auditace.tech

AuditAce

FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

BetaBp

DATED : 17 MAR 23'

AUDIT SUMMARY

Project name – BetaBP

Date: 17 March, 2023

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

Audit Status: **Passed with Medium Risk**

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	0	0	2	0	0
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- Goerli Testnet:
all tests were done on Goerli Testnet 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://goerli.etherscan.io/token/0x7909bea1782cc6959aa63197af24fa1f0a3eafc5

Token Information

Token Name : betaBP

Token Symbol: betaBP

Decimals: 18

Token Supply: 1,000,000,000

Token Address: Not deployed

Checksum:
97a5b17b0fd90ece89930aeff76cc32fef1a6f14

Owner: Not deployed

TOKEN OVERVIEW

Fees:

Buy Fees: 10%

Sell Fees: 10%

Transfer Fees:10%

Fees Privilige: Owner

Ownership : Owned

Minting: No mint function

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Priviliges: including and excluding form fee - changing max wallet and buy/transfer

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.

auditace.tech

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

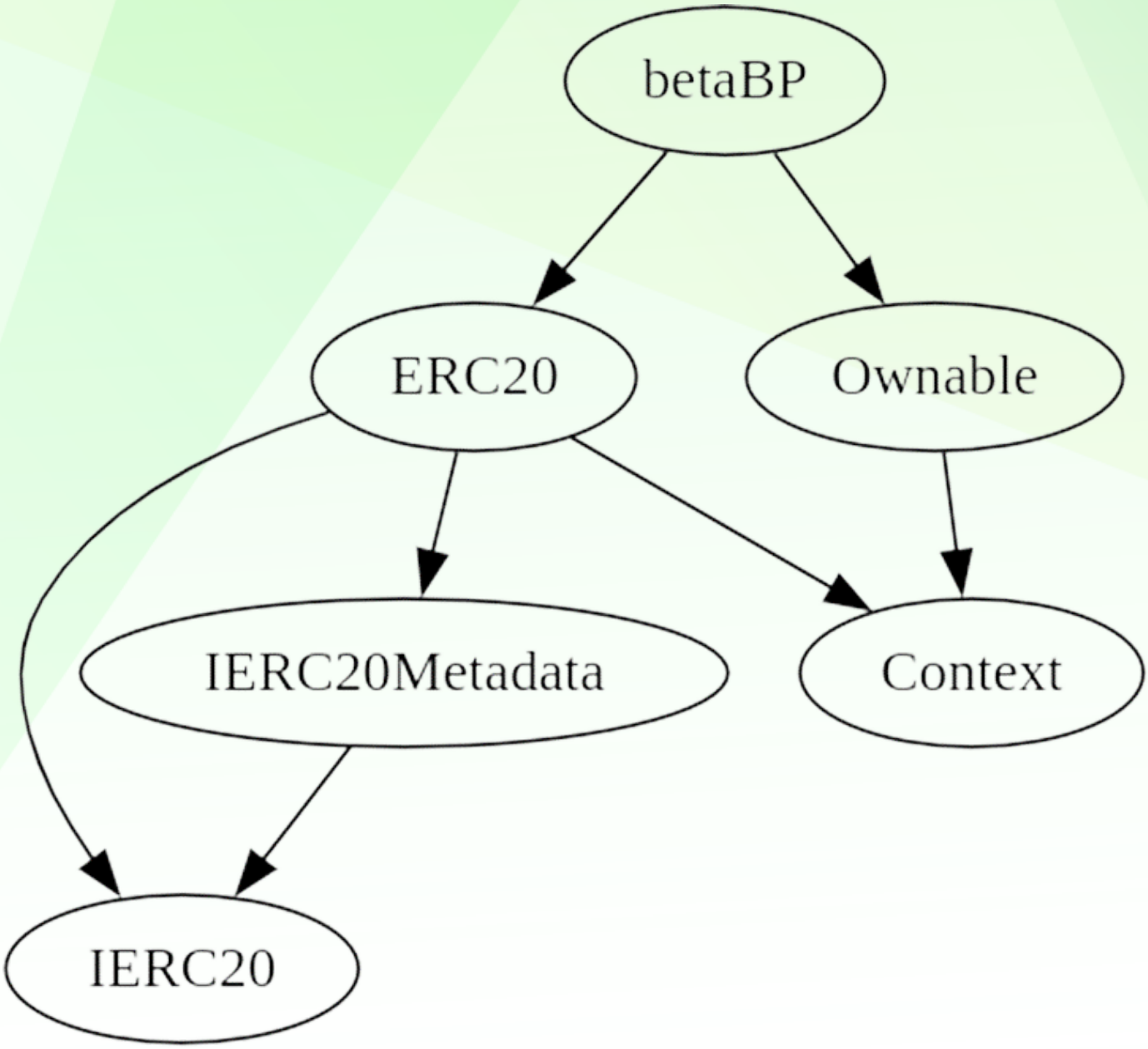
CLASSIFICATION OF RISK

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	2
Low-Risk	0
Gas Optimization / Suggestions	0

INHERITANCE TREE



POINTS TO NOTE

- Please note that contract would have 0.03% max buy and wallet at time of launch
- Owner is able to set buy and sell taxes each one up to 15% (30% total)
- Owner is able to set max buy/transfer amount with a minimum of 0.5% of supply
- Owner must enable trading for investors in order for them to be able to trade
- Owner is not able to blacklist an arbitrary wallet
- Owner is not able to disable trades
- Owner is not able to mint new tokens

OVERVIEW

betaBP has a total supply of 1 billion tokens. It has a public launch function, allowing the owner to change tax rates and maximum transaction limits upon launch. The contract also impose taxes on transactions for liquidity and honorarium purposes. Tokens can be swapped for ETH, and honorarium and liquidity funds can be withdrawn by the owner. Additionally, the contract allows for the management of fees, whitelists, and wallet restrictions.

We have recently completed a thorough security audit of the token contract in question. During the course of our audit, we identified several issues that raised concerns regarding the contract's security and overall stability. Working closely with the development team, we methodically addressed each of these issues to ensure that the token contract meets the highest standards of safety and reliability

TOKEN DISTRIBUTION

It should be noted that the owner currently holds 100% of the total supply. However, information about the distribution of these tokens is not available, and it is recommended that investors exercise caution when considering this aspect.

Trades will not be enabled right away

Owner of the contract must call “enableTrading” function in order to enable buys, sells and transfers for non-authorized wallets. Since contract is developed and owned by pinksale’s safu dev its guaranteed that trades will be enabled.

CONTRACT ASSESMENT

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
betaBP	Implementation	ERC20, Ownable		
└	publicLaunch	Public ❗	🛑	onlyOwner
└	<Constructor>	Public ❗	🛑	ERC20
└	_transfer	Internal 🔒	🛑	
└	swapAndLiquify	Internal 🔒	🛑	
└	withdrawHonorarium	External ❗	🛑	onlyOwner
└	withdrawLiquidity	Public ❗	🛑	onlyOwner
└	excludeFromFee	External ❗	🛑	onlyOwner
└	batchExcludeFromFee	External ❗	🛑	onlyOwner
└	addToWhitelist	External ❗	🛑	onlyOwner
└	addBatchToWhitelist	External ❗	🛑	onlyOwner
└	_swapTokensForEth	Internal 🔒	🛑	
└	_getETHAmountsOut	Internal 🔒		
└	updatePair	External ❗	🛑	onlyOwner
└	updateHonorariumWallet	Public ❗	🛑	onlyOwner
└	updateLiquidityWallet	Public ❗	🛑	onlyOwner
└	updateTaxForLiquidityAndHonorarium	Public ❗	🛑	onlyOwner
└	updateMaxTxAmount	Public ❗	🛑	onlyOwner
└	updateMaxWalletAmount	Public ❗	🛑	onlyOwner
└	withdrawETH	External ❗	🛑	onlyOwner
└	withdrawTokens	External ❗	🛑	onlyOwner
└	<Receive Ether>	External ❗	💵	NO❗
IERC20	Interface			
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	transfer	External ❗	🛑	NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
ERC20	Implementation	Context, IERC20, IERC20Metadata		
└	<Constructor>	Public ❗	🛑	NO❗
└	name	Public ❗		NO❗
└	symbol	Public ❗		NO❗
└	decimals	Public ❗		NO❗
└	totalSupply	Public ❗		NO❗
└	balanceOf	Public ❗		NO❗

CONTRACT ASSESMENT

└	transfer	Public ❗	🛑	NO❗
└	allowance	Public ❗		NO❗
└	approve	Public ❗	🛑	NO❗
└	transferFrom	Public ❗	🛑	NO❗
└	increaseAllowance	Public ❗	🛑	NO❗
└	decreaseAllowance	Public ❗	🛑	NO❗
└	_transfer	Internal 🔒	🛑	
└	_mint	Internal 🔒	🛑	
└	_burn	Internal 🔒	🛑	
└	_approve	Internal 🔒	🛑	
└	_spendAllowance	Internal 🔒	🛑	
└	_beforeTokenTransfer	Internal 🔒	🛑	
└	_afterTokenTransfer	Internal 🔒	🛑	
IERC20Metadata	Interface	IERC20		
└	name	External ❗		NO❗
└	symbol	External ❗		NO❗
└	decimals	External ❗		NO❗
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
IUniswapV2Router02	Interface	IUniswapV2Router01		
└	removeLiquidityETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
└	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
└	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗	💵	NO❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	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❗

CONTRACT ASSESMENT

└	swapTokensForExactETH	External ❗	🛑	NO❗
└	swapExactTokensForETH	External ❗	🛑	NO❗
└	swapETHForExactTokens	External ❗	💵	NO❗
└	quote	External ❗		NO❗
└	getAmountOut	External ❗		NO❗
└	getAmountIn	External ❗		NO❗
└	getAmountsOut	External ❗		NO❗
└	getAmountsIn	External ❗		NO❗
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❗
Ownable	Implementation	Context		
└	<Constructor>	Public ❗	🛑	NO❗
└	owner	Public ❗		NO❗
└	_checkOwner	Internal 🔒		
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
└	_transferOwnership	Internal 🔒	🛑	

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

STATIC ANALYSIS

```
Context._msgData() (contracts/Token.sol#30-33) is never used and should be removed
ERC20._burn(address,uint256) (contracts/Token.sol#274-289) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.17 (contracts/Token.sol#23) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
solc-0.8.19 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (contracts/Token.sol#114-125):
        - (success) = recipient.call{value: amount}() (contracts/Token.sol#120)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Redundant expression "this (contracts/Token.sol#31)" inContext (contracts/Token.sol#25-34)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
```

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

No issues found

FUNCTIONAL TESTING

Router (PCS V2):
0xD99D1c33F9fC3444f8101754aBC46c52416550D1

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

https://goerli.etherscan.io/tx/0x878322d94d508675431631f3763c7f03214eb1283a017250689ee32e7bf43636

2- Buying when trading not enabled (owner%)(Passed):

https://goerli.etherscan.io/tx/0x7dd9b6a347535ac245cc037c2904c707a27416871daa20ee5e5084aa70915652

3- Selling when trading not enabled (0%)(Passed):

https://goerli.etherscan.io/tx/0x97e9f373c3bbf641d26609d31607316add42bdd289a2e5276b882c12ab5a9af8

4- Transferring when trading not enabled (0% tax) (passed):

https://goerli.etherscan.io/tx/0x3bb69743341d8d82e4488b8894677f961cb2d97210d155e49a58eb86d1c92d8c

5- Buying when trading enabled (10% tax) (passed):

https://goerli.etherscan.io/tx/0x3899f26f05cfc4530db93f71516af81e96899a39b92a6bef0bdb77ead5e6958f

FUNCTIONAL TESTING

6- Selling when trading enabled (10% tax) (passed):

https://goerli.etherscan.io/tx/0xa742515b72f43fb3124b63794e853411a48e095b41317bab8a5b288b591d393f

7- Transferring when trading enabled (0% tax) (passed):

https://goerli.etherscan.io/tx/0x67c2f7915b5beb3b0be3b9216b8409bbf10a9b0144dc0fb42e76dde7b158e472

8- Withdrawing collected holonarium tax (passed):

https://goerli.etherscan.io/tx/0x04c0d8ce86b11f2cc3a98b8a762f983164900d35c335f8211210c38c0514e377

9- Withdrawing collected token and ETH to be added to liquidity (passed):

https://goerli.etherscan.io/tx/0xec608e4e895e0df7c37b2e6808215c1a5b238c3424f6fb00786017ad09a05646

MANUAL TESTING

Issue: Centralized Control over Maximum Wallet and Transaction Amounts
Severity: **Medium**
Function: **updateMaxWalletAmount, updateMaxTxAmount**
Type: Centralization
Line: ---
Range: 0.5% - 10% of total supply

Overview:
The **updateMaxWalletAmount** and **updateMaxTxAmount** functions allow the contract owner to modify the maximum wallet amount and the maximum transaction amount, respectively. While these functions provide flexibility, they also introduce a degree of centralization and potential manipulation, as the owner has the power to change these limits at any time such as Centralized control.
The functions can only be called by the contract owner, which means that the owner has the power to control the maximum wallet amount and the maximum transaction amount. This centralized control can potentially harm the decentralization aspect of the token and introduce manipulation risks.

MANUAL TESTING

Issue: Centralized Control over Tax Rates and Exceeding PinkSale SAFU Criteria
Severity: **Medium**
Function: **updateTaxForLiquidityAndHonorarium**
Type: Centralization
Lines: ---

Overview:
The **updateTaxForLiquidityAndHonorarium** function allows the contract owner to modify the tax rates for liquidity and honorarium. This introduces centralized control over these rates and can potentially exceed the PinkSale SAFU criteria when both taxes are set to their maximum allowed values:

- Centralized control: The function can only be called by the contract owner, which means that the owner has the power to control the tax rates for liquidity and honorarium. This centralized control can potentially harm the decentralization aspect of the token and introduce manipulation risks.

- Exceeding PinkSale SAFU criteria: The PinkSale platform's SAFU criteria recommend that the total tax should not exceed 25%. However, in this implementation, the total tax can be set as high as 30% if both the liquidity and honorarium taxes are set to their maximum allowed value of 15%. This exceeds the recommended criteria, which can impact the token's credibility and acceptance on the PinkSale platform.

Recommendation:

- Consider implementing a decentralized governance mechanism that allows token holders to have a say in decisions regarding the tax rates for liquidity and honorarium. This can be achieved through proposals and voting based on token holdings.

- Modify the requirements in the function to ensure that the total tax does not exceed the PinkSale SAFU criteria of 25%. You can achieve this by adding a check to ensure the sum of _taxForLiquidity and _taxForHonorarium does not exceed 25%.

MANUAL TESTING

- Alternatively, introduce a time-based lock or cool-down period after each change to limit the owner's ability to make frequent adjustments to these tax rates. This can help reduce the potential for manipulation and give users more confidence in the stability of the token's parameters.

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