auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

BETCOIN CASH

DATED : 28 JAN 23'

AUDIT SUMMARY

Project name – BETCOIN CASH

Date: 28 January , 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	1
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 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://testnet.bscscan.com/token/0xd40225b839a8a0e014fdd9cf1dab28cf7642d040

Token Information

Token Name : Betcoin Cash

Token Symbol: BETC

Decimals: 9

Token Address:
0xbf7517cc8cb82bb2f1aebf6a693ed3fe602ec218

Checksum:
2F7ADE6C239719849C39B26D1034738849B87C92DBEA5A42FBFBBC62C4D0D2EE

Deployer:
0xcb64678c574c1e3a900a0010bc98ef2f79099c57

Owner:
0xcb64678c574c1e3a900a0010bc98ef2f79099c57

TOKEN OVERVIEW

Fees:

Buy Fees: 0.5%

Sell Fees: 3%

Transfer Fees: 3%

Fees Privilige: Owner

Ownership : Owned

Minting: No mint function

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Priviliges: None

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

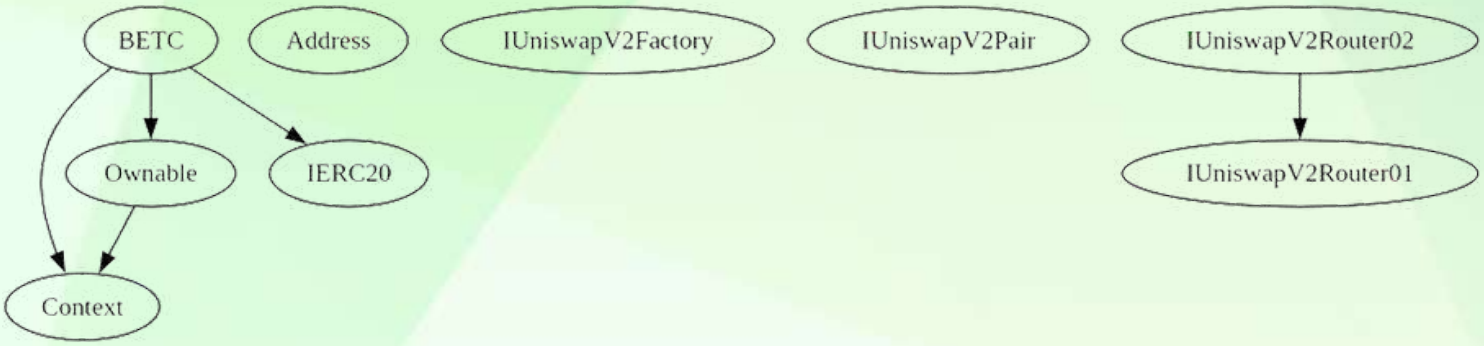
CLASSIFICATION OF RISK

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	1

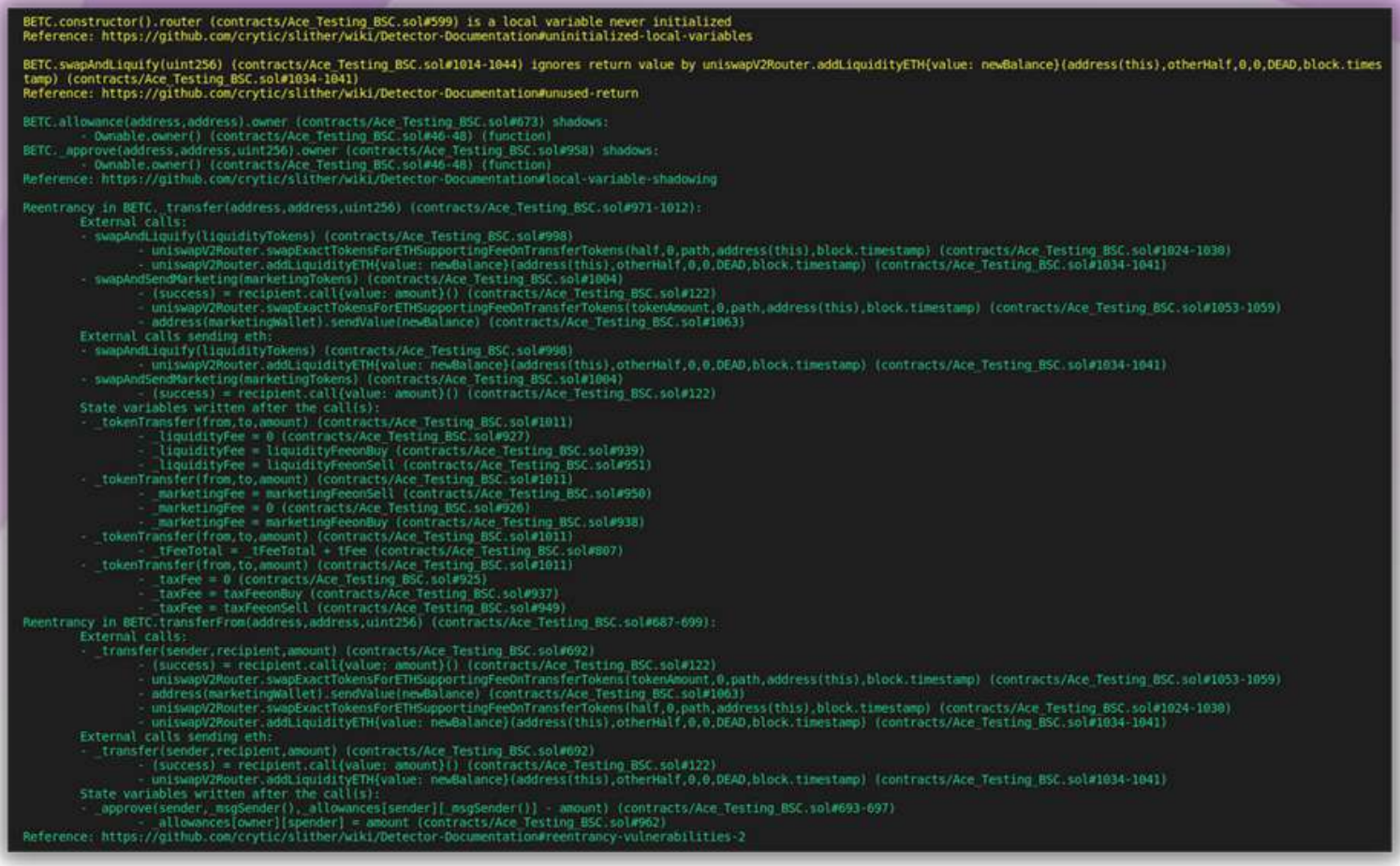
INHERITANCE TREE



POINTS TO NOTE

- Owner is not able to set taxes over 25% (Buy + Sell = 50% maximum tax)
- 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

STATIC ANALYSIS



```
BETC.constructor().router (contracts/Ace_Testing_BSC.sol#599) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

BETC.swapAndLiquify(uint256) (contracts/Ace_Testing_BSC.sol#1014-1044) ignores return value by uniswapV2Router.addLiquidityETH{value: newBalance}(address(this),otherHalf,0,0,DEAD,block.times
tamp) (contracts/Ace_Testing_BSC.sol#1034-1041)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

BETC.allowance(address,address).owner (contracts/Ace_Testing_BSC.sol#673) shadows:
        - Ownable.owner() (contracts/Ace_Testing_BSC.sol#46-48) (function)
BETC._approve(address,address,uint256).owner (contracts/Ace_Testing_BSC.sol#958) shadows:
        - Ownable.owner() (contracts/Ace_Testing_BSC.sol#46-48) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

Reentrancy in BETC._transfer(address,address,uint256) (contracts/Ace_Testing_BSC.sol#971-1012):
        External calls:
        - swapAndLiquify(liquidityTokens) (contracts/Ace_Testing_BSC.sol#998)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(half,0,path,address(this),block.timestamp) (contracts/Ace_Testing_BSC.sol#1024-1030)
                - uniswapV2Router.addLiquidityETH{value: newBalance}(address(this),otherHalf,0,0,DEAD,block.timestamp) (contracts/Ace_Testing_BSC.sol#1034-1041)
        - swapAndSendMarketing(marketingTokens) (contracts/Ace_Testing_BSC.sol#1004)
                - (success) = recipient.call{value: amount}() (contracts/Ace_Testing_BSC.sol#122)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/Ace_Testing_BSC.sol#1053-1059)
                - address(marketingWallet).sendValue(newBalance) (contracts/Ace_Testing_BSC.sol#1063)
        External calls sending eth:
        - swapAndLiquify(liquidityTokens) (contracts/Ace_Testing_BSC.sol#998)
                - uniswapV2Router.addLiquidityETH{value: newBalance}(address(this),otherHalf,0,0,DEAD,block.timestamp) (contracts/Ace_Testing_BSC.sol#1034-1041)
        - swapAndSendMarketing(marketingTokens) (contracts/Ace_Testing_BSC.sol#1004)
                - (success) = recipient.call{value: amount}() (contracts/Ace_Testing_BSC.sol#122)
        State variables written after the call(s):
        - _tokenTransfer(from,to,amount) (contracts/Ace_Testing_BSC.sol#1011)
                - liquidityFee = 0 (contracts/Ace_Testing_BSC.sol#927)
                - liquidityFee = liquidityFeeonBuy (contracts/Ace_Testing_BSC.sol#939)
                - liquidityFee = liquidityFeeonSell (contracts/Ace_Testing_BSC.sol#951)
        - _tokenTransfer(from,to,amount) (contracts/Ace_Testing_BSC.sol#1011)
                - marketingFee = marketingFeeonSell (contracts/Ace_Testing_BSC.sol#950)
                - marketingFee = 0 (contracts/Ace_Testing_BSC.sol#926)
                - marketingFee = marketingFeeonBuy (contracts/Ace_Testing_BSC.sol#938)
        - _tokenTransfer(from,to,amount) (contracts/Ace_Testing_BSC.sol#1011)
                - _tFeeTotal = _tFeeTotal + tFee (contracts/Ace_Testing_BSC.sol#807)
        - _tokenTransfer(from,to,amount) (contracts/Ace_Testing_BSC.sol#1011)
                - taxFee = 0 (contracts/Ace_Testing_BSC.sol#925)
                - taxFee = taxFeeonBuy (contracts/Ace_Testing_BSC.sol#937)
                - taxFee = taxFeeonSell (contracts/Ace_Testing_BSC.sol#949)
Reentrancy in BETC.transferFrom(address,address,uint256) (contracts/Ace_Testing_BSC.sol#687-699):
        External calls:
        - _transfer(sender,recipient,amount) (contracts/Ace_Testing_BSC.sol#692)
                - (success) = recipient.call{value: amount}() (contracts/Ace_Testing_BSC.sol#122)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/Ace_Testing_BSC.sol#1053-1059)
                - address(marketingWallet).sendValue(newBalance) (contracts/Ace_Testing_BSC.sol#1063)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(half,0,path,address(this),block.timestamp) (contracts/Ace_Testing_BSC.sol#1024-1030)
                - uniswapV2Router.addLiquidityETH{value: newBalance}(address(this),otherHalf,0,0,DEAD,block.timestamp) (contracts/Ace_Testing_BSC.sol#1034-1041)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (contracts/Ace_Testing_BSC.sol#692)
                - (success) = recipient.call{value: amount}() (contracts/Ace_Testing_BSC.sol#122)
                - uniswapV2Router.addLiquidityETH{value: newBalance}(address(this),otherHalf,0,0,DEAD,block.timestamp) (contracts/Ace_Testing_BSC.sol#1034-1041)
        State variables written after the call(s):
        - _approve(sender,_msgSender(),_allowances[sender][_msgSender()] - amount) (contracts/Ace_Testing_BSC.sol#693-697)
                - _allowances[owner][spender] = amount (contracts/Ace_Testing_BSC.sol#962)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```

Result => A static analysis of contract's source code has been performed using slither, no major issues were found (except some "minor impact" suggestions)

FUNCTIONAL TESTING

Router (PCS V2):
0xD99D1c33F9fC3444f8101754aBC46c52416550D1

0- Deploying (Passed):

https://testnet.bscscan.com/tx/0x2612047b1b8fb63b05b4f98c2ff98d3ea5505b1f8c5d1d785dc5d57cc1eac71a

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

https://testnet.bscscan.com/tx/0x12f1ed001c1dc13baeef9b3a95de0a24e3d190b3f26565cd029793329d8a52e6

no issue were found on adding liquidity.

2- Enabling trade for public (Passed):

https://testnet.bscscan.com/tx/0x2eb607828b3d0378a36bef48221500643d105f441c9068a8065dfc3901389909

3- Excluding deployer's wallet from taxes (to test taxes) (Passed):

https://testnet.bscscan.com/tx/0x87fb5b0c5024eb6cc1b82d232e5417a623f624c312fd4a240f93369b6101d94e

FUNCTIONAL TESTING

4- Enabling swap and liquify (Passed):

https://testnet.bscscan.com/tx/0x0c0814ddf4db23589bbb7313a99b83d3114f480e178236a279d5810acf507b90

5- Setting buy and sell fees to max (25%) (Passed):

https://testnet.bscscan.com/tx/0x9b65b608afc1e2b3706600ea9b1017dd9220d0b2977010f22766c2cba29753c6

https://testnet.bscscan.com/tx/0xb0a801f8894979fb2e86d82b74818e6c87041543a4146e522c06122fca6a1768

6- Buying from a non-whitelisted wallet (Passed):
https://testnet.bscscan.com/tx/0x7f1a84e6a692b55fea1c47faa491d61fefbf53288c476501dcbfc0e6536cd4d2

25% tax applied

5- Selling from a non-whitelisted wallet (Passed):

https://testnet.bscscan.com/tx/0x9ed2eee007a6bc4b9b15abf2ef01fc473905fe710b47c8a4c8d5bfb3882cbae0

25% tax applied

FUNCTIONAL TESTING

6- **Internal Swap (Passed):**

taxes were swapped to BNB and sent to marketing wallet which can be seen here:

https://testnet.bscscan.com/address/0x024238ae95bf8e4361ba58ca018e84c12c654509#internaltx

7- Auto Liquidity (Passed):

LP tokens generated and sent to dead wallet which can be seen here:

https://testnet.bscscan.com/token/0x72bf7c4286c90956d611e73fe704bae7305850ba?a=0x000000000000000000000000000000000000dead

8- Transferring from non-whitelisted to non-whitelisted (Passed):

tried to transfer 100,000,000 tokens but 75,100,000 were sent (25% transfer tax)

https://testnet.bscscan.com/tx/0x8614e4ce9b4e5c43303403e3743e52f20f74b41b27fc06e7710655f4d987636b

FUNCTIONAL TESTING

9- Transferring while transfer taxes are off (wallet to wallet transfer without fee) (Passed):

disabling transfer fees:

https://testnet.bscscan.com/tx/0x0db8bfd374a567d37e2eac3ee16bfc420bb8487b3814694d0b5336e88cc864ff

transferring 100,000,000 tokens from a non-whitelisted wallet to a non-whitelisted wallet without fee:

https://testnet.bscscan.com/tx/0x7655acc43dc39669c6b1ed4dc6c025aedb79ca776dc55c3b7f414350093fa3d5

MANUAL TESTING

Suggestions

- **Emit a transfer event at _takeMarketing and _takeLiquidity functions**

Social Media Overview

Here are the Social Media Accounts of

BetCoin

https://t.me/BetcoinCashOfficial

https://twitter.com/Betcoin_Cash

https://betcoincash.io/

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