AuditAce

FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

CyberVerse Land

DATED : 23 JAN 23'

AUDIT SUMMARY

Project name – CyberVerse Land

Date: 23 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	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:
all tests were done on Goerli 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 goerli

https://goerli.etherscan.io/token/0x2a0ce16b02c188cf1999df7c6cf5fdd4815559c8c1

Token Information

Token Name : Cyberverseland

Token Symbol: CYBERVERSE

Decimals: 18

Token Address:
0x0911BBfF1F00E94a1D3FcFa331E890F05337CD4B

Checksum:
f0e4c2f76c58916ec258f246851bea091d14d4247a2fc3e18694461b1816e13b

Deployer:
0x7271ed7709d8bB6f83766b76Db276b50e057d2b9

Owner:
0x7271ed7709d8bB6f83766b76Db276b50e057d2b9

TOKEN OVERVIEW

Fees:

Buy Fees: 1%

Sell Fees: 1%

Transfer Fees: 0%

Fees Privilige: Owner

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.

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

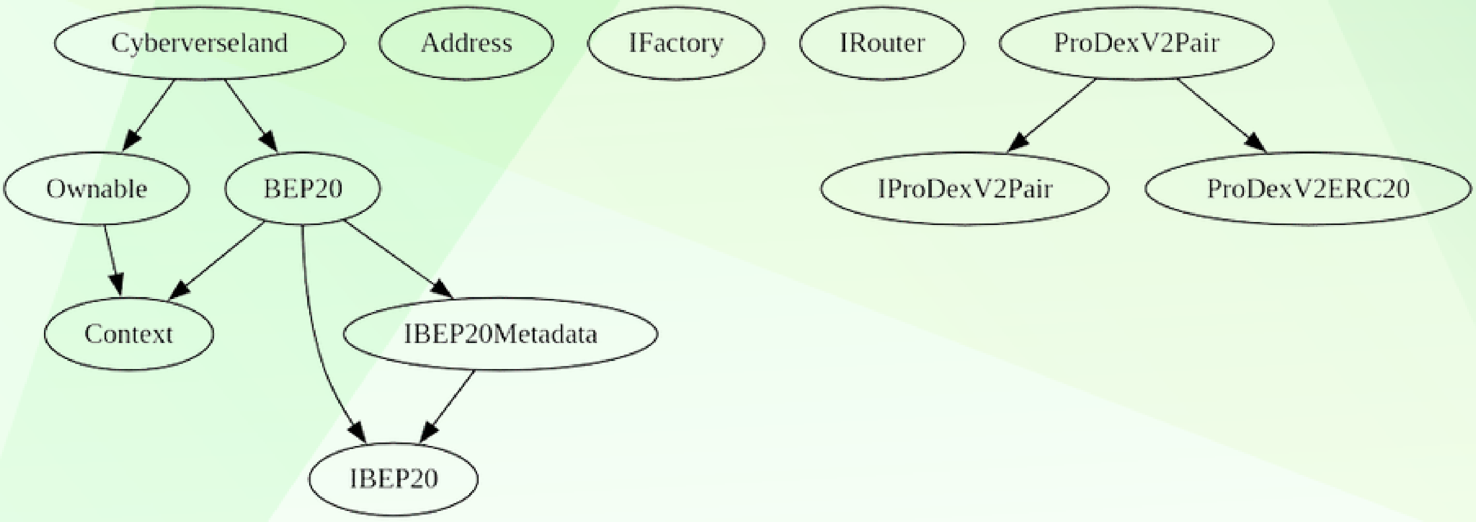
CLASSIFICATION OF RISK

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	0
◆ Low-Risk	0
◆ Gas Optimization / Suggestions	0

INHERITANCE TREE



POINTS TO NOTE

- Owner is not able to change taxes (1% buy and 1% sell, 0% 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**  |
||||||
| **Context** | Implementation |  |||
| └ | _msgSender | Internal 🔒 |   | |
| └ | _msgData | Internal 🔒 |   | |
||||||
| **IBEP20** | Interface |  |||
| └ | totalSupply | External ❗ |   |NO❗ |
| └ | balanceOf | External ❗ |   |NO❗ |
| └ | transfer | External ❗ | 🛑  |NO❗ |
| └ | allowance | External ❗ |   |NO❗ |
| └ | approve | External ❗ | 🛑  |NO❗ |
| └ | transferFrom | External ❗ | 🛑  |NO❗ |
||||||
| **IBEP20Metadata** | Interface | IBEP20 |||
| └ | name | External ❗ |   |NO❗ |
| └ | symbol | External ❗ |   |NO❗ |
| └ | decimals | External ❗ |   |NO❗ |
||||||
| **BEP20** | Implementation | Context, IBEP20, IBEP20Metadata |||
| └ | <Constructor> | Public ❗ | 🛑  |NO❗ |
| └ | name | Public ❗ |   |NO❗ |
| └ | symbol | Public ❗ |   |NO❗ |
| └ | decimals | Public ❗ |   |NO❗ |
| └ | totalSupply | Public ❗ |   |NO❗ |
| └ | balanceOf | Public ❗ |   |NO❗ |
| └ | transfer | Public ❗ | 🛑  |NO❗ |
| └ | allowance | Public ❗ |   |NO❗ |
| └ | approve | Public ❗ | 🛑  |NO❗ |
| └ | transferFrom | Public ❗ | 🛑  |NO❗ |
| └ | increaseAllowance | Public ❗ | 🛑  |NO❗ |
| └ | decreaseAllowance | Public ❗ | 🛑  |NO❗ |
| └ | _transfer | Internal 🔒 | 🛑  | |
| └ | _tokengeneration | Internal 🔒 | 🛑  | |
| └ | _approve | Internal 🔒 | 🛑  | |
||||||
| **Address** | Library |  |||
| └ | sendValue | Internal 🔒 | 🛑  | |
||||||
| **Ownable** | Implementation | Context |||
```

CONTRACT ASSESMENT

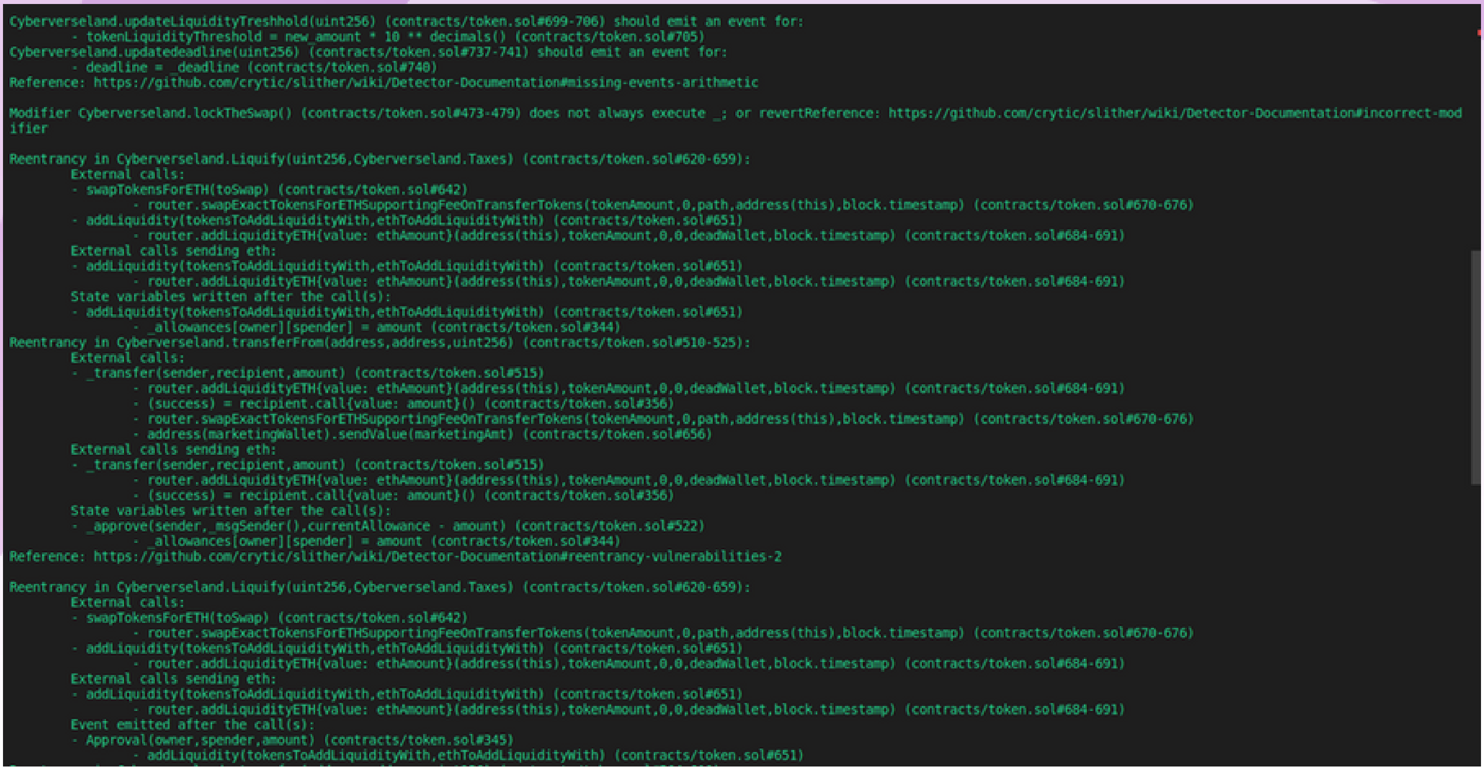
```
| └ | <Constructor> | Public ❗️ | 🛑  |NO❗️ |
| └ | owner | Public ❗️ |   |NO❗️ |
| └ | renounceOwnership | Public ❗️ | 🛑  | onlyOwner |
| └ | transferOwnership | Public ❗️ | 🛑  | onlyOwner |
| └ | _setOwner | Private 🔐 | 🛑  | |
||||||
| **IFactory** | Interface |  |||
| └ | createPair | External ❗️ | 🛑  |NO❗️ |
||||||
| **IRouter** | Interface |  |||
| └ | factory | External ❗️ |   |NO❗️ |
| └ | WETH | External ❗️ |   |NO❗️ |
| └ | addLiquidityETH | External ❗️ |  💵 |NO❗️ |
| └ | swapExactTokensForETHSupportingFeeOnTransferTokens | External ❗️ | 🛑  |NO❗️ |
||||||
| **Cyberverseland** | Implementation | BEP20, Ownable |||
| └ | <Constructor> | Public ❗️ | 🛑  | BEP20 |
| └ | approve | Public ❗️ | 🛑  |NO❗️ |
| └ | transferFrom | Public ❗️ | 🛑  |NO❗️ |
| └ | increaseAllowance | Public ❗️ | 🛑  |NO❗️ |
| └ | decreaseAllowance | Public ❗️ | 🛑  |NO❗️ |
| └ | transfer | Public ❗️ | 🛑  |NO❗️ |
| └ | _transfer | Internal 🔒 | 🛑  | |
| └ | Liquify | Private 🔐 | 🛑  | lockTheSwap |
| └ | swapTokensForETH | Private 🔐 | 🛑  | |
| └ | addLiquidity | Private 🔐 | 🛑  | |
| └ | updateLiquidityProvide | External ❗️ | 🛑  | onlyOwner |
| └ | updateLiquidityTreshhold | External ❗️ | 🛑  | onlyOwner |
| └ | SetBuyTaxes | External ❗️ | 🛑  | onlyOwner |
| └ | SetSellTaxes | External ❗️ | 🛑  | onlyOwner |
| └ | EnableTrading | External ❗️ | 🛑  | onlyOwner |
| └ | updatedeadline | External ❗️ | 🛑  | onlyOwner |
| └ | updateMarketingWallet | External ❗️ | 🛑  | onlyOwner |
| └ | updateExemptFee | External ❗️ | 🛑  | onlyOwner |
| └ | bulkExemptFee | External ❗️ | 🛑  | onlyOwner |
| └ | rescueBNB | External ❗️ | 🛑  | onlyOwner |
| └ | rescueBSC20 | External ❗️ | 🛑  | onlyOwner |
| └ | <Receive Ether> | External ❗️ |  💵 |NO❗️ |
```

CONTRACT ASSESMENT

ProDexV2Pair	Implementation	IProDexV2Pair, ProDexV2ERC20		
└	getReserves	Public ❗		NO❗
└	_safeTransfer	Private 🔐	🛑	
└	<Constructor>	Public ❗	🛑	NO❗
└	initialize	External ❗	🛑	NO❗
└	_update	Private 🔐	🛑	
└	_mintFee	Private 🔐	🛑	
└	mint	External ❗	🛑	lock
└	burn	External ❗	🛑	lock
└	swap	External ❗	🛑	lock
└	skim	External ❗	🛑	lock
└	sync	External ❗	🛑	lock

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

STATIC ANALYSIS

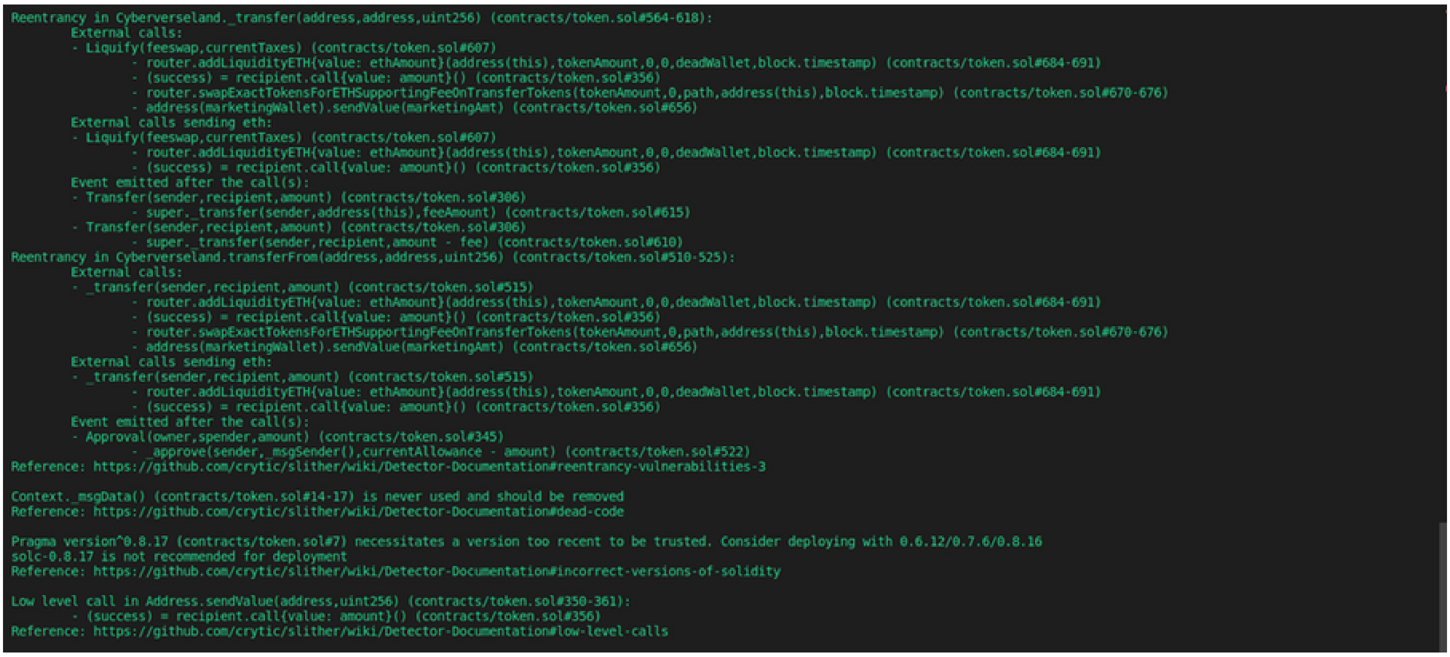


```
Cyberverseland.updateLiquidityTreshhold(uint256) (contracts/token.sol#699-706) should emit an event for:
        - tokenLiquidityThreshold = new_amount * 10 ** decimals() (contracts/token.sol#705)
Cyberverseland.updatedeadline(uint256) (contracts/token.sol#737-741) should emit an event for:
        - deadline = _deadline (contracts/token.sol#740)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

Modifier Cyberverseland.lockTheSwap() (contracts/token.sol#473-479) does not always execute _; or revertReference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-mod
ifier

Reentrancy in Cyberverseland.Liquify(uint256,Cyberverseland.Taxes) (contracts/token.sol#620-659):
        External calls:
        - swapTokensForETH(toSwap) (contracts/token.sol#642)
                - router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/token.sol#670-676)
        - addLiquidity(tokensToAddLiquidityWith,ethToAddLiquidityWith) (contracts/token.sol#651)
                - router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,deadWallet,block.timestamp) (contracts/token.sol#684-691)
        External calls sending eth:
        - addLiquidity(tokensToAddLiquidityWith,ethToAddLiquidityWith) (contracts/token.sol#651)
                - router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,deadWallet,block.timestamp) (contracts/token.sol#684-691)
        State variables written after the call(s):
        - addLiquidity(tokensToAddLiquidityWith,ethToAddLiquidityWith) (contracts/token.sol#651)
                - _allowances[owner][spender] = amount (contracts/token.sol#344)
Reentrancy in Cyberverseland.transferFrom(address,address,uint256) (contracts/token.sol#510-525):
        External calls:
        - _transfer(sender,recipient,amount) (contracts/token.sol#515)
                - router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,deadWallet,block.timestamp) (contracts/token.sol#684-691)
                - (success) = recipient.call{value: amount}() (contracts/token.sol#356)
                - router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/token.sol#670-676)
                - address(marketingWallet).sendValue(marketingAmt) (contracts/token.sol#656)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (contracts/token.sol#515)
                - router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,deadWallet,block.timestamp) (contracts/token.sol#684-691)
                - (success) = recipient.call{value: amount}() (contracts/token.sol#356)
        State variables written after the call(s):
        - _approve(sender,_msgSender(),currentAllowance - amount) (contracts/token.sol#522)
                - _allowances[owner][spender] = amount (contracts/token.sol#344)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in Cyberverseland.Liquify(uint256,Cyberverseland.Taxes) (contracts/token.sol#620-659):
        External calls:
        - swapTokensForETH(toSwap) (contracts/token.sol#642)
                - router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/token.sol#670-676)
        - addLiquidity(tokensToAddLiquidityWith,ethToAddLiquidityWith) (contracts/token.sol#651)
                - router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,deadWallet,block.timestamp) (contracts/token.sol#684-691)
        External calls sending eth:
        - addLiquidity(tokensToAddLiquidityWith,ethToAddLiquidityWith) (contracts/token.sol#651)
                - router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,deadWallet,block.timestamp) (contracts/token.sol#684-691)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (contracts/token.sol#345)
                - addLiquidity(tokensToAddLiquidityWith,ethToAddLiquidityWith) (contracts/token.sol#651)
```



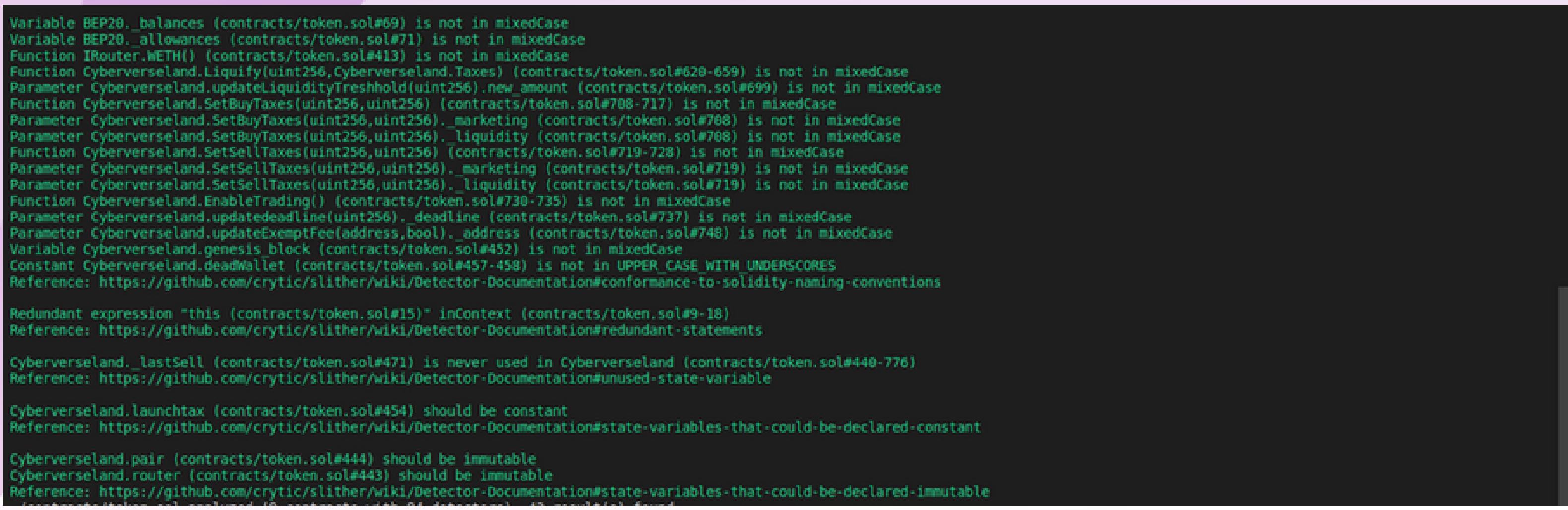
```
Reentrancy in Cyberverseland._transfer(address,address,uint256) (contracts/token.sol#564-618):
        External calls:
        - Liquify(feeswap,currentTaxes) (contracts/token.sol#607)
                - router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,deadWallet,block.timestamp) (contracts/token.sol#684-691)
                - (success) = recipient.call{value: amount}() (contracts/token.sol#356)
                - router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/token.sol#670-676)
                - address(marketingWallet).sendValue(marketingAmt) (contracts/token.sol#656)
        External calls sending eth:
        - Liquify(feeswap,currentTaxes) (contracts/token.sol#607)
                - router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,deadWallet,block.timestamp) (contracts/token.sol#684-691)
                - (success) = recipient.call{value: amount}() (contracts/token.sol#356)
        Event emitted after the call(s):
        - Transfer(sender,recipient,amount) (contracts/token.sol#306)
                - super._transfer(sender,address(this),feeAmount) (contracts/token.sol#615)
        - Transfer(sender,recipient,amount) (contracts/token.sol#306)
                - super._transfer(sender,recipient,amount - fee) (contracts/token.sol#610)
Reentrancy in Cyberverseland.transferFrom(address,address,uint256) (contracts/token.sol#510-525):
        External calls:
        - _transfer(sender,recipient,amount) (contracts/token.sol#515)
                - router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,deadWallet,block.timestamp) (contracts/token.sol#684-691)
                - (success) = recipient.call{value: amount}() (contracts/token.sol#356)
                - router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/token.sol#670-676)
                - address(marketingWallet).sendValue(marketingAmt) (contracts/token.sol#656)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (contracts/token.sol#515)
                - router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,deadWallet,block.timestamp) (contracts/token.sol#684-691)
                - (success) = recipient.call{value: amount}() (contracts/token.sol#356)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (contracts/token.sol#345)
                - _approve(sender,_msgSender(),currentAllowance - amount) (contracts/token.sol#522)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Context._msgData() (contracts/token.sol#14-17) 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#7) 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

Low level call in Address.sendValue(address,uint256) (contracts/token.sol#350-361):
        - (success) = recipient.call{value: amount}() (contracts/token.sol#356)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

STATIC ANALYSIS



```
Variable BEP20._balances (contracts/token.sol#69) is not in mixedCase
Variable BEP20._allowances (contracts/token.sol#71) is not in mixedCase
Function IRouter.WETH() (contracts/token.sol#413) is not in mixedCase
Function Cyberverseland.Liquify(uint256,Cyberverseland.Taxes) (contracts/token.sol#620-659) is not in mixedCase
Parameter Cyberverseland.updateLiquidityTreshhold(uint256).new_amount (contracts/token.sol#699) is not in mixedCase
Function Cyberverseland.SetBuyTaxes(uint256,uint256) (contracts/token.sol#708-717) is not in mixedCase
Parameter Cyberverseland.SetBuyTaxes(uint256,uint256)._marketing (contracts/token.sol#708) is not in mixedCase
Parameter Cyberverseland.SetBuyTaxes(uint256,uint256)._liquidity (contracts/token.sol#708) is not in mixedCase
Function Cyberverseland.SetSellTaxes(uint256,uint256) (contracts/token.sol#719-728) is not in mixedCase
Parameter Cyberverseland.SetSellTaxes(uint256,uint256)._marketing (contracts/token.sol#719) is not in mixedCase
Parameter Cyberverseland.SetSellTaxes(uint256,uint256)._liquidity (contracts/token.sol#719) is not in mixedCase
Function Cyberverseland.EnableTrading() (contracts/token.sol#730-735) is not in mixedCase
Parameter Cyberverseland.updatedeadline(uint256)._deadline (contracts/token.sol#737) is not in mixedCase
Parameter Cyberverseland.updateExemptFee(address,bool)._address (contracts/token.sol#748) is not in mixedCase
Variable Cyberverseland.genesis_block (contracts/token.sol#452) is not in mixedCase
Constant Cyberverseland.deadWallet (contracts/token.sol#457-458) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (contracts/token.sol#15)" inContext (contracts/token.sol#9-18)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

Cyberverseland._lastSell (contracts/token.sol#471) is never used in Cyberverseland (contracts/token.sol#440-776)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

Cyberverseland.launchtax (contracts/token.sol#454) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

Cyberverseland.pair (contracts/token.sol#444) should be immutable
Cyberverseland.router (contracts/token.sol#443) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
```

Result => No issues found

FUNCTIONAL TESTING

Functionality tests for ERC20 tokens includes:
- adding liquidity
- buying / selling /transferring (for non-excluded wallets)
- checking tax conversions, tax destinations
- checking auto liquidity

1- Adding Liquidity:

liquidity added on Uniswap v2:

https://goerli.etherscan.io/tx/0xfe45db2e8c97a295d0a1d37a361b6d394e63e58b9760662cf4ff06534b77e73f

no issue were found on adding liquidity.

2- Buying from a non-excluded wallet:

https://goerli.etherscan.io/tx/0xc2fe1e0e2c44a20d04e3fc121eeecee130dcef1dfc1cebac3f1af297584e6037

1% tax on buy, transferred to contract (not reached swap threshold yet)

3- Selling from a non-excluded wallet

https://goerli.etherscan.io/tx/0xafdace96a96b2f66e7e0f62a9365f2db74976b53cf3f6eb369d2409e023ee667

FUNCTIONAL TESTING

1% tax on sell, transferred to contract (not reached swap threshold yet)

4- Swap & liquifiy

since liquidity tax is 0 and taxes can not be changed later, then auto-liquidity is disabled forever. But to check marketing tax, we transferred 10M tokens to the contract to reach swap threshold and then we performed a sell:

https://goerli.etherscan.io/tx/0xafdace96a96b2f66e7e0f62a9365f2db74976b53cf3f6eb369d2409e023ee667

marketing wallet received converted ETH tokens received from swapping taxes.

MANUAL TESTING

NO RISKS WERE FOUND IN THE CONTRACT

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