Smart Contract

Security Assessment

For LooksYummyToken

26 Sept 2022

Ascendant

@ascendantproj

www.ascendant.finance

Table of Contents

DISCLAIMER

Ascendant Finance ("Ascendant") has conducted an independent audit to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the codes that were provided for the scope of this audit. This audit report does not constitute agreement, acceptance or advocation for the Project that was audited, and users relying on this audit report should not consider this as having any merit for financial advice in any shape, form or nature. The contracts audited do not account for any economic developments that may be pursued by the Project in question, and that the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are completely free of exploits, bugs, vulnerabilities or deprecation of technologies. Further, this audit report shall not be disclosed nor transmitted to any persons or parties on any objective, goal or justification without due written assent, acquiescence or approval by Ascendant.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any persons reading this report should invest their funds without sufficient individual due diligence regardless of the findings presented in this report. Information is provided 'as is', and Ascendant is under no covenant to the completeness, accuracy or solidity of the contracts audited. In no event will Ascendant or its partners, employees, agents or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions and/or actions with regards to the information provided in this audit report.

Cryptocurrencies and any technologies by extension directly or indirectly related to cryptocurrencies are highly volatile and speculative by nature. All reasonable due diligence and safeguards may yet be insufficient, and users should exercise considerable caution when participating in any shape or form in this nascent industry.

The audit report has made all reasonable attempts to provide clear and articulate recommendations to the Project team with respect to the rectification, amendment and/or revision of any highlighted issues, vulnerabilities or exploits within the contracts provided. It is the sole responsibility of the Project team to sufficiently test and perform checks, ensuring that the contracts are functioning as intended, specifically that the functions therein contained within said contracts have the desired intended effects, functionalities and outcomes of the Project team.

Ascendant retains full rights over all intellectual property (including expertise and new attack or exploit vectors) discovered during the audit process. Ascendant is therefore allowed and expected to re-use this knowledge in subsequent audits and to inform existing projects that may have similar vulnerabilities. Ascendant may, at its discretion, claim bug bounties from third-parties while doing so.

Executive Summary

Severity	Found
High	0
Medium	3
Low	23
Informational	143
Total	169

We performed an independent technical audit to identify Smart Contracts uncertainties. This shall protect the code from illegitimate authorization attempts or external & internal threats of any type. This also ensures end-to-end proofing of the contract from frauds. The audit was performed semi-manually. We analyzed the Smart Contracts code line-by-line and used an automation tool to report any suspicious code.

The following tools were used:

- Truffle
- Remix IDE
- Slither

Overview

This report has been prepared for LooksYummyToken on the Binance network. Ascendant provides a user-centered examination of the smart contracts to look for vulnerabilities, logic errors or other issues from both an internal and external perspective.

Summary

Project Name	LooksYummyToken
Platform	Binance
Language	Solidity

Contracts Assessed

Name	Location
LooksYummyToken.sol	Not Published
SafeMath.sol	In LooksYummyToken contract
IERC20.sol	In LooksYummyToken contract
IERC165Upgradeable.sol	In LooksYummyToken contract
StringsUpgradeable.sol	In LooksYummyToken contract
IAccessControlUpgradeable.sol	In LooksYummyToken contract

Name	Location
StorageSlotUpgradeable.sol	In LooksYummyToken contract
IBeaconUpgradeable.sol	In LooksYummyToken contract
IERC1822ProxiableUpgradeable.sol	In LooksYummyToken contract
AddressUpgradeable.sol	In LooksYummyToken contract
Initializable.sol	In LooksYummyToken contract
ERC165Upgradeable.sol	In LooksYummyToken contract
ERC1967UpgradeUpgradeable.sol	In LooksYummyToken contract
UUPSUpgradeable.sol	In LooksYummyToken contract
ContextUpgradeable.sol	In LooksYummyToken contract
AccessControlUpgradeable.sol	In LooksYummyToken contract
OwnableUpgradeable.sol	In LooksYummyToken contract
IERC20Upgradeable.sol	In LooksYummyToken contract
IERC20MetadataUpgradeable.sol	In LooksYummyToken contract
ERC20Upgradeable.sol	In LooksYummyToken contract
Math.sol	In LooksYummyToken contract
IYummyswapFactory.sol	In LooksYummyToken contract
IYummyswapPair	In LooksYummyToken contract
IYummyswapRouter02	In LooksYummyToken contract

Findings Summary

Severity	Found
High	0
Medium	3
Low	23
Informational	143
Total	169

Classification of Issues

High	Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, or impairment of the contract and its functions. Issues under this classification are recommended to be fixed with utmost urgency.
Medium	Bugs or issues that may be subject to exploit, though their impact is somewhat limited. Issues under this classification are recommended to be fixed as soon as possible.
Low	Effects are minimal in isolation and do not pose a significant danger to the project or its users. Issues under this classification are recommended to be fixed nonetheless.
Informational	Consistency, syntax or style best practices, Generally pose a negligible level of risk, if any.

Manual Review

Issues Checking Status

Issue Description	Checking Status
Compiler errors	PASS
Race conditions and Reentrancy. Cross-function race conditions.	PASS
Possible delays in data delivery.	PASS
Oracle calls.	PASS
Front running.	PASS
Timestamp dependence.	PASS
Integer Overflow and Underflow.	PASS
DoS with Revert.	PASS
DoS with block gas limit.	PASS
Methods execution permissions.	PASS
Economy model of the contract.	PASS
The impact of the exchange rate on the logic.	PASS
Private user data leaks.	PASS
Malicious Event log.	PASS
Scoping and Declarations.	PASS
Uninitialized storage pointers.	PASS

Arithmetic accuracy.	PASS
Design Logic.	PASS
Cross-function race conditions.	PASS
Safe Open Zeppelin contracts implementation and usage.	PASS
Fallback function security.	PASS

Audit Findings

Severity	Medium
Contract	LooksYummyToken.sol
Description	Logic: swapTokensforBUSD attempts to swap BEP20 tokens but calls swapExactTokensForETHSupportingFees

Code Snippet	function swapTokensForBUSD(uint256 tokenAmount) private {
	 // generate the uniswap pair path of token -> weth
	 address[] memory path = new address[](2);
	 path[0] = address(this);
	 path[1] = busdAddress;
	
	 _approve(address(this), address(yummyswapV2Router), tokenAmount);
	
	 // make the swap
	 yummyswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(

Recommendation	The correct Yummyswap function to call is swapExactTokensForTokensSupportingFeeOnTransferToken, since that's what the path specifies.
Status	

Audit Findings

Severity	Medium
Contract	LooksYummyToken.sol
Description	No fallback function
Code Snippet	N/A
Recommendation	**Note: This issue is partly dependent on Medium Issue #1.** The contract swaps tokens for BNB (or the chain's native token). The contract must contain a fallback or receiver function in order to handle the BNB, or else this function will fail when it is triggered. Regardless of whether your intention is to swap for BNB or a BEP20 token, it is good practice to include a fallback
Status	

Audit Findings

Severity	Medium
Contract	LooksYummyToken.sol
Description	Unprotected initializer

Code Snippet	function initialize(
	address Owner1,
	address Owner2,
	address Company,
	address Appserver,
	address swapRouter,
	address _busdAddress
	) public initializer {
	__ERC20_init("LooksYummyToken", "LYT");
	__Ownable_init();
	__UUPSUpgradeable_init();
	
	_grantRole(DEFAULT_ADMIN_ROLE, msg.sender);
	
	_mint(Owner1, 15 * 1e6 * 1e6);
	_mint(Owner2, 15 * 1e6 * 1e6);
	_mint(Company, 70 * 1e6 * 1e6);
	_mint(Appserver, 25 * 1e6 * 1e6);
	
	busdAddress = _busdAddress;
	...
	
	}

Recommendation	Initializer function is public and can be called by any contract. Issue is mitigated by the disableInitializers function in the constructor.
Status	

Audit Findings

Severity	Lowx3
Contract	LooksYummyToken.sol
Description	Functions lack zero-check.
Code Snippet	LooksYummyToken.setVaultAddress LooksYummyToken.setFaucetAddress LooksYummyToken.setBondingAddress
Recommendation	Require statements should be added to the above functions to prevent being set to the zero address.
Status	

Audit Findings

Severity	Low
Contract	LooksYummyToken.sol
Description	Multiple solidity versions used
Code Snippet	N/A
Recommendation	Use only one solidity version to ensure consistency.
Status	

Audit Findings

Severity	**Informational**
Contract	LooksYummyToken.sol
Description	Optimization: Math.sol and SafeMath.sol both imported
Code Snippet	N/A
Recommendation	Both Math.sol and SafeMath.sol are used but only SafeMath is likely necessary. Remove any redundancies from the source code.
Status	

Functional Test Status

Function Name	Type/Return Type	Score
IERC20Upgradeable		
allowance	read/external	PASS
approve	write/external	PASS
balanceOf	read/external	PASS
totalSupply	read/external	PASS
transfer	write/external	PASS
transferFrom	write/external	PASS
IAccessControlUpgradeable		
getRoleAdmin	read/external	PASS
grantRole	write/external	PASS
hasRole	read/external	PASS
renounceRole	write/external	PASS
revokeRole	write/external	PASS
StorageSlotUpgradeable		
getAddressSlot	internal	PASS
getBooleanSlot	internal	PASS
getBytes32Slot	internal	PASS
getUint256Slot	internal	PASS

Function Name	Type/Return Type	Score
AddressUpgradeable		
functionCall	internal	PASS
functionCallWithValue	internal	PASS
functionStaticCall	internal	PASS
isContract	internal	PASS
sendValue	internal	PASS
verifyCallResult	internal	PASS
Intializable		
_disableInitializers	internal	PASS
UUPSUpgradeable		
_authorizeUpgrade	internal	PASS
proxiableUUID	read/external	PASS
upgradeTo	write/external	PASS
Context		
_msgData	internal	PASS
_msgSender	internal	PASS
Ownable		
_checkOwner	internal	PASS

Function Name	Type/Return Type	Score
_transferOwnership	internal	PASS
owner	read/public	PASS
renounceOwnership	write/public	PASS
transferOwnership	write/public	PASS
IERC20MetadataUpgradeable		
decimals	read/external	PASS
name	read/external	PASS
symbol	read/external	PASS
IYummyswapFactory		
INIT_CODE_HASH	read/external	PASS
allPairs	read/external	PASS
allPairsLength	read/external	PASS
createPair	write/external	PASS
feeTo	write/external	PASS
feeToSetter	write/external	PASS
getPair	read/external	PASS
setDevFee	write/external	PASS
setFeeTo	write/external	PASS

setFeeToSetter	write/external	PASS
setSwapFee	write/external	PASS
IYummyswapPair		
DOMAIN_SEPARATOR	read/external	PASS
MINIMUM_LIQUIDITY	read/external	PASS
PERMIT_TYPEHASH	write/external	PASS
allowance	read/external	PASS
approve	write/external	PASS
balanceOf	read/external	PASS
burn	write/external	PASS
decimals	read/external	PASS
devFee	read/external	PASS
factory	read/external	PASS
getReserves	read/external	PASS
initialize	write/external	PASS
kLast	read/external	PASS
mint	write/external	PASS
name	read/external	PASS
nonces	read/external	PASS
permit	write/external	PASS
price0CumulativeLast	read/external	PASS
price1CumulativeLast	read/external	PASS

setDevFee	write/external	PASS
setSwapFee	write/external	PASS
skim	external	PASS
swapFee	external	PASS
symbol	read/external	PASS
sync	external	PASS
token0	read/external	PASS
token1	read/external	PASS
transfer	write/external	PASS
transferFrom	write/external	PASS
IYummyswapRouter02		
WETH	read/external	PASS
addLiquidity	write/external	PASS
addLiquidityETH	write/external	PASS
getAmountIn	read/external	PASS
getAmountOut	read/external	PASS
getAmountsIn	read/external	PASS
getAmountsOut	read/external	PASS
quote	external	PASS
removeLiquidity	write/external	PASS
removeLiquidityETH	write/external	PASS
removeLiquidityETHSupportingFeeOnTransferTokens	write/external	PASS

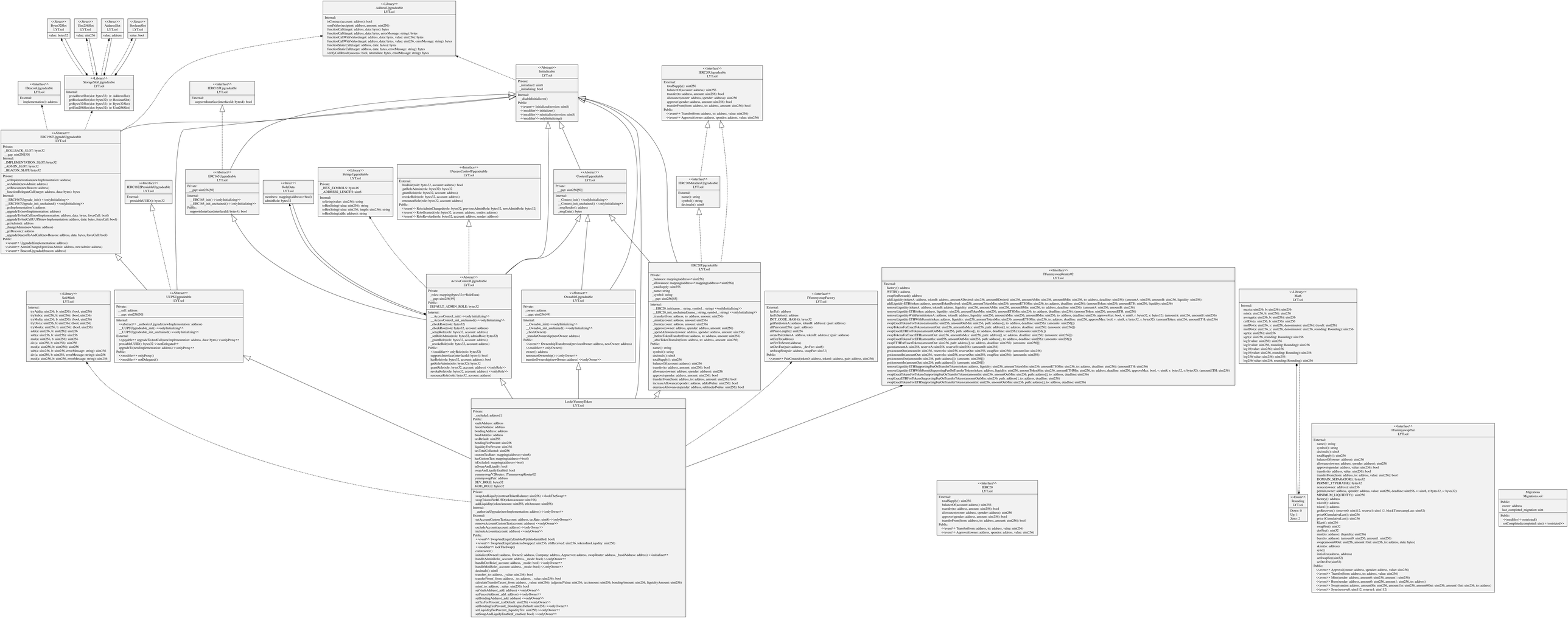
removeLiquidityETHWithPermit	write/external	PASS
removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	write/external	PASS
removeLiquidityWithPermit	write/external	PASS
swapETHForExactTokens	write/external	PASS
swapExactETHForTokens	write/external	PASS
swapExactETHForTokensSupportingFeeOnTransferTokens	write/external	PASS
swapExactTokensForTokens	write/external	PASS
swapExactTokensForTokensSupportingFeeOnTransferTokens	write/external	PASS
swapFeeReward	write/external	PASS
swapTokensForExactETH	write/external	PASS
swapTokensForExactTokens	write/external	PASS
LooksYummyToken		
addLiquidity	private	PASS
calculateTransferTaxes	read/public	PASS
decimals	read/public	PASS
excludeAccount	write/external	PASS
handleAdminRole	write/public	PASS
handleDevRole	write/public	PASS
handleModRole	write/public	PASS
includeAccount	write/external	PASS
initialize	write/public	PASS
mint	write/public	PASS

removeAccountCustomTax	write/external	PASS
setAccountCustomTax	write/external	PASS
setBondingAddress	write/public	PASS
setBondingFeePercent	write/public	PASS
setFaucetAddress	write/public	PASS
setLiquidityFeePercent	write/public	PASS
setSwapAndLiquifyEnabled	write/public	PASS
setTaxFeePercent	write/public	PASS
setVaultAddress	write/public	PASS
swapAndLiquify	private	FAIL
swapTokensForBUSD	private	FAIL
transfer	write/public	PASS
transferFrom	write/public	PASS

Omitted Results

Note: Any issues that have been omitted from this report have been deemed by the reviewing team as irrelevant, inapplicable, and/or negligible to the proper functioning of this contract. Thus, any omitted issues can be safely ignored.

Automated Review


«Struct»
Bytes32Slot
LYT.sol
value: bytes32
«Struct»
Uint256Slot
LYT.sol
value: uint256
«Struct»
AddressSlot
LYT.sol
value: address
«Struct»
BooleanSlot
LYT.sol
value: bool
«Library»
AddressUpgradeable
LYT.sol
Internal:
isContract(account: address): bool
functionCall(target: address, amount: uint256)
functionCall(target: address, data: bytes): bytes
functionCall(target: address, data: bytes, errorMessage: string): bytes
functionCallWithValue(target: address, data: bytes, value: uint256): bytes
functionCallWithValue(target: address, data: bytes, value: uint256, errorMessage: string): bytes
functionStaticCall(target: address, data: bytes): bytes
functionStaticCall(target: address, data: bytes, errorMessage: string): bytes
verifyCallResult(success: bool, returndata: bytes, errorMessage: string): bytes
«Library»
StorageSlotUpgradeable
LYT.sol
Internal:
getAddressSlot(slot: bytes32): (r: AddressSlot)
getBooleanSlot(slot: bytes32): (r: BooleanSlot)
getBytes32Slot(slot: bytes32): (r: Bytes32Slot)
getUint256Slot(slot: bytes32): (r: Uint256Slot)
«Interface»
IBeaconUpgradeable
LYT.sol
External:
implementation(): address
«Interface»
IERC165Upgradeable
LYT.sol
External:
supportsInterface(interfaceId: bytes4): bool
«Abstract»
Initializable
LYT.sol
Private:
_initialized: uint8
_initializing: bool
Internal:
_disableInitializers()
Public:
«event» Initialized(version: uint8)
«modifier» initializer()
«modifier» reinitializer(version: uint8)
«modifier» onlyInitializing()
«Interface»
IERC20Upgradeable
LYT.sol
External:
totalSupply(): uint256
balanceOf(account: address): uint256
transfer(to: address, amount: uint256): bool
allowance(owner: address, spender: address): uint256
approve(spender: address, amount: uint256): bool
transferFrom(from: address, to: address, amount: uint256): bool
Public:
«event» Transfer(from: address, to: address, value: uint256)
«event» Approval(owner: address, spender: address, value: uint256)
«Abstract»
ERC1967UpgradeUpgradeable
LYT.sol
Private:
_ROLLBACK_SLOT: bytes32
__gap: uint256[50]
Internal:
_IMPLEMENTATION_SLOT: bytes32
_ADMIN_SLOT: bytes32
_BEACON_SLOT: bytes32
Private:
_setImplementation(newImplementation: address)
_setAdmin(newAdmin: address)
_setBeacon(newBeacon: address)
_functionDelegateCall(target: address, data: bytes): bytes
Internal:
__ERC1967Upgrade_init() «only Initializing»
__ERC1967Upgrade_init_unchained() «only Initializing»
_getImplementation(): address
_upgradeToAndCallUUPS(newImplementation: address, data: bytes, forceCall: bool)
_upgradeToAndCallUUPS(newImplementation: address, data: bytes, forceCall: bool)
_getAdmin(): address
_changeAdmin(newAdmin: address)
_getBeacon(): address
_upgradeBeaconToAndCall(newBeacon: address, data: bytes, forceCall: bool)
Public:
«event» Upgraded(implementation: address)
«event» AdminChanged(previousAdmin: address, newAdmin: address)
«event» BeaconUpgraded(beacon: address)
«Interface»
IERC1822ProxiableUpgradeable
LYT.sol
External:
proxiableUUID(): bytes32
«Abstract»
ERC165Upgradeable
LYT.sol
Private:
__gap: uint256[50]
Internal:
__ERC165_init() «only Initializing»
__ERC165_init_unchained() «only Initializing»
Public:
supportsInterface(interfaceId: bytes4): bool
«Struct»
RoleData
LYT.sol
members: mapping(address=>bool)
adminRole: bytes32
«Library»
StringsUpgradeable
LYT.sol
Private:
_HEX_SYMBOLS: bytes16
_ADDRESS_LENGTH: uint8
Internal:
toString(value: uint256): string
toHexString(value: uint256): string
toHexString(value: uint256, length: uint256): string
toHexString(addr: address): string
«Interface»
IAccessControlUpgradeable
LYT.sol
External:
hasRole(role: bytes32, account: address): bool
getRoleAdmin(role: bytes32): bytes32
grantRole(role: bytes32, account: address)
revokeRole(role: bytes32, account: address)
renounceRole(role: bytes32, account: address)
Public:
«event» RoleAdminChanged(role: bytes32, previousAdminRole: bytes32, newAdminRole: bytes32)
«event» RoleGranted(role: bytes32, account: address, sender: address)
«event» RoleRevoked(role: bytes32, account: address, sender: address)
«Abstract»
ContextUpgradeable
LYT.sol
Private:
__gap: uint256[50]
Internal:
__Context_init() «only Initializing»
__Context_init_unchained() «only Initializing»
_msgSender(): address
_msgData(): bytes
«Interface»
IERC20MetadataUpgradeable
LYT.sol
External:
name(): string
symbol(): string
decimals(): uint8
«Abstract»
AccessControlUpgradeable
LYT.sol
Private:
_roles: mapping(bytes32=>RoleData)
__gap: uint256[49]
Public:
DEFAULT_ADMIN_ROLE: bytes32
Internal:
__AccessControl_init() «only Initializing»
__AccessControl_init_unchained() «only Initializing»
_checkRole(role: bytes32)
_checkRole(role: bytes32, account: address)
_setupRole(role: bytes32, account: address)
_getRoleAdmin(role: bytes32): bytes32
_grantRole(role: bytes32, account: address)
_revokeRole(role: bytes32, account: address)
Public:
«modifier» onlyRole(role: bytes32)
supportsInterface(interfaceId: bytes4): bool
hasRole(role: bytes32, account: address): bool
getRoleAdmin(role: bytes32): bytes32
grantRole(role: bytes32, account: address) «only Role»
revokeRole(role: bytes32, account: address) «only Role»
renounceRole(role: bytes32, account: address)
«Abstract»
OwnableUpgradeable
LYT.sol
Private:
owner: address
__gap: uint256[49]
Internal:
__Ownable_init() «only Initializing»
__Ownable_init_unchained() «only Initializing»
_checkOwner()
_transferOwnership(newOwner: address)
Public:
«event» OwnershipTransferred(previousOwner: address, newOwner: address)
«modifier» onlyOwner()
owner(): address
renounceOwnership() «only Owner»
transferOwnership(newOwner: address) «only Owner»
«Abstract»
ERC20Upgradeable
LYT.sol
Private:
_balances: mapping(address=>uint256)
_allowances: mapping(address=>mapping(address=>uint256))
_totalSupply: uint256
_name: string
_symbol: string
__gap: uint256[45]
Internal:
__ERC20_init(name_: string, symbol_: string) «only Initializing»
__ERC20_init_unchained(name_: string, symbol_: string) «only Initializing»
_transfer(from: address, to: address, amount: uint256)
_mint(account: address, amount: uint256)
_burn(account: address, amount: uint256)
_approve(owner: address, spender: address, amount: uint256)
_spendAllowance(owner: address, spender: address, amount: uint256)
_beforeTokenTransfer(from: address, to: address, amount: uint256)
_afterTokenTransfer(from: address, to: address, amount: uint256)
Public:
name(): string
symbol(): string
decimals(): uint8
totalSupply(): uint256
balanceOf(account: address): uint256
transfer(to: address, amount: uint256): bool
allowance(owner: address, spender: address): uint256
approve(spender: address, amount: uint256): bool
transferFrom(from: address, to: address, amount: uint256): bool
increaseAllowance(spender: address, addedValue: uint256): bool
decreaseAllowance(spender: address, subtractedValue: uint256): bool
«Interface»
IERC20Metadata
LYT.sol
External:
name(): string
symbol(): string
decimals(): uint8
«Interface»
IERC20
LYT.sol
External:
totalSupply(): uint256
balanceOf(account: address): uint256
transfer(to: address, amount: uint256): bool
allowance(owner: address, spender: address): uint256
approve(spender: address, amount: uint256): bool
transferFrom(from: address, to: address, amount: uint256): bool
Public:
«event» Transfer(from: address, to: address, value: uint256)
«event» Approval(owner: address, spender: address, value: uint256)
«Library»
SafeMath
LYT.sol
Internal:
tryAdd(a: uint256, b: uint256): (bool, uint256)
trySub(a: uint256, b: uint256): (bool, uint256)
tryMul(a: uint256, b: uint256): (bool, uint256)
tryDiv(a: uint256, b: uint256): (bool, uint256)
tryMod(a: uint256, b: uint256): (bool, uint256)
add(a: uint256, b: uint256): uint256
sub(a: uint256, b: uint256): uint256
mul(a: uint256, b: uint256): uint256
div(a: uint256, b: uint256): uint256
mod(a: uint256, b: uint256): uint256
sub(a: uint256, b: uint256, errorMessage: string): uint256
div(a: uint256, b: uint256, errorMessage: string): uint256
mod(a: uint256, b: uint256, errorMessage: string): uint256
«Abstract»
UUPSUpgradeable
LYT.sol
Private:
__self: address
__gap: uint256[50]
Internal:
«abstract» _authorizeUpgrade(newImplementation: address)
__UUPSUpgradeable_init() «only Initializing»
__UUPSUpgradeable_init_unchained() «only Initializing»
Public:
«payable» upgradeTo(newImplementation: address, data: bytes) «onlyProxy»
proxiableUUID(): bytes32 «notDelegated»
upgradeToAndCall(newImplementation: address) «onlyProxy»
Public:
«modifier» onlyProxy()
«modifier» notDelegated()
«Interface»
IUniswapV2Factory
LYT.sol
External:
feeTo(): address
feeToSetter(): address
INIT_CODE_PAIR_HASH(): bytes32
getPair(tokenA: address, tokenB: address): (pair: address)
allPairs(uint256): (pair: address)
allPairsLength(): uint256
createPair(tokenA: address, tokenB: address): (pair: address)
setFeeTo(address)
setFeeToSetter(address)
setDevFee(pair: address, _devFee: uint8)
setSwapFee(pair: address, swapFee: uint32)
Public:
«event» PairCreated(token0: address, token1: address, pair: address, uint256)
«Interface»
IUniswapV2Router02
LYT.sol
External:
factory(): address
WETH(): address
swapFeeReward(): address
addLiquidity(tokenA: address, tokenB: address, amountADesired: uint256, amountBDesired: uint256, amountAMin: uint256, amountBMin: uint256, to: address, deadline: uint256): (amountA: uint256, amountB: uint256, liquidity: uint256)
addLiquidityETH(token: address, amountTokenDesired: uint256, amountTokenMin: uint256, amountETHMin: uint256, to: address, deadline: uint256): (amountToken: uint256, amountETH: uint256, liquidity: uint256)
removeLiquidity(tokenA: address, tokenB: address, liquidity: uint256, amountAMin: uint256, amountBMin: uint256, to: address, deadline: uint256): (amountA: uint256, amountB: uint256)
removeLiquidityETH(token: address, liquidity: uint256, amountTokenMin: uint256, amountETHMin: uint256, to: address, deadline: uint256): (amountToken: uint256, amountETH: uint256)
removeLiquidityWithPermit(tokenA: address, tokenB: address, liquidity: uint256, amountAMin: uint256, amountBMin: uint256, to: address, deadline: uint256, approveMax: bool, v: uint8, r: bytes32, s: bytes32): (amountA: uint256, amountB: uint256)
removeLiquidityETHWithPermit(token: address, liquidity: uint256, amountTokenMin: uint256, amountETHMin: uint256, to: address, deadline: uint256, approveMax: bool, v: uint8, r: bytes32, s: bytes32): (amountToken: uint256, amountETH: uint256)
swapExactTokensForTokens(amountIn: uint256, amountOutMin: uint256, path: address[], to: address, deadline: uint256): (amounts: uint256[])
swapTokensForExactTokens(amountOut: uint256, amountInMax: uint256, path: address[], to: address, deadline: uint256): (amounts: uint256[])
swapExactETHForTokens(amountOutMin: uint256, path: address[], to: address, deadline: uint256): (amounts: uint256[])
swapTokensForExactETH(amountOut: uint256, amountInMax: uint256, path: address[], to: address, deadline: uint256): (amounts: uint256[])
swapExactTokensForETH(amountIn: uint256, amountOutMin: uint256, path: address[], to: address, deadline: uint256): (amounts: uint256[])
swapETHForExactTokens(amountOut: uint256, path: address[], to: address, deadline: uint256): (amounts: uint256[])
quote(amountA: uint256, reserveA: uint256, reserveB: uint256): (amountB: uint256)
getAmountOut(amountIn: uint256, reserveIn: uint256, reserveOut: uint256, swapFee: uint256): (amountOut: uint256)
getAmountIn(amountOut: uint256, reserveIn: uint256, reserveOut: uint256, swapFee: uint256): (amountIn: uint256)
getAmountsOut(amountIn: uint256, path: address[]): (amounts: uint256[])
getAmountsIn(amountOut: uint256, path: address[]): (amounts: uint256[])
removeLiquidityETHSupportingFeeOnTransferTokens(token: address, liquidity: uint256, amountTokenMin: uint256, amountETHMin: uint256, to: address, deadline: uint256): (amountETH: uint256)
removeLiquidityETHWithPermitSupportingFeeOnTransferTokens(token: address, liquidity: uint256, amountTokenMin: uint256, amountETHMin: uint256, to: address, deadline: uint256, approveMax: bool, v: uint8, r: bytes32, s: bytes32): (amountETH: uint256)
swapExactTokensForTokensSupportingFeeOnTransferTokens(amountIn: uint256, amountOutMin: uint256, path: address[], to: address, deadline: uint256)
swapExactETHForTokensSupportingFeeOnTransferTokens(amountOutMin: uint256, path: address[], to: address, deadline: uint256)
swapExactTokensForETHSupportingFeeOnTransferTokens(amountIn: uint256, amountOutMin: uint256, path: address[], to: address, deadline: uint256)
«Library»
Math
LYT.sol
Internal:
max(a: uint256, b: uint256): uint256
min(a: uint256, b: uint256): uint256
average(a: uint256, b: uint256): uint256
ceilDiv(a: uint256, b: uint256): uint256
mulDiv(x: uint256, y: uint256, denominator: uint256): (result: uint256)
mulDiv(x: uint256, y: uint256, denominator: uint256, rounding: Rounding): uint256
sqrt(a: uint256): uint256
sqrt(a: uint256, rounding: Rounding): uint256
log2(value: uint256): uint256
log2(value: uint256, rounding: Rounding): uint256
log10(value: uint256): uint256
log10(value: uint256, rounding: Rounding): uint256
log256(value: uint256): uint256
log256(value: uint256, rounding: Rounding): uint256
«Enum»
Rounding
LYT.sol
Down: 0
Up: 1
Zero: 2
«Interface»
IUniswapV2Pair
LYT.sol
External:
name(): string
symbol(): string
decimals(): uint8
totalSupply(): uint256
balanceOf(owner: address): uint256
allowance(owner: address, spender: address): uint256
approve(spender: address, value: uint256): bool
transfer(to: address, value: uint256): bool
transferFrom(from: address, to: address, value: uint256): bool
DOMAIN_SEPARATOR(): bytes32
PERMIT_TYPEHASH(): bytes32
nonces(owner: address): uint256
permit(owner: address, spender: address, value: uint256, deadline: uint256, v: uint8, r: bytes32, s: bytes32)
MINIMUM_LIQUIDITY(): uint256
factory(): address
token0(): address
token1(): address
getReserves(): (reserve0: uint112, reserve1: uint112, blockTimestampLast: uint32)
price0CumulativeLast(): uint256
price1CumulativeLast(): uint256
kLast(): uint256
swapFee(): uint32
devFee(): uint32
mint(to: address): (liquidity: uint256)
burn(to: address): (amount0: uint256, amount1: uint256)
swap(amount0Out: uint256, amount1Out: uint256, to: address, data: bytes)
skim(to: address)
sync()
initialize(address, address)
setSwapFee(uint32)
setDevFee(uint32)
Public:
«event» Approval(owner: address, spender: address, value: uint256)
«event» Transfer(from: address, to: address, value: uint256)
«event» Mint(sender: address, amount0: uint256, amount1: uint256)
«event» Burn(sender: address, amount0: uint256, amount1: uint256, to: address)
«event» Swap(sender: address, amount0In: uint256, amount1In: uint256, amount0Out: uint256, amount1Out: uint256, to: address)
«event» Sync(reserve0: uint112, reserve1: uint112)
Migrations
Migrations.sol
Public:
owner: address
last_completed_migration: uint
Public:
«modifier» restricted()
_setCompleted(completed: uint) «restricted»
LooksYummyToken
LYT.sol
Private:
_excluded: address[]
Public:
vaultAddress: address
faucetAddress: address
bondingAddress: address
burnAddress: address
taxDefault: uint256
bondingFeePercent: uint256
liquidityFeePercent: uint256
totTotalCollected: uint256
customTaxRate: mapping(address=>uint8)
hasCustomTax: mapping(address=>bool)
isExcluded: mapping(address=>bool)
isSwapAndLiquify: bool
swapAndLiquifyEnabled: bool
yummyswapV2Router: IYummyswapRouter02
yummyswapPair: address
DEV_ROLE: bytes32
MOD_ROLE: bytes32
Private:
swapAndLiquify(contractTokenBalance: uint256) «lockTheSwap»
swapTokensForBUSD(tokenAmount: uint256)
addLiquidity(tokenAmount: uint256, ethAmount: uint256)
Internal:
_authorize(Upgradeable) «onlyOwner»
External:
setAccountCustomTax(account: address, taxRate: uint8) «onlyOwner»
includeAccount(account: address) «onlyOwner»
excludeAccount(account: address) «onlyOwner»
includeInAccount(account: address) «onlyOwner»
Public:
«event» SwapAndLiquifyEnabledUpdated(enabled: bool)
«event» SwapAndLiquify(tokensSwapped: uint256, ethReceived: uint256, tokensIntoLiquidity: uint256)
«modifier» lockTheSwap()
constructor()
initialize(Owner0: address, Owner2: address, Company: address, Approver: address, swapRouter: address, _busdAddress: address) «initializer»
handleAdminRole(_account: address, _mode: bool) «onlyOwner»
handleDevRole(_account: address, _mode: bool) «onlyOwner»
handleModRole(_account: address, _mode: bool) «onlyOwner»
decimals(): uint8
transfer(to: address, _value: uint256): bool
transferFrom(_from: address, _to: address, _value: uint256): bool
calculateTransferTaxes(_from: address, _value: uint256): (adjustedValue: uint256, taxAmount: uint256, bondingAmount: uint256, liquidityAmount: uint256)
mint(_to: address, _value: uint256): bool
setVaultAddress(_add: address) «onlyOwner»
setFaucetAddress(_add: address) «onlyOwner»
setBondingAddress(_add: address) «onlyOwner»
setTaxFeePercent(_taxDefault: uint256) «onlyOwner»
setBondingFeePercent(_BondingtaxDefault: uint256) «onlyOwner»
setLiquidityFeePercent(_liquidityFee: uint256) «onlyOwner»
setSwapAndLiquifyEnabled(_enabled: bool) «onlyOwner»

Conclusion

The smart contracts reviewed in this audit contain no critical severity issues and all Medium to Low issues have either been corrected or acknowledged.

Please check the disclaimer above and note, the audit makes no statements or warranties on business model, investment attractiveness or code sustainability. The report is provided for the only contract mentioned in the report and does not include any other potential contracts deployed by Owner.

Ascendant

Ascendant

@ascendantproj

www.ascendant.finance

Ascendant