AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

Crazy Hippo

DATED : 7 Sep 23'

AUDIT SUMMARY

Project name – Crazy Hippo

Date: 7 Sep, 2023

Scope of Audit- Audit Ace was consulted to conduct the smart contract audit of the solidity source codes.

Audit Status: Passed

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	0	0	0	0	0
Acknowledged	0	0	0	0	0
Resolved	0	1	0	0	0

USED TOOLS

Tools:

1- Manual Review:
A line by line code review has been performed by audit ace team.

2- BSC Test Network: All tests were conducted on the BSC Test network, and each test has a corresponding transaction attached to it. These tests can be found in the "Functional Tests" section of the report.

3- Slither :
The code has undergone static analysis using Slither.

Testnet version:
The tests were performed using the contract deployed on the BSC Testnet, which can be found at the following address:
https://testnet.bscscan.com/token/0x62035c573954dbD4581f8f0fF89F77ed14CB9ee6

Token Information

Token Name : Crazy Hippo

Token Symbol: HIPPO

Decimals: 18

Token Supply: 200,000,000,000

Token Address:
0x890767185bC36cFDF52e24C9f05cdFC3Bd7948a9

Checksum:
3aa85371cb9853106409d78434d3d28f551c2fad

Owner:
0xC58FFC041806c4FFE34943C27703A3a7FdA5baEf
(at time of writing the audit)

Deployer:
0xC58FFC041806c4FFE34943C27703A3a7FdA5baEf

TOKEN OVERVIEW

Fees:

Buy Fees: 0-3%

Sell Fees: 0-3%

Transfer Fees: 0-3%

Fees Privilege: owner

Ownership: owned

Minting: No mint function

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Privileges: Initial distribution of the tokens

modifying fees

Enabling trades

AUDIT METHODOLOGY

The auditing process will follow a routine as special considerations by Auditace:

- Review of the specifications, sources, and instructions provided to Auditace to make sure the contract logic meets the intentions of the client without exposing the user's funds to risk.
- Manual review of the entire codebase by our experts, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
- Specification comparison is the process of checking whether the code does what the specifications, sources, and instructions provided to Auditace describe.
- Test coverage analysis determines whether the test cases are covering the code and how much code isexercised when we run the test cases.
- Symbolic execution is analysing a program to determine what inputs cause each part of a program to execute.
- Reviewing the codebase to improve maintainability, security, and control based on the established industry and academic practices.

VULNERABILITY CHECKLIST

- [x] Return values of low-level calls
- [x] Private modifier
- [x] Multiple Sends
- [x] Using Suicide
- [x] Gas Limitand Loops
- [x] Address hardcoded
- [x] Exception Disorder
- [x] Using inline assembly
- [x] Divide before multiply
- [x] Missing Zero Address Validation
- [x] Compiler version not fixed
- [x] **Gasless Send**
- [x] Using block.timestamp
- [x] Re-entrancy
- [x] Tautology or contradiction
- [x] Timestamp Dependence
- [x] Revert/require functions
- [x] Use of tx.origin
- [x] Integer overflow/underflow
- [x] Dangerous strict equalities
- [x] Using SHA3
- [x] Using throw

CLASSIFICATION OF RISK

Severity	Description
Critical	These vulnerabilities could be exploited easily and can lead to asset loss, data loss, asset, or data manipulation. They should be fixed right away.
High-Risk	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.
Medium-Risk	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.
Low-Risk	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.
Gas Optimization /Suggestion	A vulnerability that has an informational character but is not affecting any of the code.

Findings

Severity	Found
Critical	0
High-Risk	1
Medium-Risk	0
Low-Risk	0
Gas Optimization / Suggestions	0

INHERITANCE TREE

POINTS TO NOTE

- **Owner is able to adjust buy/sell/transfer fees within 0-3%**
- Owner is not able to blacklist an arbitrary wallet
- Owner is not able to disable trades
- Owner is not able to mint new tokens
- Owner is not able to set maximum wallet and maximum buy/sell/transfer limits
- **Owner must enable trades manually**

CONTRACT ASSESMENT

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				
	└	tryAdd	Internal 🔒			
	└	trySub	Internal 🔒			
	└	tryMul	Internal 🔒			
	└	tryDiv	Internal 🔒			
	└	tryMod	Internal 🔒			
	└	add	Internal 🔒			
	└	sub	Internal 🔒			
	└	mul	Internal 🔒			
	└	div	Internal 🔒			
	└	mod	Internal 🔒			
	└	sub	Internal 🔒			
	└	div	Internal 🔒			
	└	mod	Internal 🔒			
Context	Implementation				
	└	<Constructor>	Public ❗	🔴	NO ❗	
	└	_msgSender	Internal 🔒			
Ownable	Implementation	Context			
	└	<Constructor>	Public ❗	🔴	NO ❗	
	└	owner	Public ❗		NO ❗	
	└	renounceOwnership	Public ❗	🔴	onlyOwner	
	└	transferOwnership	Public ❗	🔴	onlyOwner	
BEP20Detailed	Implementation				
	└	<Constructor>	Public ❗	🔴	NO ❗	
	└	name	Public ❗		NO ❗	
	└	symbol	Public ❗		NO ❗	
	└	decimals	Public ❗		NO ❗	

CONTRACT ASSESMENT

Address	Library			
└	isContract	Internal 🔒		
SafeBEP20	Library			
└	safeTransfer	Internal 🔒	🛑	
└	safeTransferFrom	Internal 🔒	🛑	
└	safeApprove	Internal 🔒	🛑	
└	callOptionalReturn	Private 🔐	🛑	
IUniswapV2Factory	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❗
IUniswapV2Pair	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❗

CONTRACT ASSESMENT

└	mint	External ❗	🔴	NO❗
└	burn	External ❗	🔴	NO❗
└	swap	External ❗	🔴	NO❗
└	skim	External ❗	🔴	NO❗
└	sync	External ❗	🔴	NO❗
└	initialize	External ❗	🔴	NO❗
IUniswapV2Router01	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❗
IUniswapV2Router02	Interface	IUniswapV2Router01		
└	removeLiquidityETHSupportingFeeOnTransferTokens	External ❗	🔴	NO❗
└	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External ❗	🔴	NO❗
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🔴	NO❗
└	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗	💵	NO❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🔴	NO❗
HIPPO	Implementation	Context, Ownable, IBEP20, BEP20Detailed		
└	<Constructor>	Public ❗	🔴	BEP20Detailed
└	totalSupply	Public ❗		NO❗
└	balanceOf	Public ❗		NO❗
└	transfer	Public ❗	🔴	NO❗
└	allowance	Public ❗		NO❗
└	approve	Public ❗	🔴	NO❗

CONTRACT ASSESMENT

└	transferFrom	Public ❗	🛑	NO❗
└	increaseAllowance	Public ❗	🛑	NO❗
└	decreaseAllowance	Public ❗	🛑	NO❗
└	_approve	Internal 🔒	🛑	
└	enableTrading	External ❗	🛑	onlyOwner
└	isContract	Internal 🔒		
└	setBuyDevelopmentFeePercent	External ❗	🛑	onlyOwner
└	setSellDevelopmentFeePercent	External ❗	🛑	onlyOwner
└	setDevelopmentAddress	External ❗	🛑	onlyOwner
└	setSwapAndLiquifyEnabled	Public ❗	🛑	onlyOwner
└	changeNumTokensSellToFee	External ❗	🛑	onlyOwner
└	clearETH	External ❗	🛑	onlyOwner
└	clearERC20	External ❗	🛑	onlyOwner
└	excludeFromFee	Public ❗	🛑	onlyOwner
└	includeInFee	Public ❗	🛑	onlyOwner
└	isExcludedFromFee	Public ❗		NO❗
└	<Receive Ether>	External ❗	💵	NO❗
└	_transfer	Internal 🔒	🛑	
└	swapAndLiquify	Private 🔐	🛑	lockTheSwap
└	swapTokensForEth	Private 🔐	🛑	

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

STATIC ANALYSIS

```
Parameter HIPPO.clearERC20(address,address,uint256)._amount (contracts/Token.sol#822) is not in mixedCase
Variable HIPPO._balances (contracts/Token.sol#616) is not in mixedCase
Variable HIPPO._allowances (contracts/Token.sol#617) is not in mixedCase
Variable HIPPO._totalSupply (contracts/Token.sol#620) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Reentrancy in HIPPO._transfer(address,address,uint256) (contracts/Token.sol#843-908):
        External calls:
        - swapAndLiquify(contractTokenBalance) (contracts/Token.sol#866)
                - address(developmentAddress).transfer(address(this).balance) (contracts/Token.sol#914)
        State variables written after the call(s):
        - _balances[sender] = _balances[sender].sub(amount,BEP20: transfer amount exceeds balance) (contracts/Token.sol#892-895)
        - _balances[recipient] = _balances[recipient].add(TotalSent) (contracts/Token.sol#896)
        - _balances[address(this)] = _balances[address(this)].add(taxAmount) (contracts/Token.sol#897)
        - _balances[sender] = _balances[sender].sub(amount,BEP20: transfer amount exceeds balance) (contracts/Token.sol#901-904)
        - _balances[recipient] = _balances[recipient].add(amount) (contracts/Token.sol#905)
        - developmentFee = buyDevelopmentFee (contracts/Token.sol#886)
        - developmentFee = sellDevelopmentFee (contracts/Token.sol#888)
        Event emitted after the call(s):
        - Transfer(sender,recipient,TotalSent) (contracts/Token.sol#898)
        - Transfer(sender,address(this),taxAmount) (contracts/Token.sol#899)
        - Transfer(sender,recipient,amount) (contracts/Token.sol#906)
Reentrancy in HIPPO.swapAndLiquify(uint256) (contracts/Token.sol#910-917):
        External calls:
        - address(developmentAddress).transfer(address(this).balance) (contracts/Token.sol#914)
        Event emitted after the call(s):
        - SwapAndLiquify(contractTokenBalance,address(this).balance) (contracts/Token.sol#916)
Reentrancy in HIPPO.transferFrom(address,address,uint256) (contracts/Token.sol#703-718):
        External calls:
        - _transfer(sender,recipient,amount) (contracts/Token.sol#708)
                - address(developmentAddress).transfer(address(this).balance) (contracts/Token.sol#914)
        State variables written after the call(s):
        - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,BEP20: transfer amount exceeds allowance)) (contracts/Token.sol#709-716)
                - _allowances[towner][spender] = amount (contracts/Token.sol#750)
        Event emitted after the call(s):
        - Approval(towner,spender,amount) (contracts/Token.sol#751)
                - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,BEP20: transfer amount exceeds allowance)) (contracts/Token.sol#709-716)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4
INFO:Detectors:
Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/Token.sol#420) is too similar to IUniswapV2Router01.addLiqu
idity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/Token.sol#421)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
HIPPO.slitherConstructorVariables() (contracts/Token.sol#608-936) uses literals with too many digits:
        - numTokensSellToFee = 2000000 * 10 ** 18 (contracts/Token.sol#633)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
HIPPO._owner (contracts/Token.sol#645) should be immutable
HIPPO._totalSupply (contracts/Token.sol#620) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Slither:./contracts/Token.sol analyzed (12 contracts with 88 detectors), 46 result(s) found
```

Result => A static analysis of contract's source code has been performed using slither,

No major issues were found in the output

FUNCTIONAL TESTING

1- Adding liquidity (passed):

https://testnet.bscscan.com/tx/0xd3955fa8af71d3cc0736fd0bce7db6e48474ac99ee3ac1067239b711a0cf38b9

2- Buying when excluded (0% tax) (passed):

https://testnet.bscscan.com/tx/0x28bb41330f0bfec5fd9bc67847a9c4c95f96a9ebf2cd6e98294a90849dd68ed9

3- Selling when excluded (0% tax) (passed):

https://testnet.bscscan.com/tx/0x3bea7a77960dc7d055871c5d5f29b2084799c3498bd7ade6c5095ab91d649742

4- Transferring when excluded from fees (0% tax) (passed):

https://testnet.bscscan.com/tx/0xdcd4f69606430207ef977e5f70f77433be541bfa34babad80235cccff0555da9

5- Buying when not excluded from fees (tax 0-3%) (passed):

https://testnet.bscscan.com/tx/0x0c1fd75c5c8b2d6d76e252a8d450b4ce896a3129fc859d5bca7ec071e5c879fb

6- Selling when not excluded from fees (tax 0-3%) (passed):

https://testnet.bscscan.com/tx/0x60e2475ac4c210aa468baa6863f4e9d314fe31aa40fcae8d3dd5d92457568e21

FUNCTIONAL TESTING

7- Transferring when not excluded from fees (0-3% tax) (passed):

https://testnet.bscscan.com/tx/0x0dd9267fde1d84d5e12272c735f9e8ede000d2670bc2b7cdad33616895a51bda

8- Internal swap (BNB set to Marketing wallet) (passed):

https://testnet.bscscan.com/tx/0x60e2475ac4c210aa468baa6863f4e9d314fe31aa40fcae8d3dd5d92457568e21

High Risk

Centralization – Enabling Trades

Severity: High
function: enableTrading
Status: Resolved (contract owned by safu developer)
Overview:
The **launch** function permits only the contract owner to activate trading capabilities. Until this function is executed, no investors can buy, sell, or transfer their tokens. This places a high degree of control and centralization in the hands of the contract owner.

```
function enableTrading() external onlyOwner {
    require(!tradingEnabled, "Trading is already enabled");
    tradingEnabled = true;
    startTradingBlock = block.number;
}
```

Suggestion

To reduce centralization and potential manipulation, consider one of the following approaches:

1. Automatically enable trading after a specified condition, such as the completion of a presale, is met.
2. If manual activation is still desired, consider transferring the ownership of the contract to a trustworthy, third-party entity like a certified "PinkSale Safu" developer. This can provide investors with more confidence in the eventual activation of trading capabilities, mitigating concerns of potential bad faith actions by the original owner

DISCLAIMER

All the content provided in this document is for general information only and should not be used as financial advice or a reason to buy any investment. Team provides no guarantees against the sale of team tokens or the removal of liquidity by the project audited in this document. Always Do your own research and protect yourselves from being scammed. The Auditace team has audited this project for general information and only expresses their opinion based on similar projects and checks from popular diagnostic tools. Under no circumstances did Auditace receive a payment to manipulate those results or change the awarding badge that we will be adding in our website. Always Do your own research and protect yourselves from scams. This document should not be presented as a reason to buy or not buy any particular token. The Auditace team disclaims any liability for the resulting losses.

ABOUT AUDITACE

We specializes in providing thorough and reliable audits for Web3 projects. With a team of experienced professionals, we use cutting-edge technology and rigorous methodologies to evaluate the security and integrity of blockchain systems. We are committed to helping our clients ensure the safety and transparency of their digital assets and transactions.

https://auditace.tech/

https://t.me/Audit_Ace

https://twitter.com/auditace_

https://github.com/Audit-Ace