auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

Cheburashka Token

DATED : 03 FEB 23'

AUDIT SUMMARY

Project name – Cheburashka Token

Date: 03 February, 2023

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

Audit Status: Passed (Contract is developed by Pinksale safu dev

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	0	0	0	0	2
Acknowledged	0	0	0	0	0
Resolved	0	0	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 done on BSC Test network, each test has its transaction has attached to it.

3- Slither : Static Analysis

Testnet Link: all tests were done using this contract, tests are done on BSC Testnet

https://testnet.bscscan.com/token/0xa56c19D0ff296F0D25A88b6a0e52ceAe8CA7F0dD

Token Information

Token Name : Cheburashka Token

Token Symbol: CHT

Decimals: 9

Token Supply: 110,000,000

Token Address:
0x546Cc57679c6c68F8bF4fcdA2FCBB4E7099A0d12

Checksum:
ebaa52509eda53f48bbc96f5d90bb98b0e28ce8cd84fd1c36a61f7f4ff354a15

Owner:
0xc5D18c97363EA2CAf8b0286024960b68b5E143e6

Deployer:
0xc5D18c97363EA2CAf8b0286024960b68b5E143e6

TOKEN OVERVIEW

Fees:

Buy Fees: 0%

Sell Fees: 0%

Transfer Fees: 0%

Fees Privilige: None

Ownership : Owned

Minting: No mint function

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Priviliges: No

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

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

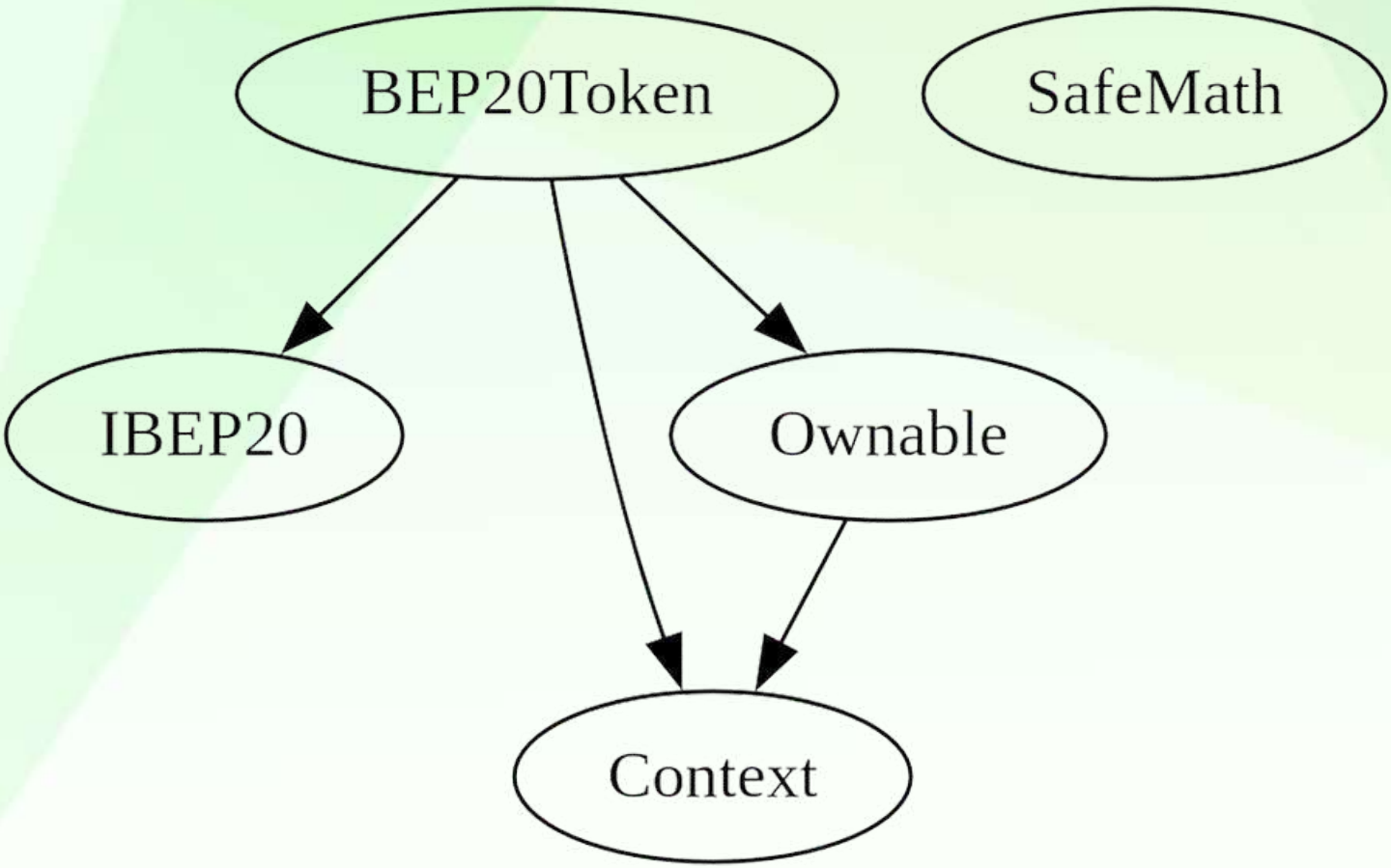
CLASSIFICATION OF RISK

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	0
Medium-Risk	0
Low-Risk	0
Gas Optimization / Suggestions	2

INHERITANCE TREE



POINTS TO NOTE

- Owner is not able to set tax (0% tax for buy, sell, transfer)
- Owner is not able to blacklist an arbitrary wallet
- Owner is not able to set max buy/sell/transfer amounts
- Owner is not able to disable trades
- Owner is not able to mint new tokens

CONTRACT ASSESMENT

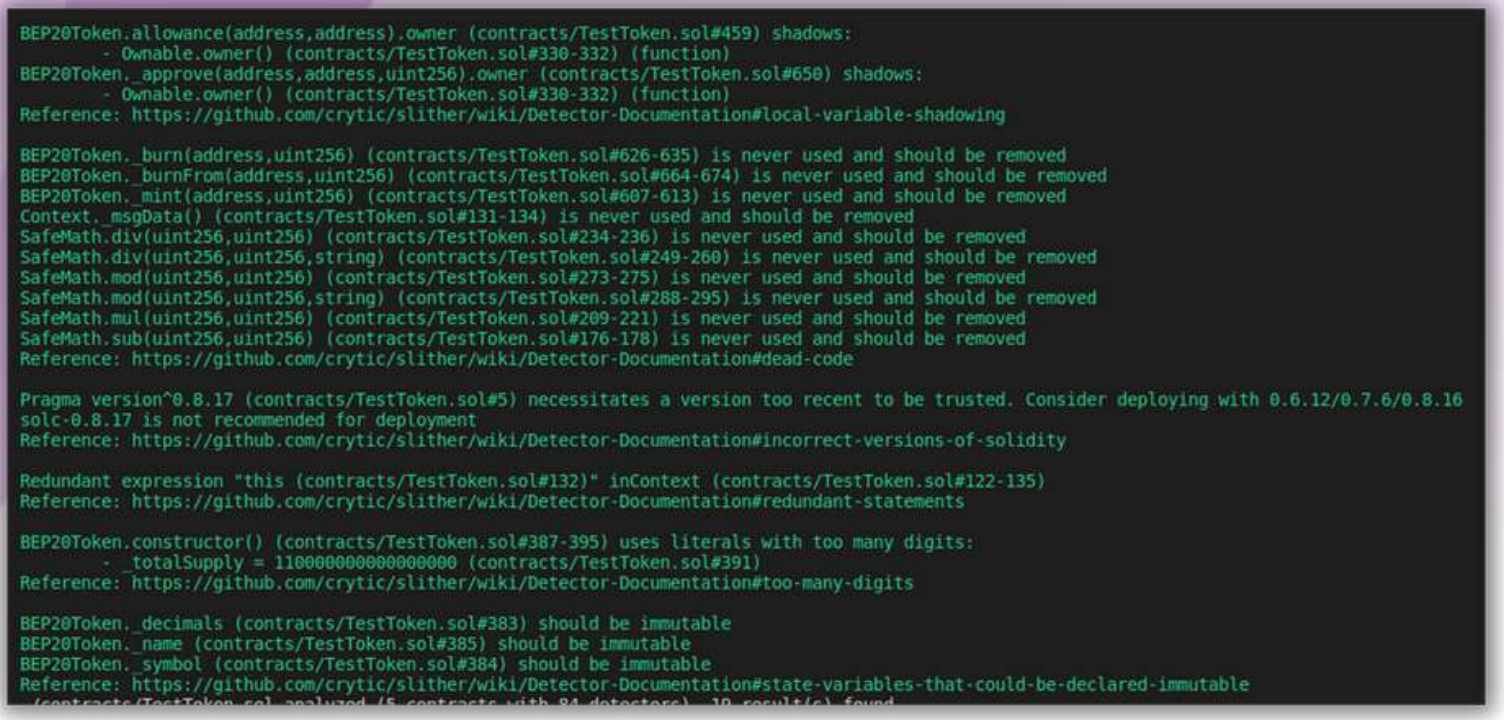
Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
IBEP20	Interface			
└	totalSupply	External ❗		NO❗
└	decimals	External ❗		NO❗
└	symbol	External ❗		NO❗
└	name	External ❗		NO❗
└	getOwner	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	transfer	External ❗	🛑	NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
Context	Implementation			
└	<Constructor>	Internal 🔒	🛑	
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
SafeMath	Library			
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
└	mod	Internal 🔒		
Ownable	Implementation	Context		
└	<Constructor>	Internal 🔒	🛑	
└	owner	Public ❗		NO❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
└	_transferOwnership	Internal 🔒	🛑	
BEP20Token	Implementation	Context, IBEP20, Ownable		
└	<Constructor>	Public ❗	🛑	NO❗
└	getOwner	External ❗		NO❗
└	decimals	External ❗		NO❗

CONTRACT ASSESMENT

└	symbol	External ❗			NO❗
└	name	External ❗			NO❗
└	totalSupply	External ❗			NO❗
└	balanceOf	External ❗			NO❗
└	transfer	External ❗	🛑		NO❗
└	allowance	External ❗			NO❗
└	approve	External ❗	🛑		NO❗
└	transferFrom	External ❗	🛑		NO❗
└	increaseAllowance	Public ❗	🛑		NO❗
└	decreaseAllowance	Public ❗	🛑		NO❗
└	_transfer	Internal 🔒	🛑		
└	_mint	Internal 🔒	🛑		
└	_burn	Internal 🔒	🛑		
└	_approve	Internal 🔒	🛑		
└	_burnFrom	Internal 🔒	🛑		

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

STATIC ANALYSIS



```
BEP20Token.allowance(address,address).owner (contracts/TestToken.sol#459) shadows:
        - Ownable.owner() (contracts/TestToken.sol#330-332) (function)
BEP20Token._approve(address,address,uint256).owner (contracts/TestToken.sol#650) shadows:
        - Ownable.owner() (contracts/TestToken.sol#330-332) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

BEP20Token._burn(address,uint256) (contracts/TestToken.sol#626-635) is never used and should be removed
BEP20Token._burnFrom(address,uint256) (contracts/TestToken.sol#664-674) is never used and should be removed
BEP20Token._mint(address,uint256) (contracts/TestToken.sol#607-613) is never used and should be removed
Context._msgData() (contracts/TestToken.sol#131-134) is never used and should be removed
SafeMath.div(uint256,uint256) (contracts/TestToken.sol#234-236) is never used and should be removed
SafeMath.div(uint256,uint256,string) (contracts/TestToken.sol#249-260) is never used and should be removed
SafeMath.mod(uint256,uint256) (contracts/TestToken.sol#273-275) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (contracts/TestToken.sol#288-295) is never used and should be removed
SafeMath.mul(uint256,uint256) (contracts/TestToken.sol#209-221) is never used and should be removed
SafeMath.sub(uint256,uint256) (contracts/TestToken.sol#176-178) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.17 (contracts/TestToken.sol#5) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Redundant expression "this (contracts/TestToken.sol#132)" inContext (contracts/TestToken.sol#122-135)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

BEP20Token.constructor() (contracts/TestToken.sol#387-395) uses literals with too many digits:
        - _totalSupply = 110000000000000000 (contracts/TestToken.sol#391)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

BEP20Token._decimals (contracts/TestToken.sol#383) should be immutable
BEP20Token._name (contracts/TestToken.sol#385) should be immutable
BEP20Token._symbol (contracts/TestToken.sol#384) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
```

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

Compiler version is outdated (0.5.16)

FUNCTIONAL TESTING

Router (PCS V2):
0xD99D1c33F9fC3444f8101754aBC46c52416550D1

0- Deploying (Passed):

https://testnet.bscscan.com/tx/0xd59fd47c7c23529fa96b5764021a5f2347c86e191ecf80cfe082c1d2d32d6f42

1- Adding Liquidity (Passed):
liquidity added on Pancakeswap V2:

https://testnet.bscscan.com/tx/0x0db75be0e4d8d952ed08ed0167f39f3c795b3219f5833069aa0f36e0167e8907

no issue were found on adding liquidity.

2- Buying(Passed):

https://testnet.bscscan.com/tx/0x7826e149dbf4804f8350ef6c240f1bdbf3ace8a560f0d77c751ffe57c4d36f4a

0% tax

3- Selling (Passed):

https://testnet.bscscan.com/tx/0x94374cb578ca86437fd13d6540c68a13c75a52a6b1ebddfcbd38301c985f19cd

0% tax

MANUAL TESTING

Suggestions

- Use latest compiler version, current version is 0.5.16 which is a very old and outdated compiler.
- Upgrade compiler version to > 0.8.0 and delete SafeMath from the contract

Social Media Overview

Here are the Social Media Accounts of

Cheburashka Token

https://t.me/chebburashka_official

https://twitter.com/Cheburashka0_

https://chebburashka.com/

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