must be no more than 120 but current length is 142  max-line-length

✖ 6 problems (6 errors, 0 warnings)
```

Functional Testing

Functional Testing was done by deploying contract on Kovan Testnet

Test 1: Contract: 0x40ee888a25cE26F325836ABFe7F4CF7C2184CA99

Test 2: Contract: 0x35d27aBeDAc3665c9Ce53aDa486ef435442D2320

Disclaimer

The audit does not give any warranties on the security of the code. One audit cannot be considered enough. We always recommend proceeding with several independent audits and a public bug bounty program to ensure the security of the code. Besides, a security audit, please don’t consider this report as investment advice.

Closing Summary

Several issues of medium and low severity have been reported during the audit, out of which, most of them have been fixed. Some suggestions have also been made to improve the code quality and gas optimization. There were NO critical or major issues found that can break the intended behavior.

SALVATION

QuillAudits

Canada, India, Singapore and United Kingdom

audits.quillhash.com

hello@quillhash.com