auditace.tech

AuditAce

FROM INCEPTION TO SUCCESS

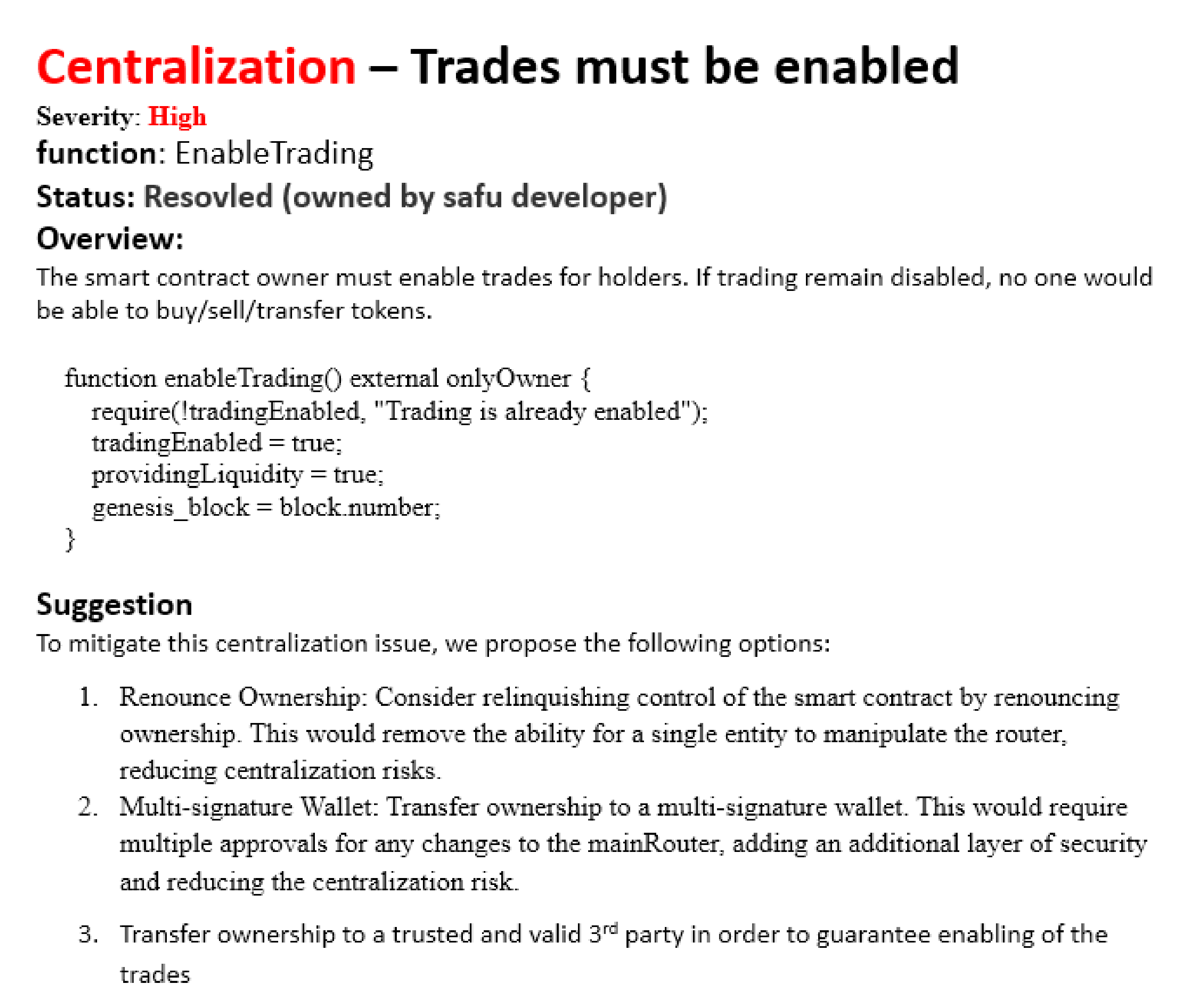
Smart Contract Audit

FOR

VPW

DATED : 9 June 23'

HIGH RISK FINDING



Centralization – Trades must be enabled

Severity: **High**

function: EnableTrading

Status: Resovled (owned by safu developer)

Overview:

The smart contract owner must enable trades for holders. If trading remain disabled, no one would be able to buy/sell/transfer tokens.

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

Suggestion

To mitigate this centralization issue, we propose the following options:

1. Renounce Ownership: Consider relinquishing control of the smart contract by renouncing ownership. This would remove the ability for a single entity to manipulate the router, reducing centralization risks.
2. Multi-signature Wallet: Transfer ownership to a multi-signature wallet. This would require multiple approvals for any changes to the mainRouter, adding an additional layer of security and reducing the centralization risk.
3. Transfer ownership to a trusted and valid 3rd party in order to guarantee enabling of the trades

AUDIT SUMMARY

Project name – VPW

Date: 9 June, 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:
Contract has been tested on binance smart chain testnet which can be found in below link:
https://testnet.bscscan.com/token/0x812d58fa4be6bcb3a3e08cb0c4f0bfb9c4b3b51f#code

Token Information

Token Name : Virtual Peradox World

Token Symbol: VPW

Decimals: 18

Token Supply: 599,000,000

Token Address:
0x7BB224B336ECa8f5FfBf45bD460eFc61f018F962

Checksum:
5970ffc8793efd442c45649650ac16c09e7ca8c4

Owner:
0x054f5759416785897Dbff570a2Dc3e18d4F38Fa6

Deployer:
0x054f5759416785897Dbff570a2Dc3e18d4F38Fa6

TOKEN OVERVIEW

Fees:

Buy Fees: 1%

Sell Fees: 1%

Transfer Fees: 1%

Fees Privilege: None

Ownership: Owned

Minting: None

Max Tx Amount/ Max Wallet Amount: Yes

Blacklist: No

Other Privileges: - initial distribution of tokens

- including or excluding from fees

- changing swap threshold

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

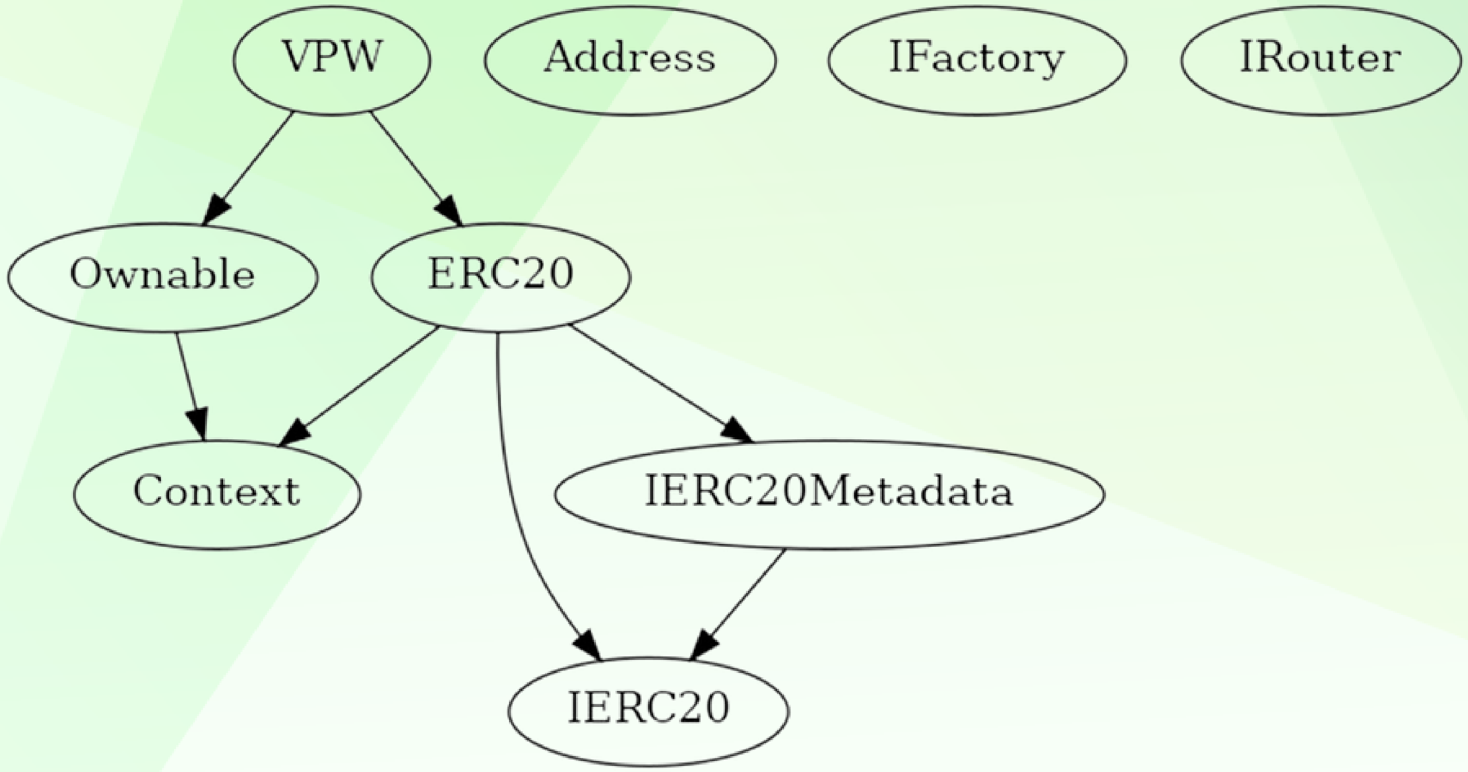
CLASSIFICATION OF RISK

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	1
◆ Medium-Risk	0
◆ Low-Risk	0
◆ Gas Optimization / Suggestions	0

INHERITANCE TREE



POINTS TO NOTE

- owner is not able to set change buy/sell/transfer fees (1% static)
- owner is not able to blacklist an arbitrary wallet
- owner is not able to set limit for buy/sell/transfer/holding amounts
- owner is not able to mint new tokens
- owner is not able to disable trades
- owner can exclude/include an address from fees
- owner can change internal swap thershold
- owner can enable/disable internal swap (I.e marketing and development BNB not trades)
- owner can claim stuck tokens
- owner can transfer ownership
- owner can renounce ownership

CONTRACT ASSESMENT

```
|  Contract  |        Type        |       Bases      |                  |                 |
|:----------:|:-------------------:|:----------------:|:----------------:|:---------------:|
|     └      |  **Function Name**  |  **Visibility**  |  **Mutability**  |  **Modifiers**  |
||||||
| **Context** | Implementation |  |||
| └ | _msgSender | Internal 🔒 |   | |
| └ | _msgData | Internal 🔒 |   | |
||||||
| **IERC20** | Interface |  |||
| └ | totalSupply | External ❗️ |   |NO❗️ |
| └ | balanceOf | External ❗️ |   |NO❗️ |
| └ | transfer | External ❗️ | 🛑  |NO❗️ |
| └ | allowance | External ❗️ |   |NO❗️ |
| └ | approve | External ❗️ | 🛑  |NO❗️ |
| └ | transferFrom | External ❗️ | 🛑  |NO❗️ |
||||||
| **IERC20Metadata** | Interface | IERC20 |||
| └ | name | External ❗️ |   |NO❗️ |
| └ | symbol | External ❗️ |   |NO❗️ |
| └ | decimals | 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❗️ |
| └ | transfer | Public ❗️ | 🛑  |NO❗️ |
| └ | allowance | Public ❗️ |   |NO❗️ |
| └ | approve | Public ❗️ | 🛑  |NO❗️ |
| └ | transferFrom | Public ❗️ | 🛑  |NO❗️ |
| └ | increaseAllowance | Public ❗️ | 🛑  |NO❗️ |
| └ | decreaseAllowance | Public ❗️ | 🛑  |NO❗️ |
| └ | _transfer | Internal 🔒 | 🛑  | |
| └ | _tokengeneration | Internal 🔒 | 🛑  | |
| └ | _burn | Internal 🔒 | 🛑  | |
| └ | _approve | Internal 🔒 | 🛑  | |
| └ | _beforeTokenTransfer | Internal 🔒 | 🛑  | |
||||||
| **Address** | Library |  |||
| └ | sendValue | Internal 🔒 | 🛑  | |
```

CONTRACT ASSESMENT

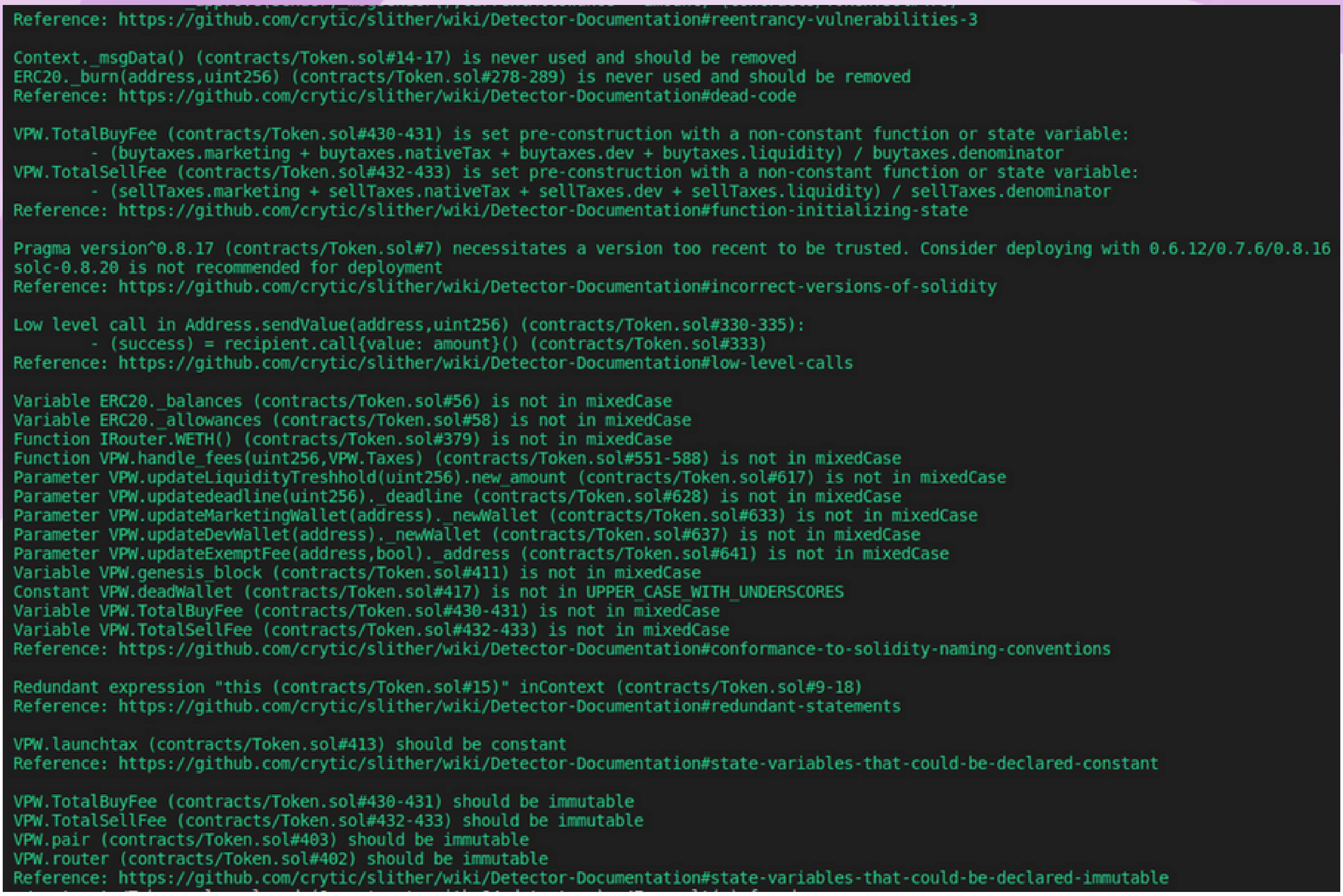
```
||||||
| **Ownable** | Implementation | Context |||
| └ | <Constructor> | Public ❗️ | 🛑 |NO ❗️ |
| └ | owner | Public ❗️ |   |NO ❗️ |
| └ | renounceOwnership | Public ❗️ | 🛑 | onlyOwner |
| └ | transferOwnership | Public ❗️ | 🛑 | onlyOwner |
| └ | _setOwner | Private 🔐 | 🛑 | |
||||||
| **IFactory** | Interface |  |||
| └ | createPair | External ❗️ | 🛑 |NO ❗️ |
||||||
| **IRouter** | Interface |  |||
| └ | factory | External ❗️ |   |NO ❗️ |
| └ | WETH | External ❗️ |   |NO ❗️ |
| └ | addLiquidityETH | External ❗️ | 💵 |NO ❗️ |
| └ | swapExactTokensForETHSupportingFeeOnTransferTokens | External ❗️ | 🛑 |NO ❗️ |
||||||
| **VPW** | Implementation | ERC20, Ownable |||
| └ | <Constructor> | Public ❗️ | 🛑 | ERC20 |
| └ | approve | Public ❗️ | 🛑 |NO ❗️ |
| └ | transferFrom | Public ❗️ | 🛑 |NO ❗️ |
| └ | increaseAllowance | Public ❗️ | 🛑 |NO ❗️ |
| └ | decreaseAllowance | Public ❗️ | 🛑 |NO ❗️ |
| └ | transfer | Public ❗️ | 🛑 |NO ❗️ |
| └ | _transfer | Internal 🔒 | 🛑 | |
| └ | handle_fees | Private 🔐 | 🛑 | mutexLock |
| └ | swapTokensForETH | Private 🔐 | 🛑 | |
| └ | addLiquidity | Private 🔐 | 🛑 | |
| └ | updateLiquidityProvide | External ❗️ | 🛑 | onlyOwner |
| └ | updateLiquidityTreshhold | External ❗️ | 🛑 | onlyOwner |
| └ | enableTrading | External ❗️ | 🛑 | onlyOwner |
| └ | updatedeadline | External ❗️ | 🛑 | onlyOwner |
| └ | updateMarketingWallet | External ❗️ | 🛑 | onlyOwner |
| └ | updateDevWallet | External ❗️ | 🛑 | onlyOwner |
| └ | updateExemptFee | External ❗️ | 🛑 | onlyOwner |
| └ | bulkExemptFee | External ❗️ | 🛑 | onlyOwner |
| └ | rescueBNB | External ❗️ | 🛑 |NO ❗️ |
| └ | rescueBEP20 | External ❗️ | 🛑 |NO ❗️ |
| └ | <Receive Ether> | External ❗️ | 💵 |NO ❗️ |
```

CONTRACT ASSESMENT

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

STATIC ANALYSIS



```
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
ERC20._burn(address,uint256) (contracts/Token.sol#278-289) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

VPW.TotalBuyFee (contracts/Token.sol#430-431) is set pre-construction with a non-constant function or state variable:
        - (buytaxes.marketing + buytaxes.nativeTax + buytaxes.dev + buytaxes.liquidity) / buytaxes.denominator
VPW.TotalSellFee (contracts/Token.sol#432-433) is set pre-construction with a non-constant function or state variable:
        - (sellTaxes.marketing + sellTaxes.nativeTax + sellTaxes.dev + sellTaxes.liquidity) / sellTaxes.denominator
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-state

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-0.8.20 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#330-335):
        - (success) = recipient.call{value: amount}() (contracts/Token.sol#333)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Variable ERC20._balances (contracts/Token.sol#56) is not in mixedCase
Variable ERC20._allowances (contracts/Token.sol#58) is not in mixedCase
Function IRouter.WETH() (contracts/Token.sol#379) is not in mixedCase
Function VPW.handle_fees(uint256,VPW.Taxes) (contracts/Token.sol#551-588) is not in mixedCase
Parameter VPW.updateLiquidityTreshhold(uint256).new_amount (contracts/Token.sol#617) is not in mixedCase
Parameter VPW.updatedeadline(uint256)._deadline (contracts/Token.sol#628) is not in mixedCase
Parameter VPW.updateMarketingWallet(address)._newWallet (contracts/Token.sol#633) is not in mixedCase
Parameter VPW.updateDevWallet(address)._newWallet (contracts/Token.sol#637) is not in mixedCase
Parameter VPW.updateExemptFee(address,bool)._address (contracts/Token.sol#641) is not in mixedCase
Variable VPW.genesis_block (contracts/Token.sol#411) is not in mixedCase
Constant VPW.deadWallet (contracts/Token.sol#417) is not in UPPER_CASE_WITH_UNDERSCORES
Variable VPW.TotalBuyFee (contracts/Token.sol#430-431) is not in mixedCase
Variable VPW.TotalSellFee (contracts/Token.sol#432-433) is not in mixedCase
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

VPW.launchtax (contracts/Token.sol#413) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

VPW.TotalBuyFee (contracts/Token.sol#430-431) should be immutable
VPW.TotalSellFee (contracts/Token.sol#432-433) should be immutable
VPW.pair (contracts/Token.sol#403) should be immutable
VPW.router (contracts/Token.sol#402) 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,

No major issues were found in the output

FUNCTIONAL TESTING

1- Adding liquidity (passed):

https://testnet.bscscan.com/tx/0x394541469f999a97fb96c473ba491f816505d9318bc7746d127b7f09fe719d58

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

https://testnet.bscscan.com/tx/0x20f30746694fa8400dd6363107595299380905b28283ac6935aaf24717414270

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

https://testnet.bscscan.com/tx/0xd1dd73342317657ef18704ea748ec673e316b0c13c38f90473cb83bfcbeaf5aa

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

https://testnet.bscscan.com/tx/0x85bd3ece6124eaf14a3744c53f8830dee8bf17c2c8f92f50fa1d41b9e64fe9ee

5- Buying when not excluded from fees (1% tax) (passed):

https://testnet.bscscan.com/tx/0x28d750cde7ed8c2f4925512c3173a15e73594a3c143ff6fd5ed763bbab2de499

6- Selling when not excluded from fees (1% tax) (passed):

https://testnet.bscscan.com/tx/0xa47e15f16212f5dedca2535f26ce533cd356fa80e745b17a60887e5f7a73f686

FUNCTIONAL TESTING

7- Transferring when not excluded from fees (1% tax) (passed):

https://testnet.bscscan.com/tx/0x805ab4627f41a31d024d1e275d218b3abe40cdabd0b80e2a97dcc85a912fc334

8- Development and marketing BNB (Internal swap) (passed):

https://testnet.bscscan.com/tx/0x7374161405ddee4de1a357efd392f982418e03161cee3f3a9edf9690ab236ea9

FUNCTIONAL TESTING

Centralization – Trades must be enabled

Severity: **High**

function: EnableTrading

Status: Resovled (owned by safu developer)

Overview:

The smart contract owner must enable trades for holders. If trading remain disabled, no one would be able to buy/sell/transfer tokens.

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

Suggestion

To mitigate this centralization issue, we propose the following options:

1. Renounce Ownership: Consider relinquishing control of the smart contract by renouncing ownership. This would remove the ability for a single entity to manipulate the router, reducing centralization risks.
2. Multi-signature Wallet: Transfer ownership to a multi-signature wallet. This would require multiple approvals for any changes to the mainRouter, adding an additional layer of security and reducing the centralization risk.
3. Transfer ownership to a trusted and valid 3rd party in order to guarantee enabling of the trades

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