Versatile Finance

Smart Contract Security Audit

Nano Network Token

07 November 2022

Summary

Project Name: Nano Network

Contract Address: 0x17e1717FB52d00A74A07ACef8CD48369A4F77Ed2

Client contact: Nano Network Team

Blockchain: Binance smart chain

Language: Solidity

Buy Tax: 0 - 15%

Sell Tax: 0 - 15%

Token supply: 1,000,000,000

Token ticker: NANO

Decimals: 18

Marketing fee receiver: 0xbb23f1c21e316c614ded5be16ee2797f0c4a54cd

Buyback Wallet: 0x8ce840395329e3efd684a684b046fee786330b2a

Dividend Tracker: 0xd2c620d5f8c5fc0be5d44dffba38f83a9a188314

Get Payout Token: 0xe06f46afd251b06152b478d8ee3acea534063994

Contract deployer address: 0x7299336E094dd0f5a74f6bdCbfE7fECc401b81C4

Contract's current owner address: 0x7299336e094dd0f5a74f6bdcbfe7fecc401b81c4

Background

Versatile Finance was commissioned by Nano Network Team to perform an audit of the smart contract.

https://bscscan.com/address/0x17e1717FB52d00A74A07ACef8CD48369A4F77Ed2

The purpose of this audit was to achieve the following:

- Identify potential security issues with smart contracts
- Formally check the logic behind given smart contracts.

Information in this report should be used for understanding the risk exposure of smart contracts, and as a guide to improving the security posture of smart contracts by remediating the issues that were identified.

What is an audit

A smart contract audit is a comprehensive review process designed to discover logical errors, security vulnerabilities, and optimization opportunities within code. Versatile Finance manages this a step further by verifying economic logic to ensure the stability of smart contracts and highlighting privileged functionality to create a report that is easy to understand for developers and community members.

Techniques and Methods

- The code quality
- Use of best practices
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.
- Code risk issue analysis and recommendations
- Ownership privileges
- Code documentation and comments match logic and expected behavior.
- Token distribution and calculations are as per the intended behavior mentioned in the whitepaper.

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

Structural Analysis

We analyze the design patterns and structure of smart contracts. A thorough check is done to ensure the smart contract is structured in a way that will not have any issues.

Static Analysis

A static Analysis of Smart Contracts is done to identify contract vulnerabilities. In this step, a series of automated tools and manual tests are used to test the security of smart contracts.

Code Review / Manual Analysis

Manual Analysis or review of code is done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts is completely manually analyzed line by line, and the logic is checked and compared with what's mentioned in the whitepaper to make sure everything's functioned as intended.

Gas Consumption

We check the behavior of smart contracts in production. Manual testings are done in DEXs to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Issue Categories

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

High severity issues

No High Severity Issues Found

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-level severity issues

No Medium Severity Issues Found

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 can still be fixed. This can put users' funds at risk and has a medium to the high probability of exploitation.

Low-level severity issues

No Low Severity Issues Found

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. These issues have a low probability of occurring or may have a minimal impact.

Informational

No Informational Issues Found

These are severity four 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.

Centralization

1 Centralization issue found

Auto LP goes to the owner wallet, it should go to an unreachable address.

```
ftrace | funcSig
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
    );
}
```

Contracts Description Table

Contract	Type	Bases		
└	Function Name	Visibility	Mutability	Modifiers
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
IUniswapV2Pair	Interface			
└	name	External ❗		NO❗
└	symbol	External ❗		NO❗
└	decimals	External ❗		NO❗
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	transfer	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
└	DOMAIN_SEPARATOR	External ❗		NO❗

⌞	PERMIT_TYPEHASH	External ❗		NO❗
⌞	nonces	External ❗		NO❗
⌞	permit	External ❗	🛑	NO❗
⌞	MINIMUM_LIQUIDITY	External ❗		NO❗
⌞	factory	External ❗		NO❗
⌞	token0	External ❗		NO❗
⌞	token1	External ❗		NO❗
⌞	getReserves	External ❗		NO❗
⌞	price0CumulativeLast	External ❗		NO❗
⌞	price1CumulativeLast	External ❗		NO❗
⌞	kLast	External ❗		NO❗
⌞	mint	External ❗	🛑	NO❗
⌞	burn	External ❗	🛑	NO❗
⌞	swap	External ❗	🛑	NO❗
⌞	skim	External ❗	🛑	NO❗
⌞	sync	External ❗	🛑	NO❗
⌞	initialize	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❗
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		
└		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 🔒	🛑	
└	_mint	Internal 🔒	🛑	
└	_burn	Internal 🔒	🛑	
└	_approve	Internal 🔒	🛑	
└	_beforeTokenTransfer	Internal 🔒	🛑	

DividendPayingTokenOptionalInterface	Interface			
L	withdrawableDividendOf	External ❗		NO❗
L	withdrawnDividendOf	External ❗		NO❗
L	accumulativeDividendOf	External ❗		NO❗
DividendPayingTokenInterface	Interface			
L	dividendOf	External ❗		NO❗
L	distributeDividends	External ❗	💵	NO❗
L	withdrawDividend	External ❗	🛑	NO❗
SafeMath	Library			
L	add	Internal 🔒		
L	sub	Internal 🔒		
L	sub	Internal 🔒		
L	mul	Internal 🔒		
L	div	Internal 🔒		
L	div	Internal 🔒		
L	mod	Internal 🔒		

└	mod	Internal 🔒		
Ownable	Implementation	Context		
└		Public ❗	🛑	NO❗
└	owner	Public ❗		NO❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
SafeMathInt	Library			
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	sub	Internal 🔒		
└	add	Internal 🔒		
└	abs	Internal 🔒		
└	toUint256Safe	Internal 🔒		
SafeMathUint	Library			
└	toInt256Safe	Internal 🔒		
IUniswapV2Router 01	Interface			

L	factory	External ❗		NO❗
L	WETH	External ❗		NO❗
L	addLiquidity	External ❗	🛑	NO❗
L	addLiquidityETH	External ❗	💵	NO❗
L	removeLiquidity	External ❗	🛑	NO❗
L	removeLiquidityETH	External ❗	🛑	NO❗
L	removeLiquidityWithPermit	External ❗	🛑	NO❗
L	removeLiquidityETHWithPermit	External ❗	🛑	NO❗
L	swapExactTokensForTokens	External ❗	🛑	NO❗
L	swapTokensForExactTokens	External ❗	🛑	NO❗
L	swapExactETHForTokens	External ❗	💵	NO❗
L	swapTokensForExactETH	External ❗	🛑	NO❗
L	swapExactTokensForETH	External ❗	🛑	NO❗
L	swapETHForExactTokens	External ❗	💵	NO❗
L	quote	External ❗		NO❗
L	getAmountOut	External ❗		NO❗
L	getAmountIn	External ❗		NO❗
L	getAmountsOut	External ❗		NO❗
L	getAmountsIn	External ❗		NO❗

IUniswapV2Router02	Interface	IUniswapV2Router01		
L	removeLiquidityETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
L	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
L	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
L	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗	💵	NO❗
L	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
DividendPayingToken	Implementation	ERC20, DividendPayingTokenInterface, DividendPayingTokenOptionalInterface		
L		Public ❗	🛑	ERC20
L		External ❗	💵	NO❗
L	distributeDividends	Public ❗	💵	NO❗
L	withdrawDividend	Public ❗	🛑	NO❗
L	_withdrawDividendOfUser	Internal 🔒	🛑	
L	dividendOf	Public ❗		NO❗
L	withdrawableDividendOf	Public ❗		NO❗
L	withdrawnDividendOf	Public ❗		NO❗

L	accumulativeDividendOf	Public ❗		NO❗
L	_transfer	Internal 🔒	🛑	
L	_mint	Internal 🔒	🛑	
L	_burn	Internal 🔒	🛑	
L	_setBalance	Internal 🔒	🛑	
NanoNetwork	Implementation	ERC20, Ownable		
L		Public ❗	🛑	ERC20
L	decimals	Public ❗		NO❗
L		External ❗	💵	NO❗
L	updateStakingAmounts	Public ❗	🛑	onlyOwner
L	isTrading	Internal 🔒		
L	EnableTrading	External ❗	🛑	onlyOwner
L	setPresaleWallet	External ❗	🛑	onlyOwner
L	setExcludeFees	Public ❗	🛑	onlyOwner
L	setExcludeDividends	Public ❗	🛑	onlyOwner
L	setIncludeDividends	Public ❗	🛑	onlyOwner
L	setCanTransferBefore	External ❗	🛑	onlyOwner
L	setLimitsInEffect	External ❗	🛑	onlyOwner

L	setGasPriceLimit	External ❗	🛑	onlyOwner
L	setcooldowntimer	External ❗	🛑	onlyOwner
L	enableStaking	Public ❗	🛑	onlyOwner
L	stake	Public ❗	🛑	NO❗
L	setSwapTriggerAmount	Public ❗	🛑	onlyOwner
L	enableSwapAndLiquify	Public ❗	🛑	onlyOwner
L	setAutomatedMarketMakerPair	Public ❗	🛑	onlyOwner
L	setAllowCustomTokens	Public ❗	🛑	onlyOwner
L	setAllowAutoReinvest	Public ❗	🛑	onlyOwner
L	_setAutomatedMarketMakerPair	Private 🔒	🛑	
L	updateGasForProcessing	Public ❗	🛑	onlyOwner
L	setMarketingWallet	External ❗	🛑	onlyOwner
L	setBuyBackWallet	External ❗	🛑	onlyOwner
L	transferAdmin	Public ❗	🛑	onlyOwner
L	updateTransferFee	Public ❗	🛑	onlyOwner
L	updateFees	Public ❗	🛑	onlyOwner
L	getStakingInfo	External ❗		NO❗
L	getTotalDividendsDistributed	External ❗		NO❗
L	isExcludedFromFees	Public ❗		NO❗

L	withdrawableDividendOf	Public ❗		NO❗
L	dividendTokenBalanceOf	Public ❗		NO❗
L	getAccountDividendsInfo	External ❗		NO❗
L	getAccountDividendsInfoAtIndex	External ❗		NO❗
L	processDividendTracker	External ❗	🛑	NO❗
L	claim	External ❗	🛑	NO❗
L	getLastProcessedIndex	External ❗		NO❗
L	getNumberOfDividendTokenHolders	External ❗		NO❗
L	setAutoClaim	External ❗	🛑	NO❗
L	setReinvest	External ❗	🛑	NO❗
L	setDividendsPaused	External ❗	🛑	onlyOwner
L	isExcludedFromAutoClaim	External ❗		NO❗
L	isReinvest	External ❗		NO❗
L	_transfer	Internal 🔒	🛑	
L	getStakingBalance	Private 🔐		
L	swapAndLiquify	Private 🔐	🛑	
L	swapTokensForEth	Private 🔐	🛑	
L	updatePayoutToken	Public ❗	🛑	onlyOwner
L	getPayoutToken	Public ❗		NO❗

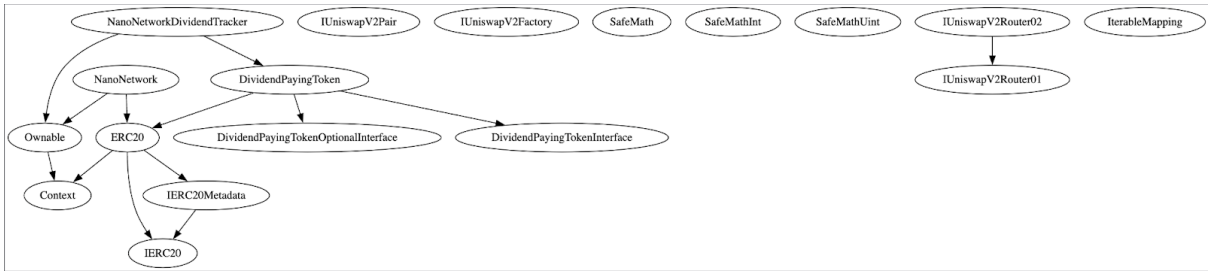
L	setMinimumTokenBalanceForAutoDividends	Public ❗	🛑	onlyOwner
L	setMinimumTokenBalanceForDividends	Public ❗	🛑	onlyOwner
L	addLiquidity	Private 🔐	🛑	
L	forceSwapAndSendDividends	Public ❗	🛑	onlyOwner
L	swapAndSendDividends	Private 🔐	🛑	
L	setPoolsForExcludeFromDailyVolume	External ❗	🛑	onlyOwner
L	setExcludeFromDailyVolumeLimit	External ❗	🛑	onlyOwner
L	setMaxDailySellLimit	External ❗	🛑	onlyOwner
L	todayVolume	External ❗		NO❗
L	airdropToWallets	External ❗	🛑	onlyOwner
NanoNetworkDividendTracker	Implementation	DividendPayingToken, Ownable		
L		Public ❗	🛑	DividendPayingToken
L	decimals	Public ❗		NO❗
L	name	Public ❗		NO❗
L	symbol	Public ❗		NO❗
L	_transfer	Internal 🔒		

L	withdrawDividend	Public ❗		NO❗
L	isExcludedFromAutoClaim	External ❗		onlyOwner
L	isReinvest	External ❗		onlyOwner
L	setAllowCustomTokens	External ❗	🛑	onlyOwner
L	setAllowAutoReinvest	External ❗	🛑	onlyOwner
L	excludeFromDividends	External ❗	🛑	onlyOwner
L	includeFromDividends	External ❗	🛑	onlyOwner
L	setAutoClaim	External ❗	🛑	onlyOwner
L	setReinvest	External ❗	🛑	onlyOwner
L	setMinimumTokenBalanceForAutoDividends	External ❗	🛑	onlyOwner
L	setMinimumTokenBalanceForDividends	External ❗	🛑	onlyOwner
L	setDividendsPaused	External ❗	🛑	onlyOwner
L	getLastProcessedIndex	External ❗		NO❗
L	getNumberOfTokenHolders	External ❗		NO❗
L	getAccount	Public ❗		NO❗
L	getAccountAtIndex	Public ❗		NO❗
L	setBalance	External ❗	🛑	onlyOwner
L	process	Public ❗	🛑	NO❗
L	processAccount	Public ❗	🛑	onlyOwner

L	updateUniswapV2Router	Public ❗	🛑	onlyOwner
L	updatePayoutToken	Public ❗	🛑	onlyOwner
L	getPayoutToken	Public ❗		NO❗
L	_reinvestDividendOfUser	Private 🔐	🛑	
L	_withdrawDividendOfUser	Internal 🔒	🛑	
IterableMapping	Library			
L	get	Internal 🔒		
L	getIndexOfKey	Internal 🔒		
L	getKeyAtIndex	Internal 🔒		
L	size	Internal 🔒		
L	set	Internal 🔒	🛑	
L	remove	Internal 🔒	🛑	

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable


NanoNetworkDividendTracker
IUniswapV2Pair
IUniswapV2Factory
SafeMath
SafeMathInt
SafeMathUint
IUniswapV2Router02
IterableMapping
NanoNetwork
DividendPayingToken
IUniswapV2Router01
Ownable
ERC20
DividendPayingTokenOptionalInterface
DividendPayingTokenInterface
Context
IERC20Metadata
IERC20

Owner privileges

The owner can update reward percentage in each staking duration

```
ftrace | funcSig
function updateStakingAmounts(uint256 duration, uint256 bonus)
    public
    onlyOwner
{
    require(stakingAmounts[duration] != bonus);
    require(bonus <= 100, "Staking bonus can't exceed 100");
    stakingAmounts[duration] = bonus;
    emit UpdateStakingAmounts(duration, bonus);
}
```

The owner can enable trading,once enable cannot disable again

```
ftrace | funcSig
function EnableTrading() external onlyOwner {
    launchblock = block.number;
    tradingEnabled = true;
    emit TradingEnabled();
}
```

The owner can whitelist pre-sale address

```
// use for pre sale wallet, adds all exclusions to it
ftrace | funcSig
function setPresaleWallet(address wallet) external onlyOwner {
    canTransferBeforeTradingIsEnabled[wallet] = true;
    _isExcludedFromFees[wallet] = true;
    _excludedFromCheckingDailyVolume[wallet] = true;
    dividendTracker.excludeFromDividends(wallet);
    emit SetPreSaleWallet(wallet);
}
```

The owner can include/exclude wallets from fees

```
// exclude a wallet from fees
ftrace | funcSig
function setExcludeFees(address account, bool excluded) public onlyOwner {
    _isExcludedFromFees[account] = excluded;
    emit ExcludeFromFees(account, excluded);
}
```

The owner can exclude wallets from rewards

```
// exclude from dividends (rewards)
ftrace | funcSig
function setExcludeDividends(address account) public onlyOwner {
    dividendTracker.excludeFromDividends(account);
}
```

The owner can include wallets from rewards.

```
// include in dividends
ftrace | funcSig
function setIncludeDividends(address account) public onlyOwner {
    dividendTracker.includeFromDividends(account);
    dividendTracker.setBalance(account, getStakingBalance(account));
}
```

The owner can enable/remove access to wallets who can do transactions before enabling trade.

```
//allow a wallet to trade before trading enabled
ftrace | funcSig
function setCanTransferBefore(address wallet, bool enable)
    external
    onlyOwner
{
    canTransferBeforeTradingIsEnabled[wallet] = enable;
}
```

The owner can enable/disable limits

```
// turn limits on and off
ftrace | funcSig
function setLimitsInEffect(bool value) external onlyOwner {
    limitsInEffect = value;
}
```

The owner can change sell cool down timer maximum up to 5 minutes

```
// set cooldown timer, can only be between 0 and 300 seconds (5 mins max)
ftrace | funcSig
function setcooldowntimer(uint256 value) external onlyOwner {
    require(value <= 300, "cooldown timer cannot exceed 5 minutes");
    cooldowntimer = value;
}
```

The owner can enable/disable staking

```
ftrace | funcSig
function enableStaking(bool enable) public onlyOwner {
    require(stakingEnabled != enable);
    stakingEnabled = enable;
    emit EnableStaking(enable);
}
```

The owner can change swap point

```
// rewards threshold
ftrace | funcSig
function setSwapTriggerAmount(uint256 amount) public onlyOwner {
    swapTokensAtAmount = amount * (10**18);
}
```

The owner can enable/disable swapping

```
ftrace | funcSig
function enableSwapAndLiquify(bool enabled) public onlyOwner {
    require(swapAndLiquifyEnabled != enabled);
    swapAndLiquifyEnabled = enabled;
    emit EnableSwapAndLiquify(enabled);
}
```

The owner can add/remove new pairs

```
ftrace | funcSig
function setAutomatedMarketMakerPair(address pair, bool value)
    public
    onlyOwner
{
    _setAutomatedMarketMakerPair(pair, value);
}
```

The owner can enable/disable custom tokens for rewards

```
ftrace | funcSig
function setAllowCustomTokens(bool allow) public onlyOwner {
    dividendTracker.setAllowCustomTokens(allow);
}
```

The owner can enable/disable reinvest.

```
ftrace | funcSig
function setAllowAutoReinvest(bool allow) public onlyOwner {
    dividendTracker.setAllowAutoReinvest(allow);
}
```

The owner can change max gas fee for process reward tracker.

```
ftrace | funcSig
function updateGasForProcessing(uint256 newValue) public onlyOwner {
    require(newValue >= 200000 && newValue <= 1000000);
    emit GasForProcessingUpdated(newValue, gasForProcessing);
    gasForProcessing = newValue;
}
```

The owner can change marketing wallet.

```
// set new marketing wallet
ftrace | funcSig
function setMarketingWallet(address wallet) external onlyOwner {
    _isExcludedFromFees[wallet] = true;
    _excludedFromCheckingDailyVolume[wallet] = true;
    marketingWallet = payable(wallet);
    emit updateMarketingWallet(wallet);
}
```

The owner can change buyback wallet

```
//set new buyback wallet
ftrace | funcSig
function setBuyBackWallet(address wallet) external onlyOwner {
    _isExcludedFromFees[wallet] = true;
    _excludedFromCheckingDailyVolume[wallet] = true;
    buybackWallet = payable(wallet);
    emit updateBuyBackWallet(wallet);
}
```

The owner can transfer ownership

```
ftrace | funcSig
function transferAdmin(address newOwner) public onlyOwner {
    dividendTracker.excludeFromDividends(newOwner);
    _isExcludedFromFees[newOwner] = true;
    _excludedFromCheckingDailyVolume[newOwner] = true;
    transferOwnership(newOwner);
}
```

The owner can change transfer fees maximum up to 5%

```
ftrace | funcSig
function updateTransferFee(uint256 newTransferFee) public onlyOwner {
    require(newTransferFee <= 5, "transfer fee cannot exceed 5%");
    transferFee = newTransferFee;
    emit UpdateTransferFee(transferFee);
}
```

The owner can change buy and sell fees each maximum up to 15%

```
ftrace | funcSig
function updateFees(
    uint256 marketingBuy,
    uint256 marketingSell,
    uint256 buybackBuy,
    uint256 buybackSell,
    uint256 liquidityBuy,
    uint256 liquiditySell,
    uint256 RewardsBuy,
    uint256 RewardsSell
) public onlyOwner {
    buyMarketingFees = marketingBuy;
    buyLiquidityFee = liquidityBuy;
    buyRewardsFee = RewardsBuy;
    buyBuyBackFee = buybackBuy;
    sellBuyBackFee = buybackSell;
    sellMarketingFees = marketingSell;
    sellLiquidityFee = liquiditySell;
    sellRewardsFee = RewardsSell;

    totalSellFees = sellRewardsFee
        .add(sellLiquidityFee)
        .add(sellMarketingFees)
        .add(sellBuyBackFee);

    totalBuyFees = buyRewardsFee
        .add(buyLiquidityFee)
        .add(buyMarketingFees)
        .add(buyBuyBackFee);

    require(
        totalSellFees <= 15 && totalBuyFees <= 15,
        "total fees cannot exceed 15% sell or buy"
    );

    emit UpdateFees(
        sellMarketingFees,
        sellLiquidityFee,
        sellRewardsFee,
        sellBuyBackFee,
        buyBuyBackFee,
        buyMarketingFees,
        buyLiquidityFee,
        buyRewardsFee
    );
}
```

The owner can pause reward tracker

```
ftrace | funcSig
function setDividendsPaused(bool value) external onlyOwner {
    dividendTracker.setDividendsPaused(value);
}
```

The owner can update reward tracker default payout token

```
ftrace | funcSig
function updatePayoutToken(address token) public onlyOwner {
    dividendTracker.updatePayoutToken(token);
    emit UpdatePayoutToken(token);
}
```

The owner can change minimum tokens balance to have get auto rewards

```
ftrace | funcSig
function setMinimumTokenBalanceForAutoDividends(uint256 value)
    public
    onlyOwner
{
    dividendTracker.setMinimumTokenBalanceForAutoDividends(value);
}
```

The owner can change minimum tokens balance to have rewards

```
ftrace | funcSig
function setMinimumTokenBalanceForDividends(uint256 value)
    public
    onlyOwner
{
    dividendTracker.setMinimumTokenBalanceForDividends(value);
}
```

The owner can manually trigger the swap

```
ftrace | funcSig
function forceSwapAndSendDividends(uint256 tokens) public onlyOwner {
    tokens = tokens * (10**18);
    uint256 totalAmount = buyAmount.add(sellAmount);
    uint256 fromBuy = tokens.mul(buyAmount).div(totalAmount);
    uint256 fromSell = tokens.mul(sellAmount).div(totalAmount);

    swapAndSendDividends(tokens);

    buyAmount = buyAmount.sub(fromBuy);
    sellAmount = sellAmount.sub(fromSell);
}
```

The owner can include/exclude wallets or contract from the calculating daily volume

```
ftrace | funcSig
function setPoolsForExcludeFromDailyVolume(address _pool, bool _flag)
    external
    onlyOwner
{
    _poolsToExcludeFromDaylyVolume[_pool] = _flag;
}
```

The owner can include/exclude wallets from the daily volume limit

```
ftrace | funcSig
function setExcludeFromDailyVolumeLimit(address _wallet, bool _flag)
    external
    onlyOwner
{
    _excludedFromCheckingDailyVolume[_wallet] = _flag;
}
```

The owner can change daily sell limit maximum up to 1.5 million

```
ftrace | funcSig
function setMaxDailySellLimit(uint256 _limit) external onlyOwner {
    require (_limit >= 1_500_000 ether, "can't be less than 1.5 million");
    maxDailySellLimit = _limit;
}
```

The owner can airdrop tokens from the owner wallet

```
ftrace | funcSig
function airdropToWallets(
    address[] memory airdropWallets,
    uint256[] memory amount
) external onlyOwner {
    require(
        airdropWallets.length == amount.length,
        "Arrays must be the same length"
    );
    require(
        airdropWallets.length <= 200,
        "Wallets list length must be <= 200"
    );
    for (uint256 i = 0; i < airdropWallets.length; i++) {
        address wallet = airdropWallets[i];
        uint256 airdropAmount = amount[i] * (10**18);
        super._transfer(msg.sender, wallet, airdropAmount);
        dividendTracker.setBalance(payable(wallet), balanceOf(wallet));
    }
}
```

Audit Results

Vulnerability Category	Status
Arbitrary Jump/Storage Write	pass
BRC20 Token standards	pass
Compiler errors	pass
Latest compiler version	pass
Authorization of function call to untrusted contract	pass
Dependence on Predictable Variables	pass
Ether/Token Theft	pass
Gas consumption	pass
Safemath features	pass
Fallback usage	pass
Deprecated items	pass
Redundant code	pass
Overriding variables	pass
Flash Loans	pass
Front Running	pass
Improper Events	pass
Improper Authorization Scheme	pass
Integer Over/Underflow	pass
Business logic issues	pass

Orcle issues	pass
Race Conditions	pass
Reentrancy	pass
Signature Issues	pass
Unbounded Loops	pass
Unused Code	pass
Pseudo random number generator (PRNG)	pass
Fake deposit	pass

Audit conclusion

Versatile Finance team has performed in-depth testing, line by line manual code review, and automated audit of the smart contract. The smart contract was analysed mainly for common smart contract vulnerabilities, exploits, manipulations, and hacks. According to the smart contract audit.

Smart contract functional Status: **PASS**

Number of risk issues: **1**

Solidity code functional issue level: **PASS**

Number of owner privileges: **30**

Centralization risk correlated to the active owner: **MEDIUM**

Smart contract active ownership: **ACTIVE**

Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice as of the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the disclaimer below – please make sure to read it in full.

DISCLAIMER: By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and Versatile Finance and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers, and other representatives) (Versatile Finance) owe no duty of care towards you or any other person, nor does Versatile Finance make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties, or other terms of any kind except as set out in this disclaimer, and Versatile Finance hereby excludes all representations, warranties, conditions, and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, Versatile Finance hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against Versatile Finance, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of the use of this report, and any reliance on this report. The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.