CYSRO

Cyber Security

SMART CONTRACT SECURITY AUDIT

Salsa Valley

Table of Contents

Project Summary

Project name	SALSA VALLEY
Platform	Ethereum
Language	Solidity
Contract address	salsa.sol
Repository	NA
Contract owner address	NA
Deployers contract address	NA
Decimal	NA
Total supply	NA
Website	NA
Social media	NA
Audit methodology	Whitebox Testing
Delivery Date	July 17, 2021

Introduction

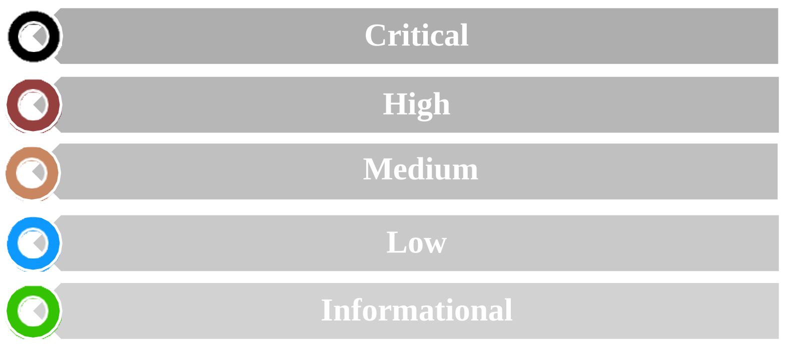
Given the opportunity to review SALSA VALLEY Project's smart contract source code, we in the report outline our systematic approach to evaluate potential security issues in the smart contract implementation, expose possible semantic inconsistencies between smart contract code and design document, and provide additional suggestions or recommendations for improvement. Our results show that the given version of smart contracts is ready to launch after resolving the mentioned issues, there are no critical or high issues found related to business logic, security or performance.

Scope of work

The files that needed to be evaluated for the security assessment were given to us by the Team. The files listed below were used for this audit. Other files and contracts that are not listed here are not audited by us hence we will not be responsible for any security issues caused by those contracts.

	File	Checksum
1	salsa.sol	NA

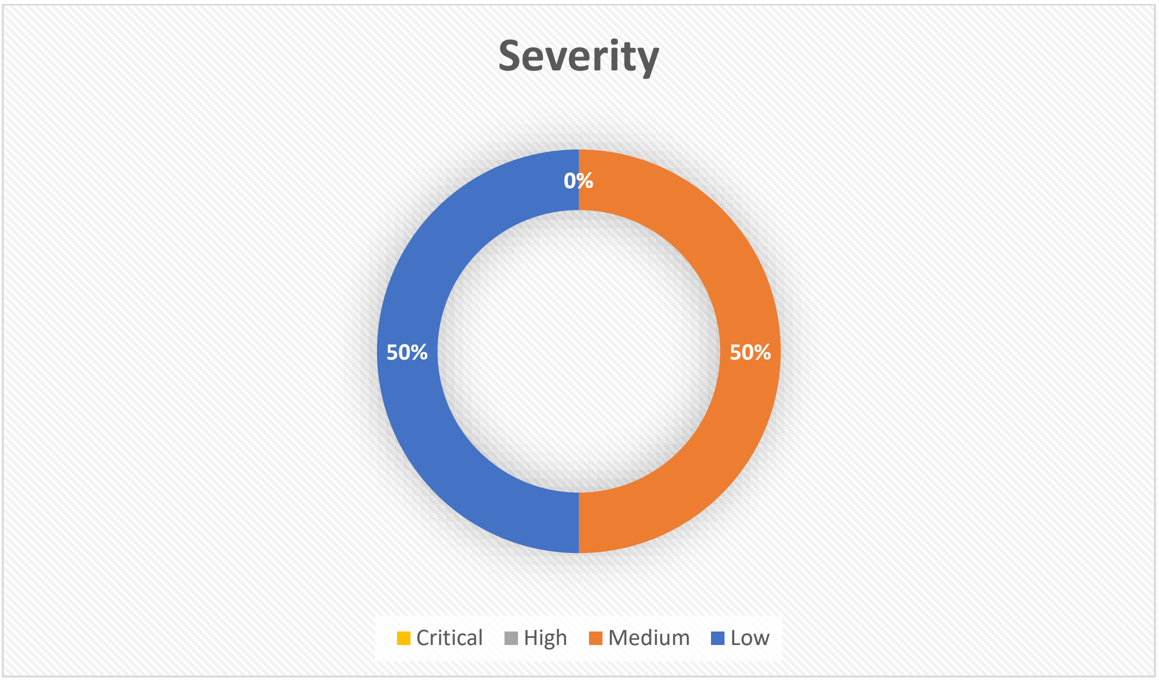
Vulnerability severity information



Level	Description
Critical	Critical severity vulnerabilities will have a significant effect on the security of the DeFi project, and it is strongly recommended to fix the critical vulnerabilities.
High	High severity vulnerabilities will affect the normal operation of the DeFi project. It is strongly recommended to fix high-risk vulnerabilities.
Medium	Medium severity vulnerability will affect the operation of the DeFi project. It is recommended to fix medium-risk vulnerabilities.
Low	Low severity vulnerabilities may affect the operation of the DeFi project in certain scenarios. It is suggested that the project party should evaluate and consider whether these vulnerabilities need to be fixed.
Informational	There are safety risks theoretically, but it is extremely difficult to reproduce in engineering.

Findings

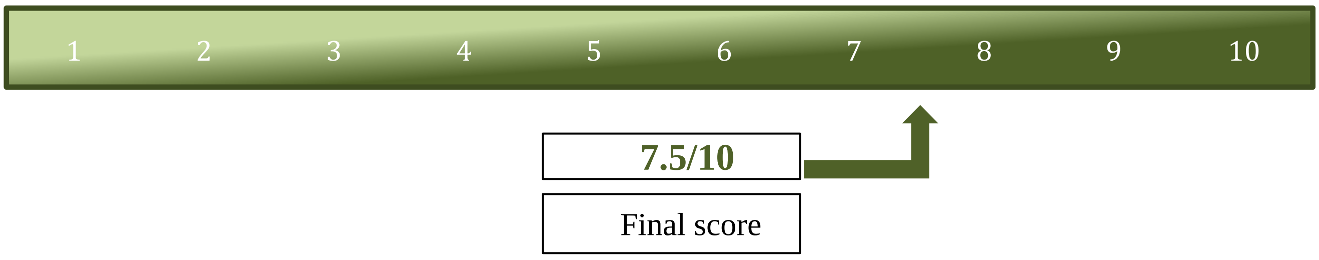
Total issues: 2



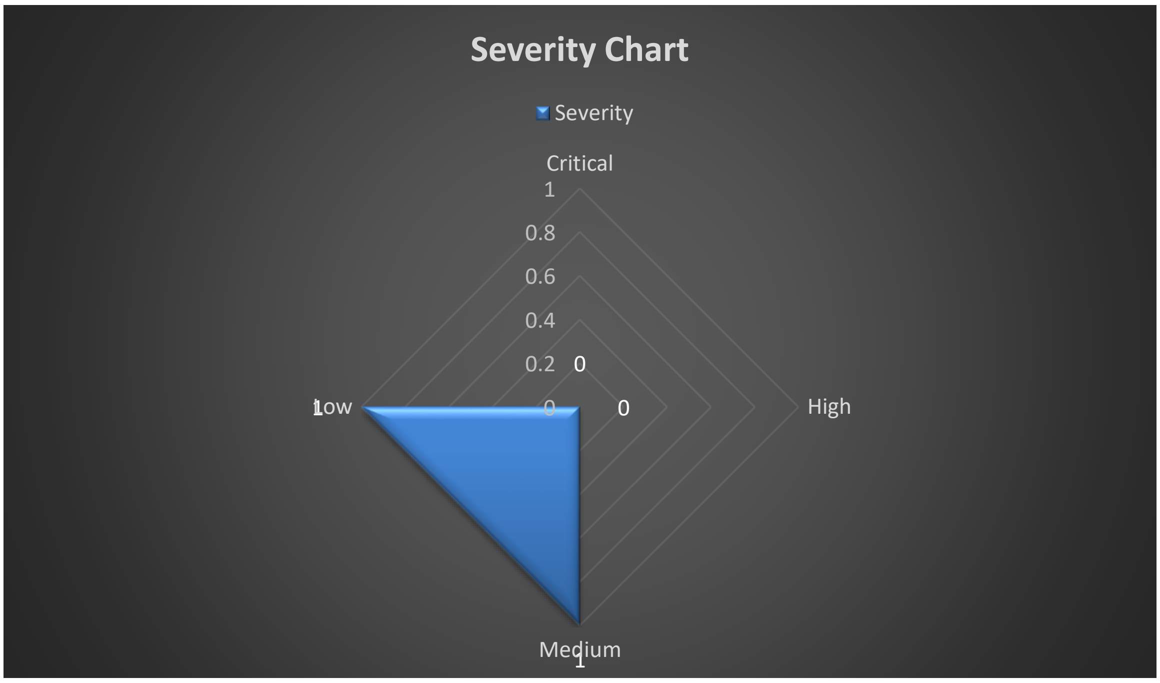
Critical	High	Medium	Low	Informational
0	0	1	1	**0**

Security Score

As a result of the audit, the code contains no major issues. Therefore, the security score is 7.5/10.



Severity chart



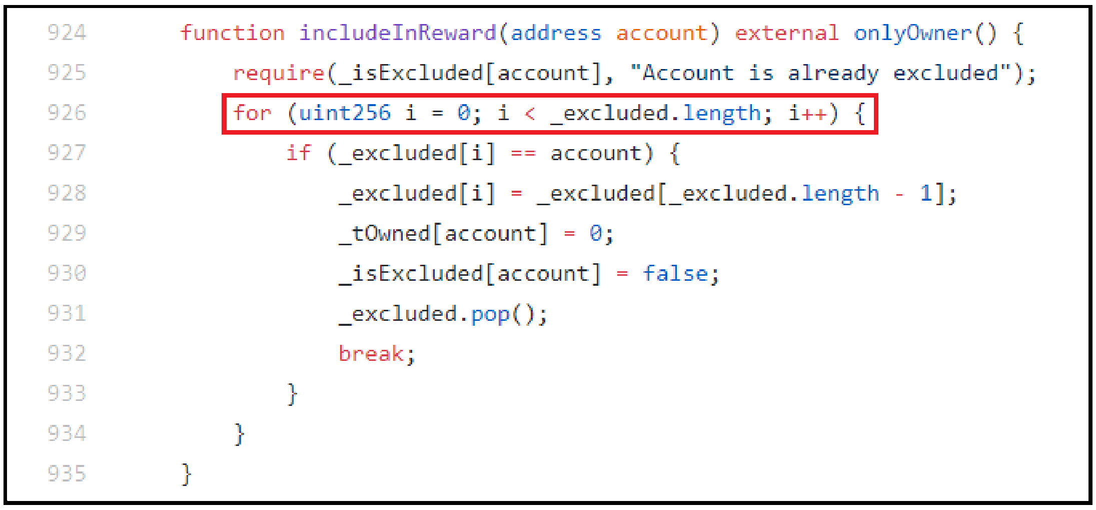
We have so far identified that there are potential issues with severity of 0 Critical, 0 High, 1 Medium, and 1 Low. Overall, these smart contracts are well-designed and engineered.

1. DoS With Block Gas Limit

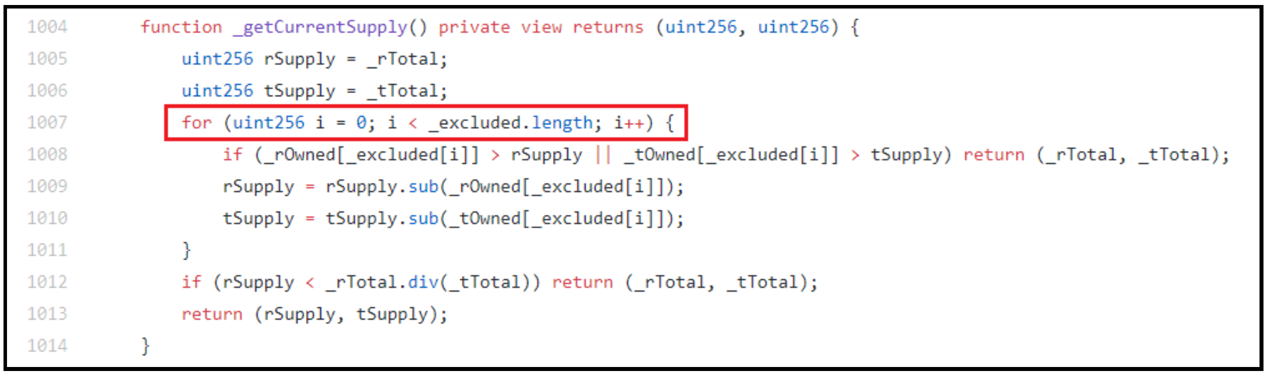
Severity	Location	Classification	Status
Medium	Salsa.sol		Open

Description

Loop over unbounded data structure. Gas consumption in function "includeInReward" in contract "Salsa" depends on the size of data structures or values that may grow unboundedly. If the data structure grows too large, the gas required to execute the code will exceed the block gas limit, effectively causing a denial-of-service condition. Consider that an attacker might attempt to cause this condition on purpose.



```
    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;
            }
        }
    }
```



```
    function _getCurrentSupply() private view returns (uint256, uint256) {
        uint256 rSupply = _rTotal;
        uint256 tSupply = _tTotal;
        for (uint256 i = 0; i < _excluded.length; i++) {
            if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) return (_rTotal, _tTotal);
            rSupply = rSupply.sub(_rOwned[_excluded[i]]);
            tSupply = tSupply.sub(_tOwned[_excluded[i]]);
        }
        if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
        return (rSupply, tSupply);
    }
```

Relationships

CWE-400: Uncontrolled Resource Consumption

Remediations

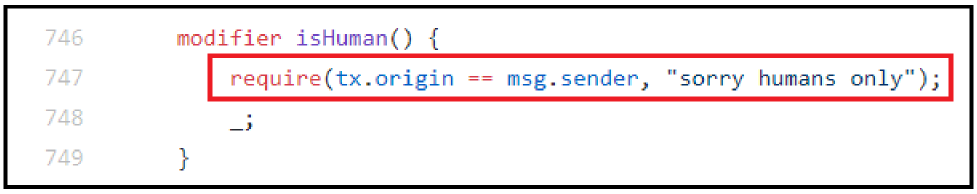
Caution is advised when you expect to have large arrays that grow over time. Actions that require looping across the entire data structure should be avoided. If you absolutely must loop over an array of unknown size, then you should plan for it to potentially take multiple blocks, and therefore require multiple transactions.

2. Authorization through tx.origin

Severity	Location	Classification	Status
Low	Salsa.sol		**Open**

Description

Use of "tx.origin" as a part of authorization control. The tx.origin environment variable has been found to influence a control flow decision. Note that using "tx.origin" as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use "msg.sender" instead.



```
    modifier isHuman() {
        require(tx.origin == msg.sender, "sorry humans only");
        _;
    }
```

Relationships

CWE-477: Use of Obsolete Function

Remediations

tx.origin should not be used for authorization. Use msg.sender instead.

Function overview

Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
IBEP20	Interface			
	totalSupply	External		NO
	balanceOf	External		NO
	transfer	External	●	NO
	allowance	External		NO
	approve	External	●	NO
	transferFrom	External	●	NO
SafeMath	Library			
	add	Internal		
	sub	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	div	Internal		
	mod	Internal		
	mod	Internal		
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
Address	Library			
	isContract	Internal		
	sendValue	Internal	●	
	functionCall	Internal	●	
	functionCall	Internal	●	
	functionCallWithValue	Internal	●	
	functionCallWithValue	Internal	●	
	_functionCallWithValue	Private	●	
Ownable	Implementation	Context		
	<Constructor>	Internal	●	
	owner	Public		NO
	renounceOwnership	Public	●	onlyOwner
	transferOwnership	Public	●	onlyOwner
	geUnlockTime	Public		NO
	lock	Public	●	onlyOwner
	unlock	Public	●	NO
IPancakeFactory	Interface			
	feeTo	External		NO
	feeToSetter	External		NO
	getPair	External		NO

	allPairs	External		NO
	allPairsLength	External		NO
	createPair	External	🛑	NO
	setFeeTo	External	🛑	NO
	setFeeToSetter	External	🛑	NO
IPancakePair	Interface			
	name	External		NO
	symbol	External		NO
	decimals	External		NO
	totalSupply	External		NO
	balanceOf	External		NO
	allowance	External		NO
	approve	External	🛑	NO
	transfer	External	🛑	NO
	transferFrom	External	🛑	NO
	DOMAIN_SEPARATOR	External		NO
	PERMIT_TYPEHASH	External		NO
	nonces	External		NO
	permit	External	🛑	NO
	MINIMUM_LIQUIDITY	External		NO
	factory	External		NO
	token0	External		NO
	token1	External		NO
	getReserves	External		NO
	price0CumulativeLast	External		NO
	price1CumulativeLast	External		NO
	kLast	External		NO
	mint	External	🛑	NO
	swap	External	🛑	NO
	skim	External	🛑	NO
	sync	External	🛑	NO
	initialize	External	🛑	NO
IPancakeRouter01	Interface			
	factory	External		NO
	WETH	External		NO
	addLiquidity	External	🛑	NO
	addLiquidityETH	External	💵	NO
	removeLiquidity	External	🛑	NO
	removeLiquidityETH	External	🛑	NO
	removeLiquidityWithPermit	External	🛑	NO
	removeLiquidityETHWithPermit	External	🛑	NO
	swapExactTokensForTokens	External	🛑	NO
	swapTokensForExactTokens	External	🛑	NO
	swapExactETHForTokens	External	💵	NO
	swapTokensForExactETH	External	🛑	NO
	swapExactTokensForETH	External	🛑	NO
	swapETHForExactTokens	External	💵	NO

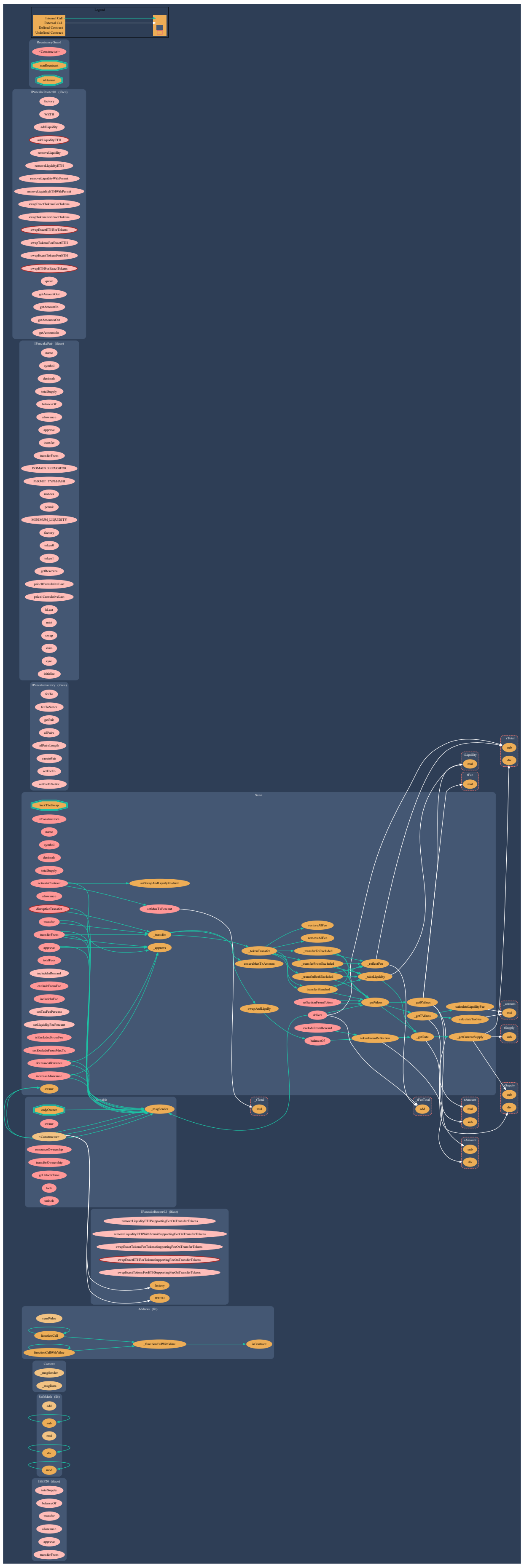
	quote	External		NO
	getAmountOut	External		NO
	getAmountIn	External		NO
	getAmountsOut	External		NO
	getAmountsIn	External		NO
IPancakeRouter02	Interface	IPancakeRouter01		
	removeLiquidityETHSupportingFeeOnTransferTokens	External	●	NO
	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External	●	NO
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	●	NO
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	💵	NO
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	●	NO
ReentrancyGuard	Implementation			
	<Constructor>	Public	●	NO
Salsa	Implementation	Context, IBEP20, Ownable, ReentrancyGuard		
	<Constructor>	Public	●	NO
	name	Public		NO
	symbol	Public		NO
	decimals	Public		NO
	totalSupply	Public		NO
	balanceOf	Public		NO
	transfer	Public	●	NO
	allowance	Public		NO
	approve	Public	●	NO
	transferFrom	Public	●	NO
	increaseAllowance	Public	●	NO
	decreaseAllowance	Public	●	NO
	totalFees	Public		NO
	deliver	Public	●	NO
	reflectionFromToken	Public		NO
	tokenFromReflection	Public		NO
	excludeFromReward	Public	●	onlyOwner
	includeInReward	External	●	onlyOwner
	_transferBothExcluded	Private	●	
	excludeFromFee	Public	●	onlyOwner
	includeInFee	Public	●	onlyOwner
	setTaxFeePercent	External	●	onlyOwner
	setLiquidityFeePercent	External	●	onlyOwner
	setSwapAndLiquifyEnabled	Public	●	onlyOwner

_reflectFee	Private	●	
_getValues	Private		
_getTValues	Private		
_getRValues	Private		
_getRate	Private		
_getCurrentSupply	Private		
_takeLiquidity	Private	●	
calculateTaxFee	Private		
calculateLiquidityFee	Private		
removeAllFee	Private	●	
restoreAllFee	Private	●	
isExcludedFromFee	Public		NO
_approve	Private	●	
_transfer	Private	●	
_tokenTransfer	Private	●	
_transferStandard	Private	●	
_transferToExcluded	Private	●	
_transferFromExcluded	Private	●	
setMaxTxPercent	Public	●	onlyOwner
setExcludeFromMaxTx	Public	●	onlyOwner
ensureMaxTxAmount	Private	●	
disruptiveTransfer	Public	💵	NO
swapAndLiquify	Private	●	
activateContract	Public	●	onlyOwner

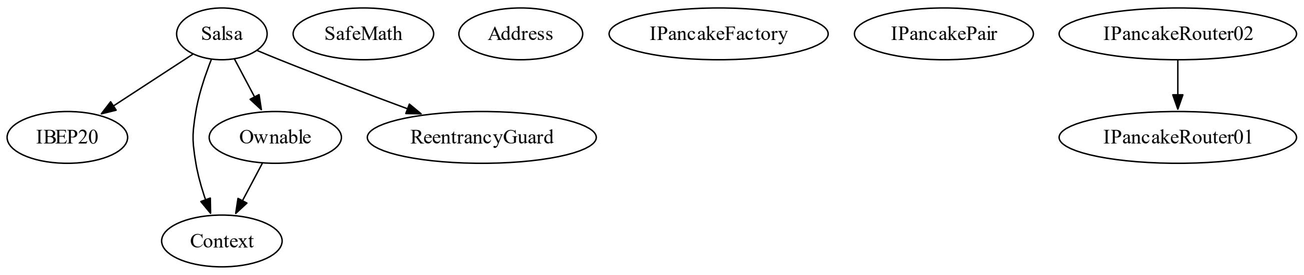
Where Symbol Meaning

- **Function can modify state** ■ **Function is payable**

Functional Flow diagram



Inheritance graph



Liquidity lock

Liquidity locked period	Status
NA	NA

Token Ownership renounced

Token ownership Renounced	Status
NA	NA

Deployers actions

Can the deployer/owner mint a new token?	Status
NA	NA

Can the deployer/owner blacklist any wallet from selling?	Status
NA	NA

Can deployer/owner lock or burn user funds?	Status
NA	NA

Can the deployer/owner pause the contract?	Status
NA	NA

Can the deployer/owner increase the fees?	Status
NA	NA

SWC Attacks

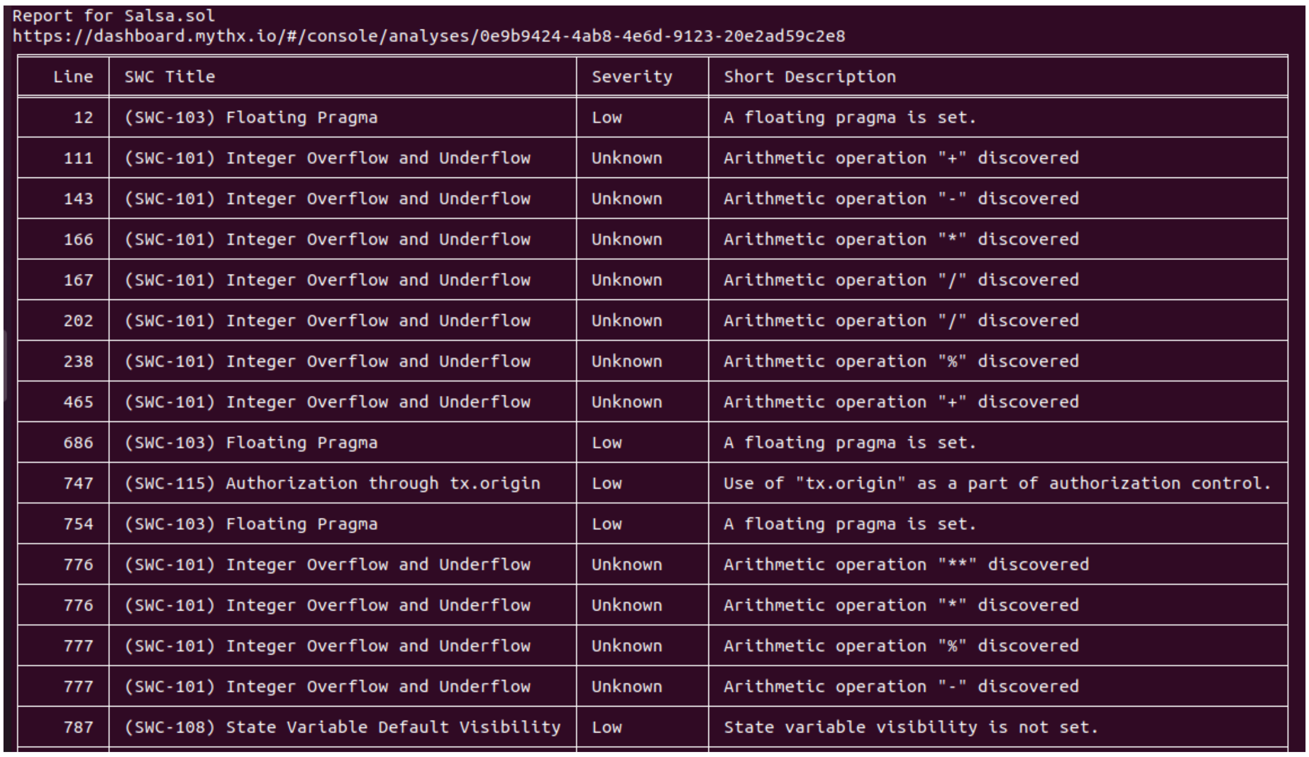
Line	SWC	Severity	Description	Status
Salsa.sol	128	Medium	Loop over unbounded data structure. Gas consumption in function "includeInReward" in contract "Salsa" depends on the size of data structures or values that may grow unboundedly. If the data structure grows too large, the gas required to execute the code will exceed the block gas limit, effectively causing a denial-of-service condition. Consider that an attacker might attempt to cause this condition on purpose.	Open
Salsa.sol	115	Low	Use of "tx.origin" as a part of authorization control. The tx.origin environment variable has been found to influence a control flow decision. Note that using "tx.origin" as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use "msg.sender" instead.	Open

Test Results

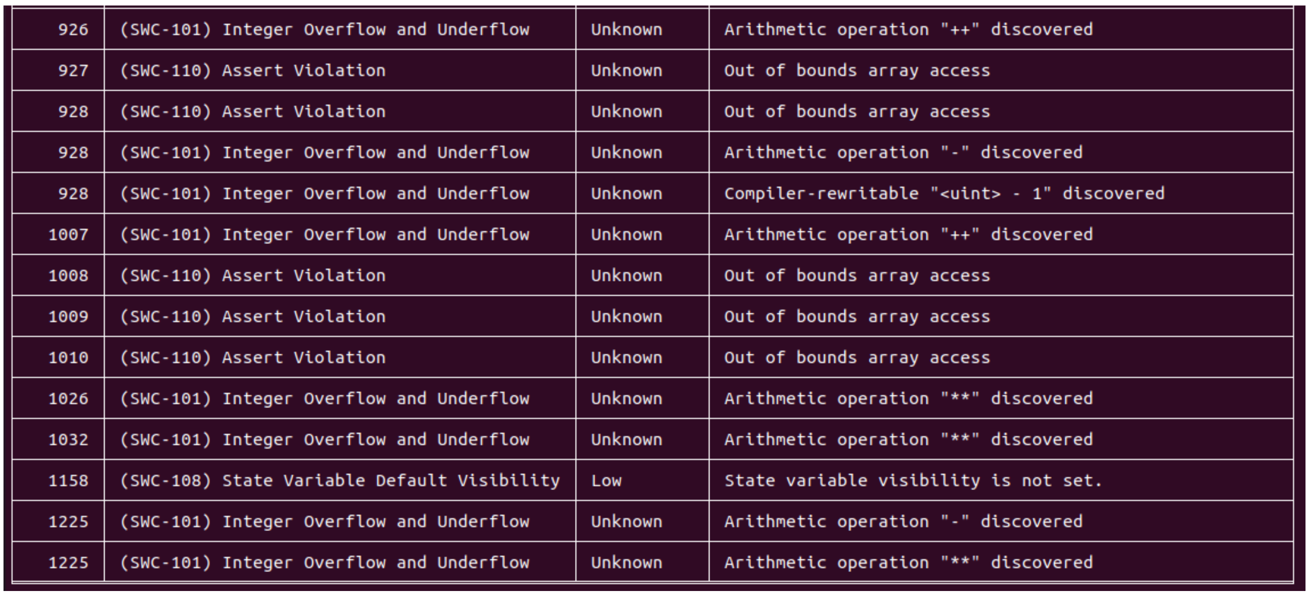
Slither results

NA

Mythx results

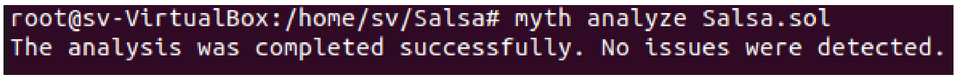


```
Report for Salsa.sol
https://dashboard.mythx.io/#/console/analyses/0e9b9424-4ab8-4e6d-9123-20e2ad59c2e8
| Line | SWC Title                                | Severity | Short Description                                         |
|------|------------------------------------------|----------|-----------------------------------------------------------|
|   12 | (SWC-103) Floating Pragma                | Low      | A floating pragma is set.                                 |
|  111 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "+" discovered                       |
|  143 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "-" discovered                       |
|  166 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "*" discovered                       |
|  167 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "/" discovered                       |
|  202 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "/" discovered                       |
|  238 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "%" discovered                       |
|  465 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "+" discovered                       |
|  686 | (SWC-103) Floating Pragma                | Low      | A floating pragma is set.                                 |
|  747 | (SWC-115) Authorization through tx.origin | Low     | Use of "tx.origin" as a part of authorization control.    |
|  754 | (SWC-103) Floating Pragma                | Low      | A floating pragma is set.                                 |
|  776 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "**" discovered                      |
|  776 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "*" discovered                       |
|  777 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "%" discovered                       |
|  777 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "-" discovered                       |
|  787 | (SWC-108) State Variable Default Visibility | Low   | State variable visibility is not set.                     |
```



```
|  926 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "++" discovered                      |
|  927 | (SWC-110) Assert Violation               | Unknown  | Out of bounds array access                                |
|  928 | (SWC-110) Assert Violation               | Unknown  | Out of bounds array access                                |
|  928 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "-" discovered                       |
|  928 | (SWC-101) Integer Overflow and Underflow | Unknown  | Compiler-rewritable "<uint> - 1" discovered               |
| 1007 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "++" discovered                      |
| 1008 | (SWC-110) Assert Violation               | Unknown  | Out of bounds array access                                |
| 1009 | (SWC-110) Assert Violation               | Unknown  | Out of bounds array access                                |
| 1010 | (SWC-110) Assert Violation               | Unknown  | Out of bounds array access                                |
| 1026 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "**" discovered                      |
| 1032 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "**" discovered                      |
| 1158 | (SWC-108) State Variable Default Visibility | Low   | State variable visibility is not set.                     |
| 1225 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "-" discovered                       |
| 1225 | (SWC-101) Integer Overflow and Underflow | Unknown  | Arithmetic operation "**" discovered                      |
```

Mythril results



```
root@sv-VirtualBox:/home/sv/Salsa# myth analyze Salsa.sol
The analysis was completed successfully. No issues were detected.
```

Linter results

NA

Conclusion

In this audit, we thoroughly analyzed SALSA VALLEY's Smart Contract. The current code base is well organized but there are promptly some Medium and Low type of issues found in the first phase of Smart Contract Audit.

Meanwhile, we need to emphasize that smart contracts as a whole are still in an early, but exciting stage of development. To improve this report, we greatly appreciate any constructive feedback or suggestions, on our methodology, audit findings, or potential gaps in scope/coverage.

Disclaimer

Cysro has analysed this smart contract in accordance with the best practices at the date of this report. This report is based on extensive methodological examination and analysis of code, in relation to the cyber security vulnerabilities, blockchain security, and cryptocurrency. The report only represents advice and remediations for clients to improve the quality of code while intending to diminish the inherent risks of blockchains. Cysro recommends conducting a bug bounty program to confirm a high level of security of this smart contract. Cysro does not provide any assurance of a complete bug-free contract.

While Cysro has given its best in conducting the analysis and producing this report, it is important to note that you should not rely on this report to make any decision for investment or involvement in any particular project. This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. Please conduct your own due diligence before investing in any asset. Cysro shall not be liable for any losses incurred in these cases.

The analysis of the security by Cysro is solely based on the smart contract. No other applications or functionalities were reviewed.

About

Cysro is a privately held London and India based cyber security and blockchain technology company. It is built by a team of ethical hackers to aid businesses in battling off cyberattacks.

We specialize in providing services of penetration testing, smart contract auditing, and know your customer. Our mission is to offer the best services possible with the right people, right methodology, right scope, and right report.

Our detailed audit reports shall assist you in comprehending your risk exposure, addressing security issues, and improving data security for your business.

CYSRO

Cyber Security

SMART CONTRACT SECURITY AUDIT