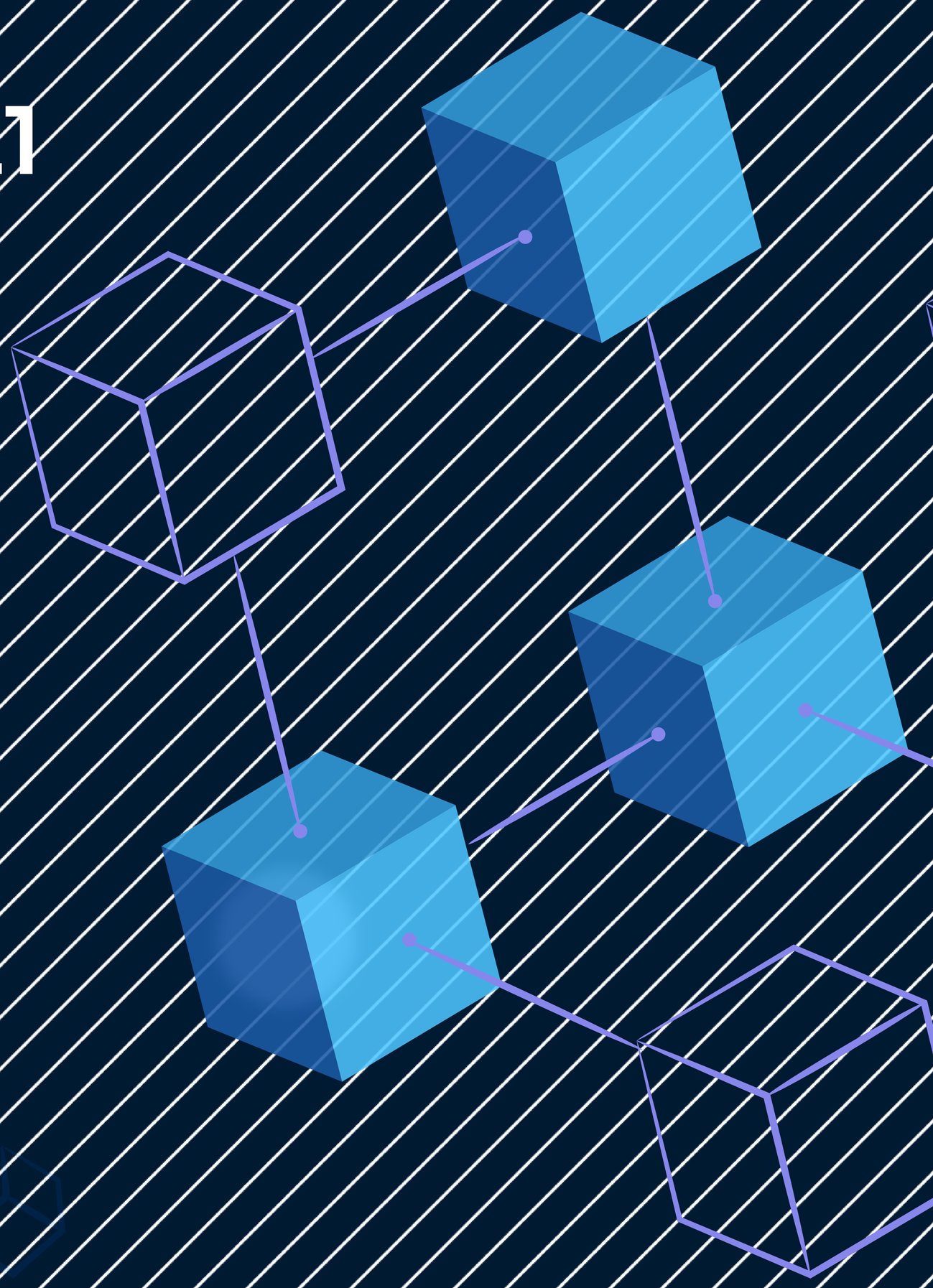
QuillAudits

Audit Report
November, 2021



For

FUNDUM

Contents

Scope of the Audit

The scope of this Audit was to analyze and document the Fundum smart contracts codebase for quality, security, and correctness.

Checked Vulnerabilities

We have scanned the smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that we considered:

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- Unsafe type inference
- Implicit visibility level

Techniques and Methods

Throughout the audit of smart contract, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of BEP-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

Structural Analysis
In this step, we have analysed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis
Static analysis of smart contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis
Manual analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analysed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption
In this step, we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit
Remix IDE, Truffle, Truffle Team, Solhint, Mythril, Slither, Solidity statistic analysis, Theo.

Issue Categories

Every issue in this report has been assigned to a severity level. There are four levels of severity, and each of them has been explained below.

Risk-level	Description
High	A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.
Medium	The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.
Low	Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.
Informational	These are severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Number of issues per severity

Type	High	Medium	Low	Informational
Open	0	0	0	0
Acknowledged	0	0	2	0
Closed	0	0	2	0

Introduction

During the period of **November 01, 2021, to November 14, 2021** - QuillAudits Team performed a security audit for **Fundum** smart contracts.

The code for the Audit was taken from the following : https://bscscan.com/address/0x19434Bae46D74854318fFec043775De2F6f961d4#code

Note	Date	Commit hash
Version 1	November	https://testnet.bscscan.com/address/0x9a05c012528c61d7e788369b5b5324adff0b2fe4#code
Version 2	November	https://bscscan.com/address/0x19434Bae46D74854318fFec043775De2F6f961d4#code

Issues Found

A. Contract – Fundum

High severity issues

No issues were found.

Medium severity issues

No issues were found.

Low level severity issues

1. **Approve Race**

```
Line 74:
    function _approve(address owner, address spender, uint256 amount)
      internal {
        require(owner != address(0), "BEP20: approve from the zero address");
        require(spender != address(0), "BEP20: approve to the zero address");

        _allowances[owner][spender] = amount;
        emit Approval(owner, spender, amount);
    }
```

Description

The standard ERC20 implementation contains a widely-known racing condition in its approve function, wherein a spender is able to witness the token owner broadcast a transaction altering their approval and quickly sign and broadcast a transaction using transferFrom to move the current approved amount from the owner's balance to the spender. If the spender's transaction is validated before the owner's, the spender will be able to get both approval amounts of both transactions.

Remediation

Use increaseAllowance and decreaseAllowance functions to modify the approval amount instead of using the approve function to modify it.

Acknowledged
The Fundum team has acknowledged the risk. The Team used the approve function to be able to approve the auto swap to BUSD by the router; this also includes the increaseAllowance function. But the increaseAllowance function would only be called by the user on a case by case basis as the user sees fit. Also, for future use in staking and other future options, this function will/could be used.

Status: Acknowledged

2. Usage of block.timestamp

```
Line 66:
        function addLiquidity(uint256 tokenAmount, uint256 ethAmount) private {
         // approve token transfer to cover all possible scenarios
         _approve(address(this), address(uniswapV2Router), tokenAmount);
         // add the liquidity
         uniswapV2Router.addLiquidityETH{value: ethAmount}(
            address(this),
            tokenAmount,
            0, // slippage is unavoidable
            0, // slippage is unavoidable
            owner(),
            block.timestamp
        );
        }
    function swapTokensForETH(uint256 tokenAmount) private lockTheSwap {
        // generate the uniswap pair path of token -> weth
        address[] memory path = new address[](2);
        path[0] = address(this);
        path[1] = uniswapV2Router.WETH();
        _approve(address(this), address(uniswapV2Router), tokenAmount);
        // make the swap
        uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
            tokenAmount,
            0, // accept any amount of ETH
            path,
            address(this),
            block.timestamp
        );
        }
    function swapTokensForBUSD(address toAddress, uint256 tokenAmount) private {
        // generate the uniswap pair path of token -> token
        address[] memory path = new address[](3);
        path[0] = address(this);
        path[1] = uniswapV2Router.WETH();
        path[2] = busdAddress;
        _approve(address(this), address(uniswapV2Router), tokenAmount);
```

```
    // make the swap
    uniswapV2Router.swapExactTokensForTokensSupportingFeeOnTransferTokens(
        tokenAmount,
        0, // accept any amount of Tokens
        path,
        toAddress, // The contract
        block.timestamp.add(120)
    );
    emit SwapTokensForBUSD(tokenAmount, path);
}
```

Description

Block.timestamp is used in the contract. The variable block is a set of variables. The timestamp does not always reflect the current time and may be inaccurate. The value of a block can be influenced by miners. Maximal Extractable Value attacks require a timestamp of up to 900 seconds. There is no guarantee that the value is right; all that is guaranteed is that it is higher than the timestamp of the previous block.

Acknowledged

The Fundum team doesn't use the block.timestamp for anything that is date/time sensitive.

Status: Closed

3. Floating pragma

```
Line 3:
pragma solidity ^0.8.4;
```

Description

The contract makes use of the floating-point pragma 0.8.4 Contracts should be deployed using the same compiler version and flags that were used during the testing process. Locking the pragma helps ensure that contracts are not unintentionally deployed using another pragma, such as an obsolete version that may introduce issues in the contract system.

Remediation

Consider locking the pragma version. It is advised that floating pragma not be used in production. Both truffle-config.js and hardhat.config.js support locking the pragma version.

Fixed
The Fundum team has fixed the issue in version 2 by locking the pragma version.

Status: Closed

4. Renounce Ownership

```
Line 12:
contract Fundum is Context, IBEP20, Ownable {
```

Description
Typically, the contract's Owner is the account that deploys the contract. As a result, the Owner can perform certain privileged activities on his behalf. The renounceOwnership function is used in smart contracts to renounce ownership. Otherwise, if the contract's ownership has not been transferred previously, it will never have an Owner, which is risky.

Remediation
It is advised that the Owner cannot call renounceOwnership without first transferring ownership to a different address. Additionally, if a multi-signature wallet is utilized, executing the renounceOwnership method for two or more users should be confirmed. Alternatively, the Renounce Ownership functionality can be disabled by overriding it.

Acknowledged
The Fundum team has acknowledged the risk. The contract is owned by a registered company, and it will be placed in a multisig environment, which is currently being developed. As soon as the multisig environment is ready, the contract will be added to it to make renouncing redundant.

Status: Acknowledged

Test Cases for Functional Testing

Some of the test cases which were part of the functional testing are listed below:

Fundum.sol

1. User cannot call the transfer function if the trading is not open

PASS

2. Accounts Who are Excluded from Fee Should not pay any Amount

PASS

3. Normal Users Should Pay Fees using the transfer function

PASS

BSC Test Contracts

Fundum: 0x9a05c012528c61d7e788369b5b5324adff0b2fe4

Automated Tests

Slither

```
Address.isContract(address) (Address.sol#6-17) uses assembly
        - INLINE ASM (Address.sol#13-15)
Address._functionCallWithValue(address,bytes,uint256,string) (Address.sol#75-98) uses assembly
        - INLINE ASM (Address.sol#90-93)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Address._functionCallWithValue(address,bytes,uint256,string) (Address.sol#75-98) is never used and should be removed
Address.functionCall(address,bytes) (Address.sol#33-38) is never used and should be removed
Address.functionCall(address,bytes,string) (Address.sol#40-46) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (Address.sol#48-60) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (Address.sol#62-73) is never used and should be removed
Address.isContract(address) (Address.sol#6-17) is never used and should be removed
Address.sendValue(address,uint256) (Address.sol#19-31) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.4 (Address.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 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) (Address.sol#19-31):
        - (success) = recipient.call{value: amount}() (Address.sol#26)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (Address.sol#75-98):
        - (success,returndata) = target.call{value: weiValue}(data) (Address.sol#83-85)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

SafeMath.add(uint256,uint256) (SafeMath.sol#6-11) is never used and should be removed
SafeMath.div(uint256,uint256) (SafeMath.sol#39-41) is never used and should be removed
SafeMath.div(uint256,uint256,string) (SafeMath.sol#43-53) is never used and should be removed
SafeMath.mod(uint256,uint256) (SafeMath.sol#55-57) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (SafeMath.sol#59-66) is never used and should be removed
SafeMath.mul(uint256,uint256) (SafeMath.sol#28-37) is never used and should be removed
SafeMath.sub(uint256,uint256) (SafeMath.sol#13-15) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (SafeMath.sol#17-26) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.4 (SafeMath.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Context._msgData() (Context.sol#10-13) is never used and should be removed
Context._msgSender() (Context.sol#6-8) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.4 (Context.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

```
Pragma version^0.8.4 (Context.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Redundant expression "this (Context.sol#11)" inContext (Context.sol#5-14)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

Ownable.unlock() (Ownable.sol#61-69) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp > _lockTime,Contract is locked until 7 days) (Ownable.sol#66)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Context._msgData() (Context.sol#10-13) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.4 (Context.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.4 (Ownable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Redundant expression "this (Context.sol#11)" inContext (Context.sol#5-14)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

owner() should be declared external:
        - Ownable.owner() (Ownable.sol#23-25)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (Ownable.sol#32-35)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (Ownable.sol#37-44)
getUnlockTime() should be declared external:
        - Ownable.getUnlockTime() (Ownable.sol#46-48)
getTime() should be declared external:
        - Ownable.getTime() (Ownable.sol#50-52)
lock(uint256) should be declared external:
        - Ownable.lock(uint256) (Ownable.sol#54-59)
unlock() should be declared external:
        - Ownable.unlock() (Ownable.sol#61-69)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external

Pragma version^0.8.4 (IBEP20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

```
Fundum.addLiquidity(uint256,uint256) (Fundum.sol#406-419) ignores return value by uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Fundum.sol#411-418)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

Fundum.allowance(address,address).owner (Fundum.sol#175) shadows:
        - Ownable.owner() (Ownable.sol#23-25) (function)
Fundum._approve(address,address,uint256).owner (Fundum.sol#372) shadows:
        - Ownable.owner() (Ownable.sol#23-25) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

Fundum.updateMinSwapBalance(uint256) (Fundum.sol#482-485) should emit an event for:
        - minSwapBalance = _minBalance * 10 ** 9 (Fundum.sol#484)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

Reentrancy in Fundum._distributeTaxFees(uint256) (Fundum.sol#460-480):
        External calls:
        - swapAndLiquify(taxBalance.mul(5).div(10)) (Fundum.sol#461)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Fundum.sol#411-418)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Fundum.sol#430-436)
        - swapTokensForBUSD(teamWallet,taxBalance.mul(1).div(10)) (Fundum.sol#463)
                - uniswapV2Router.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount,0,path,toAddress,block.timestamp.add(120)) (Fundum.sol#449-455)
        External calls sending eth:
        - swapAndLiquify(taxBalance.mul(5).div(10)) (Fundum.sol#461)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Fundum.sol#411-418)
        State variables written after the call(s):
        - swapTokensForBUSD(teamWallet,taxBalance.mul(1).div(10)) (Fundum.sol#463)
                - _allowances[owner][spender] = amount (Fundum.sol#379)
Reentrancy in Fundum._distributeTaxFees(uint256) (Fundum.sol#460-480):
        External calls:
        - swapAndLiquify(taxBalance.mul(5).div(10)) (Fundum.sol#461)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Fundum.sol#411-418)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Fundum.sol#430-436)
        - swapTokensForBUSD(teamWallet,taxBalance.mul(1).div(10)) (Fundum.sol#463)
                - uniswapV2Router.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount,0,path,toAddress,block.timestamp.add(120)) (Fundum.sol#449-455)
        - swapTokensForBUSD(marketingWallet,taxBalance.mul(2).div(10)) (Fundum.sol#465)
                - uniswapV2Router.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount,0,path,toAddress,block.timestamp.add(120)) (Fundum.sol#449-455)
        External calls sending eth:
        - swapAndLiquify(taxBalance.mul(5).div(10)) (Fundum.sol#461)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Fundum.sol#411-418)
        State variables written after the call(s):
        - swapTokensForBUSD(marketingWallet,taxBalance.mul(2).div(10)) (Fundum.sol#465)
                - _allowances[owner][spender] = amount (Fundum.sol#379)
Reentrancy in Fundum._distributeTaxFees(uint256) (Fundum.sol#460-480):
        External calls:
        - swapAndLiquify(taxBalance.mul(5).div(10)) (Fundum.sol#461)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Fundum.sol#411-418)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Fundum.sol#430-436)
        - swapTokensForBUSD(teamWallet,taxBalance.mul(1).div(10)) (Fundum.sol#463)
                - uniswapV2Router.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount,0,path,toAddress,block.timestamp.add(120)) (Fundum.sol#449-455)
        - swapTokensForBUSD(marketingWallet,taxBalance.mul(2).div(10)) (Fundum.sol#465)
                - uniswapV2Router.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount,0,path,toAddress,block.timestamp.add(120)) (Fundum.sol#449-455)
```

```
Reentrancy in Fundum._distributeTaxFees(uint256) (Fundum.sol#460-480):
        External calls:
        - swapAndLiquify(taxBalance.mul(5).div(10)) (Fundum.sol#461)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Fundum.sol#411-418)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Fundum.sol#430-436)
        - swapTokensForBUSD(teamWallet,taxBalance.mul(1).div(10)) (Fundum.sol#463)
                - uniswapV2Router.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount,0,path,toAddress,block.timestamp.add(120)) (Fundum.sol#449-455)
        - swapTokensForBUSD(marketingWallet,taxBalance.mul(2).div(10)) (Fundum.sol#465)
                - uniswapV2Router.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount,0,path,toAddress,block.timestamp.add(120)) (Fundum.sol#449-455)
        - swapTokensForBUSD(buybackWallet,taxBalance.mul(2).div(10)) (Fundum.sol#467)
                - uniswapV2Router.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount,0,path,toAddress,block.timestamp.add(120)) (Fundum.sol#449-455)
        External calls sending eth:
        - swapAndLiquify(taxBalance.mul(5).div(10)) (Fundum.sol#461)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Fundum.sol#411-418)
        State variables written after the call(s):
        - swapTokensForBUSD(buybackWallet,taxBalance.mul(2).div(10)) (Fundum.sol#467)
                - _allowances[owner][spender] = amount (Fundum.sol#379)
        - _balances[address(this)] = _balances[address(this)].sub(lessAmount) (Fundum.sol#476)
        - _balances[uniswapV2Pair] = _balances[uniswapV2Pair].add(lessAmount) (Fundum.sol#477)
        - tradeTokens = tradeTokens.sub(lessAmount) (Fundum.sol#475)
Reentrancy in Fundum._transfer(address,address,uint256) (Fundum.sol#297-356):
        External calls:
        - _distributeTaxFees(remainTax) (Fundum.sol#331)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Fundum.sol#411-418)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Fundum.sol#430-436)
                - uniswapV2Router.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount,0,path,toAddress,block.timestamp.add(120)) (Fundum.sol#449-455)
        External calls sending eth:
        - _distributeTaxFees(remainTax) (Fundum.sol#331)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Fundum.sol#411-418)
        State variables written after the call(s):
        - totalTax = totalTax.add(fee) (Fundum.sol#338)
Reentrancy in Fundum.constructor() (Fundum.sol#72-106):
        External calls:
        - uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),busdAddress) (Fundum.sol#82-83)
        State variables written after the call(s):
        - _balances[deployerWallet] = lp1Tokens (Fundum.sol#94)
        - _balances[lp2Wallet] = lp2Tokens (Fundum.sol#96)
        - _balances[release2Wallet] = reserveTokens (Fundum.sol#98)
        - _balances[address(this)] = tradeTokens (Fundum.sol#100)
        - _isExcludedFromFee[deployerWallet] = true (Fundum.sol#86)
        - _isExcludedFromFee[release2Wallet] = true (Fundum.sol#87)
        - _isExcludedFromFee[teamWallet] = true (Fundum.sol#88)
        - _isExcludedFromFee[marketingWallet] = true (Fundum.sol#89)
        - _isExcludedFromFee[buybackWallet] = true (Fundum.sol#90)
        - _isExcludedFromFee[lp2Wallet] = true (Fundum.sol#91)
        - _isExcludedFromFee[address(this)] = true (Fundum.sol#92)
        - uniswapV2Router = _uniswapV2Router (Fundum.sol#84)
```

```
Pragma version^0.8.4 (Address.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.4 (Context.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.4 (Fundum.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.4 (IBEP20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.4 (Ownable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.4 (SafeMath.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.4 (Uniswap.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 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) (Address.sol#19-31):
        - (success) = recipient.call{value: amount}() (Address.sol#26)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (Address.sol#75-98):
        - (success,returndata) = target.call{value: weiValue}(data) (Address.sol#83-85)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter Fundum.updateMinSwapBalance(uint256)._minBalance (Fundum.sol#482) is not in mixedCase
Parameter Fundum.setSwapAndLiquifyEnabled(bool)._enabled (Fundum.sol#487) is not in mixedCase
Parameter Fundum.setOpenTrading(bool)._enabled (Fundum.sol#492) is not in mixedCase
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (Uniswap.sol#67) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (Uniswap.sol#69) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (Uniswap.sol#99) is not in mixedCase
Function IUniswapV2Router01.WETH() (Uniswap.sol#143) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (Context.sol#11)" inContext (Context.sol#5-14)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (Uniswap.sol#148) is too similar to IUniswapV2Router01.addLiquidity(address,address,uint25
6,uint256,uint256,uint256,address,uint256).amountBDesired (Uniswap.sol#149)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

Fundum.slitherConstructorVariables() (Fundum.sol#12-511) uses literals with too many digits:
        - deadAddress = 0x000000000000000000000000000000000000dEaD (Fundum.sol#45-46)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

Fundum._decimals (Fundum.sol#24) should be constant
Fundum._name (Fundum.sol#22) should be constant
Fundum._symbol (Fundum.sol#23) should be constant
Fundum._totalSupply (Fundum.sol#20) should be constant
Fundum.busdAddress (Fundum.sol#44) should be constant
Fundum.buybackWallet (Fundum.sol#30) should be constant
Fundum.deadAddress (Fundum.sol#45-46) should be constant
Fundum.deployerWallet (Fundum.sol#26) should be constant
Fundum.lp1Tokens (Fundum.sol#52) should be constant
Fundum.lp2Tokens (Fundum.sol#53) should be constant
Fundum.lp2Wallet (Fundum.sol#31) should be constant
Fundum.marketingWallet (Fundum.sol#29) should be constant
Fundum.release2Wallet (Fundum.sol#27) should be constant
```

```
Fundum.buybackWallet (Fundum.sol#30) should be constant
Fundum.deadAddress (Fundum.sol#45-46) should be constant
Fundum.deployerWallet (Fundum.sol#26) should be constant
Fundum.lp1Tokens (Fundum.sol#52) should be constant
Fundum.lp2Tokens (Fundum.sol#53) should be constant
Fundum.lp2Wallet (Fundum.sol#31) should be constant
Fundum.marketingWallet (Fundum.sol#29) should be constant
Fundum.release2Wallet (Fundum.sol#27) should be constant
Fundum.teamWallet (Fundum.sol#28) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

increaseAllowance(address,uint256) should be declared external:
        - Fundum.increaseAllowance(address,uint256) (Fundum.sol#242-252)
decreaseAllowance(address,uint256) should be declared external:
        - Fundum.decreaseAllowance(address,uint256) (Fundum.sol#268-281)
updateMinSwapBalance(uint256) should be declared external:
        - Fundum.updateMinSwapBalance(uint256) (Fundum.sol#482-485)
prepareForPreSale() should be declared external:
        - Fundum.prepareForPreSale() (Fundum.sol#497-501)
afterPreSale() should be declared external:
        - Fundum.afterPreSale() (Fundum.sol#503-507)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (Ownable.sol#32-35)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (Ownable.sol#37-44)
getUnlockTime() should be declared external:
        - Ownable.getUnlockTime() (Ownable.sol#46-48)
getTime() should be declared external:
        - Ownable.getTime() (Ownable.sol#50-52)
lock(uint256) should be declared external:
        - Ownable.lock(uint256) (Ownable.sol#54-59)
unlock() should be declared external:
        - Ownable.unlock() (Ownable.sol#61-69)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external

Pragma version^0.8.4 (Uniswap.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Function IUniswapV2Pair.DOMAIN_SEPARATOR() (Uniswap.sol#67) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (Uniswap.sol#69) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (Uniswap.sol#99) is not in mixedCase
Function IUniswapV2Router01.WETH() (Uniswap.sol#143) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (Uniswap.sol#148) is too similar to IUniswapV2Router01.addLiquidity(address,address,uint25
6,uint256,uint256,uint256,address,uint256).amountBDesired (Uniswap.sol#149)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
. analyzed (20 contracts with 75 detectors), 128 result(s) found
```

Results

No major issues were found. Some false positive errors were reported by the tool. All the other issues have been categorized above according to their level of severity.

Closing Summary

Overall, smart contracts are very well written and adhere to guidelines. Many issues were discovered during the initial Audit; the majority of these issues were fixed by the Team.

Disclaimer

Quillhash audit is not a security warranty, investment advice, or an endorsement of the **Fundum** Contracts. This Audit does not provide a security or correctness guarantee of the audited smart contracts. The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One Audit cannot be considered enough. We recommend that the **Fundum** Team put in place a bug bounty program to encourage further Analysis of the smart contract by other third parties.

Audit Report
November, 2021

For

FUNDUM

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com