RugFreeCoins Audit

Mini Tennis Token

Smart Contract Security Audit

December 4, 2021

Contents

Audit details

Audited project

Mini Tennis Token

Contract Address

0xf844C1811C3156B1e9a7bae10Ce1Ff6b02a02D70

Client contact

Mini Tennis Token Team

Blockchain

Binance smart chain

Project website

https://minitennistoken.site/

Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice as at 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 Rugfreecoins and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (Rugfreecoins) owe no duty of care towards you or any other person, nor does Rugfreecoins 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 Rugfreecoins 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, Rugfreecoins hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against Rugfreecoins, 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 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.

Background

Rugfreecoins was commissioned by Mini Tennis Token to perform an audit of the smart contract.

https://bscscan.com/address/0xf844C1811C3156B1e9a7bae10Ce1Ff6b02a02D70

The focus of this audit is to verify that the smart contract is secure, resilient and working according to the specifications.

The information in this report should be used to understand the risk exposure of the smart contract, project feasibility, long term sustainability and as a guide to improve the security posture of the smart contract by remediating the issues that were identified.

About the project

Mini Tennis Token is a token built on the Binance Smart Chain that is based on a betting game related to Tennis. Each transaction, purchase incurs a 15% fee, and sell incur a 20% fee.

It will have a PVE: mode in which you will compete against the AI, but the winnings will be very limited.

And it will have a PVP mode: in which you will compete against other players in real-time and bet their MNT in each encounter.

In the store, users can buy their favorite tennis players or go for something cheaper by selecting them randomly and tennis players will be selected randomly; you will also have 3 items available that you must equip: Rackets, balls, and accessories (you can only use one of each). These characters, rackets, balls, and accessories will be able to level up, all will buy them with basic statistics and as you go up the level, the statistics will increase and will allow you to have an advantage over your opponents. In the same way, later we will add a Marketplace, where they will be able to market with their characters and objects (including those that you level up). We'll also add the «character merge» feature where you can add RARITY to the character or item in question.

Features

- ❖ 3% of each transaction when selling gets sent amongst all holders in ADA rewards. The holders will be eligible to receive ADA, every one hour, and rewards are proportional to how many tokens each individual holds.

- ❖ The **sustainability fee of 7% when buying and 9% when selling for marketing** is what allows Mini Tennis to hold the aforementioned promise. Tokens will be swapped into BNBs and will be sent to a marketing wallet, which will be allocated for manual buyback, marketing, and for the team. This way, the Mini Tennis token will have enough funds to promote the coin and spend for future development without selling tokens as the traditional way.

- ❖ **The fee of 3% when buying and 6% when the selling** will be swapped into BNB and will be allocated for the project.

- ❖ The additional component included under the sustainability section is a **liquidity fee of 2% when buying and selling**, which is a redistribution mechanism that ensures the trading pool always has sufficient liquidity. This is a key element for decentralized exchanges like Pancakeswap.

Tokenomics

15% fee when buying

- ❖ 3% of trade goes to holders pockets in ADA.
- ❖ 3% of trade goes to the project in BNB.
- ❖ 3% of trade goes to the manual buyback in BNB. (marketing wallet)
- ❖ 3% of trade goes to the marketing wallet in BNB. (marketing wallet)
- ❖ 2% of trade goes to the liquidity pool.
- ❖ 1% of trade goes to the team wallet in BNB. (marketing wallet).

20% fee when selling

- ❖ 3% of trade goes to holders pockets in ADA.
- ❖ 6% of trade goes to the project in BNB.
- ❖ 4% of trade goes to the manual buyback in BNB. (marketing wallet)
- ❖ 4% of trade goes to the marketing wallet in BNB. (marketing wallet)
- ❖ 2% of trade goes to the liquidity pool.
- ❖ 1% of trade goes to the team wallet in BNB. (marketing wallet)

Roadmap

Phase 1

- ❖ Website launch
- ❖ Initial Marketing Push
- ❖ Announcement Private and Public Presale
- ❖ Launch on PCS
- ❖ Minted Fantasy Characters
- ❖ Rugfree Coins Audit

Phase 2

- ❖ Huge Influencers
- ❖ DexTools trending
- ❖ Listing on Coinsniper, Coinhunt, etc.
- ❖ 5k Holders
- ❖ Listing on Coingecko
- ❖ Listing on CMC
- ❖ Market place website NFTs
- ❖ Game Update to its PVP Version

Phase 3

- ❖ Tennis ads on Big Games
- ❖ Huge influencers Push
- ❖ Exclusive CAMEOS with players
- ❖ Listing Hot Exchanges
- ❖ Certik Audit
- ❖ Constant game update

Target market and the concept

- ❖ Anyone who's interested in Crypto space with long-term investment plans.
- ❖ Anyone who's ready to earn a passive income in ADA by holding tokens.
- ❖ Anyone who's interested in trading tokens.
- ❖ Anyone who's interested in supporting the good cause.
- ❖ Anyone who's interested in collecting NFTs or trading NFTs and using it for the tennis game.
- ❖ Anyone who's interested in playing the mini tennis game and winning rewards.
- ❖ Anyone who's interested in taking part with the future plans of the Mini Tennis token.
- ❖ Anyone who's interested in making financial transactions with any other party using Mini Tennis token as the currency.

Core concept

Protocol
MiniTennis is ready with 3 use cases;

01 Firts Case

The first use case is that Holders will be rewarded with 3% of all purchases in ADA currency. They will have great potential for massive profits, with all the marketing and revenue taking place.

02 Second Case

The second use case will be with our P2E ecosystem. We will release a game in the future, which will reward MiniTennis owners even more.

Step by step they will know the progress of the game and obviously, the holders of the currency will be the most rewarded.

03 Third Case

The third use case will be the link with our next projects. We have 4 more millionaire projects in mind, to work in the very near future. Of them, you will have a funnel percentage towards this contract to ensure the life and durability of the token.

The ADA reward system

3% of each transaction when selling gets sent amongst all holders in ADA rewards. The holders will be eligible to receive ADA, in every one hour, and rewards are proportional to how many tokens each individual holds.

Sustainable mechanism

The **sustainability fee of 7% when buying and 9% when selling to the marketing wallet will be allocated for manual buyback, marketing, and team**. This is what allows Mini Tennis Token to promote the token and use funds to further the development of the platform. Tokens will be swapped into BNB and will be sent to a marketing and gaming pool wallet per transaction. This way, Mini Tennis Token will have access to the funds without selling tokens as the traditional way, which will enable them to consume funds without hurting the project.
.

The liquidity fee of 2% when buying and selling, which is a redistribution mechanism that ensures the trading pool always has sufficient liquidity.

Game:

MiniTennis will be based on a betting game mainly.

It will have a PVE: mode in which you will compete against the AI, but the winnings will be very limited.
And it will have a PVP mode: in which you will compete against other players in real time and bet their MNT in each encounter.

In the store, you can buy your favorite tennis players or go for something cheaper by selecting random and your tennis player will be selected randomly; you will also have 3 items available that you must equip: Rackets, balls and accessories (you can only use one of each). These characters, rackets, balls and accessories, will be able to level up, all will buy them with basic statistics and as you go up the level, the statistics will increase and will allow you to have an advantage over your opponents.

Game:

In the same way, later we will add a Marketplace, where they will be able to market with their characters and objects (including those that you level up) . We'll also add the "character merge" feature where you can add RARITY to the character or item in question.

IMPORTANT: We are currently in BETA version, little by little we will be developing everything proposed to ensure organic growth and the best experience within a BSC game.

WWW.MINITENNISTOKEN.SITE

Potential to grow with score points

1.	Project efficiency	10/10
2.	Project uniqueness	10/10
3	Information quality	9/10
4	Service quality	9/10
5	System quality	9/10
6	Impact on the community	9/10
7	Impact on the business	9/10
8	Preparing for the future	9/10
Total Points		**9.25/10**

Contract details

Token contract details for 04th December 2021

Contract name	MiniTennis
Contract address	0xf844C1811C3156B1e9a7bae10Ce1Ff6b02a02D70
Token supply	1,000,000,000,000
Token ticker	MNT
Decimals	18
Token holders	1
Transaction count	1
Core wallet address	0x0e8dd193e73e32873584e1de0b09341fb2596298
Dividend tracker	0xe5c3684f912496957c589a62ba1f31967e23b3f1
Marketing wallet address	0x670e23c192bf2c4e127b9245af9ec5a92f5be70d
Contract deployer address	0x24168B2D4cE31FE0A2Dec225cF8d3B6439020878
Contract's current owner address	0x516c3bb8459852062d393a1785da6c22f6f7179d

Token distribution

Tokens are distributed as follows:
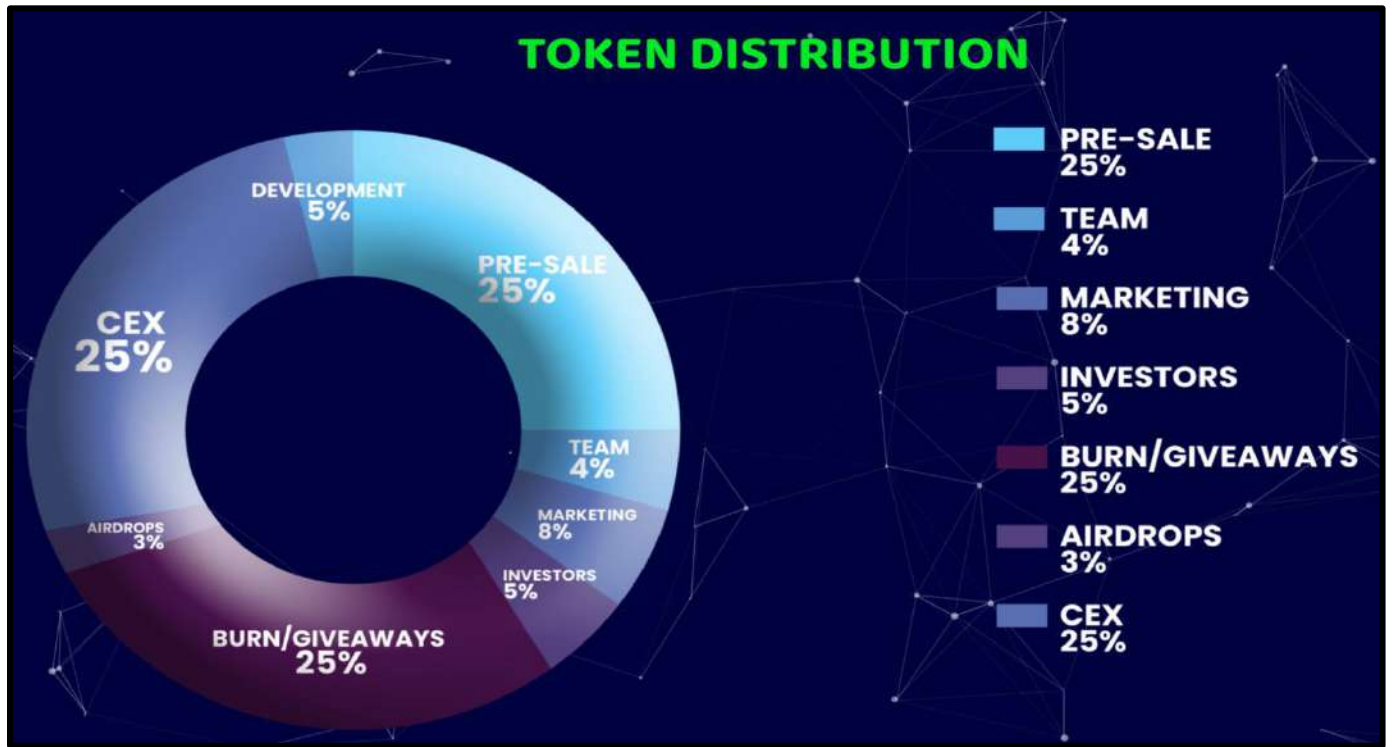
Contract code function details

No	Category	Item	Result
1	Coding conventions	BRC20 Token standards	pass
		compile errors	pass
		Compiler version security	pass
		visibility specifiers	pass
		Gas consumption	pass
		SafeMath features	pass
		Fallback usage	pass
		tx.origin usage	pass
		deprecated items	pass
		Redundant code	pass
		Overriding variables	pass
2	Function call audit	Authorization of function call	pass
		Low level function (call/delegate call) security	pass
		Returned value security	pass
		Selfdestruct function security	pass
3	Business security	Access control of owners	low issue
		Business logics	pass
		Business implementations	pass
4	Integer overflow/underflow		pass
5	Reentrancy		pass
6	Exceptional reachable state		pass
7	Transaction ordering dependence		pass
8	Block properties dependence		pass
9	Pseudo random number generator (PRNG)		pass
10	DoS (Denial of Service)		pass
11	Token vesting implementation		pass
12	Fake deposit		pass
13	Event security		pass

Contract description table

Below table represents the summary of the contracts and methods in the token contract. We scanned the whole contract and listed down all the Interfaces, functions and implementations with its visibility and mutability.

Contract	Type	Bases		
L	Function Name	Visibility	Mutability	Modifiers
MiniTennis	**Implementation**	**ERC20, Ownable**		
L		Public ⟦	⬢	ERC20
L		External ⟦	⬚⬚	NO⟦
L	updateDividendTracker	Public ⟦	⬢	onlyOwner
L	updateUniswapV2 Router	Public ⟦	⬢	onlyOwner
L	excludeFromFees	Public ⟦	⬢	onlyOwner
L	excludeMultipleAccountsFromFees	Public ⟦	⬢	onlyOwner
L	setSwapTokensAtAmount	External ⟦	⬢	onlyOwner
L	setMaxTokensToSwap	External ⟦	⬢	onlyOwner
L	setMarketingWallet	External ⟦	⬢	onlyOwner
L	setAirdropAddress	External ⟦	⬢	onlyOwner
L	setWaivePurchaseFees	External ⟦	⬢	onlyOwner
L	setSellFee	External ⟦	⬢	onlyOwner

L	setDividendsFee	External ◻	⬤	onlyOwner
L	setMarketingFee	External ◻	⬤	onlyOwner
L	setLiquiditFee	External ◻	⬤	onlyOwner
L	calculateTotalFees	Private 🔑	⬤	
L	setAutomatedMarketMakerPair	Public ◻	⬤	onlyOwner
L	blacklistAddress	External ◻	⬤	onlyOwner
L	_setAutomatedMarketMakerPair	Private 🔑	⬤	
L	setMaxTxAmount	External ◻	⬤	onlyOwner
L	updateGasForProcessing	Public ◻	⬤	onlyOwner
L	updateClaimWait	External ◻	⬤	onlyOwner
L	getClaimWait	External ◻		NO◻
L	getTotalDividendsDistributed	External ◻		NO◻
L	isExcludedFromFees	Public ◻		NO◻
L	withdrawableDividendOf	Public ◻		NO◻
L	dividendTokenBalanceOf	Public ◻		NO◻
L	excludeFromDividends	External ◻	⬤	onlyOwner
L	getAccountDividendsInfo	External ◻		NO◻
L	getAccountDividendsInfoAtIndex	External ◻		NO◻

15

L	processDividendTracker	External ‖	⬤	NO‖
L	claim	External ‖	⬤	NO‖
L	claimAddress	External ‖	⬤	onlyOwner
L	getLastProcessedIndex	External ‖		NO‖
L	setLastProcessedIndex	External ‖	⬤	onlyOwner
L	getNumberOfDividendTokenHolders	External ‖		NO‖
L	_transfer	Internal 🔒	⬤	
L	swapTokens	Private 🔏	⬤	
L	swapTokensOnDemand	External ‖	⬤	onlyOwner
L	swapTokensForBnb	Private 🔏	⬤	
L	addLiquidity	Private 🔏	⬤	
L	swapAndSendDividends	Private 🔏	⬤	
L	swapBnbForAda	Private 🔏	⬤	
MiniTennisDividendTracker	**Implementation**	**Ownable, DividendPayingToken**		
L		Public ‖	⬤	DividendPayingToken
L	_transfer	Internal 🔒	⬤	
L	withdrawDividend	Public ‖	⬤	NO‖

16

L	excludeFromDivid ends	External 🛡	⬢	onlyOwn er
L	updateClaimWait	External 🛡	⬢	onlyOwn er
L	setLastProcessedI ndex	External 🛡	⬢	onlyOwn er
L	getLastProcessedI ndex	External 🛡		NO🛡
L	getNumberOfToke nHolders	External 🛡		NO🛡
L	getAccount	Public 🛡		NO🛡
L	getAccountAtIndex	Public 🛡		NO🛡
L	canAutoClaim	Private 🔐		
L	setBalance	External 🛡	⬢	onlyOwn er
L	process	Public 🛡	⬢	NO🛡
L	processAccount	Public 🛡	⬢	onlyOwn er
Context	**Implementation**			
L	_msgSender	Internal 🔒		
L	_msgData	Internal 🔒		
DividendPayingToken	**Implementation**	**ERC20, Ownable, DividendPayingToken Interface, DividendPayingToken OptionalInterface**		

L		Public 🛢	⬤	ERC20
L	distributeADADividends	Public 🛢	⬤	onlyOwner
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 🔒	⬤	
DividendPayingToken Interface	**Interface**			
L	dividendOf	External 🛢		NO🛢
L	withdrawDividend	External 🛢	⬤	NO🛢
DividendPayingToken OptionalInterface	**Interface**			

ERC20	Implementation	Context, IERC20, IERC20Metadata		
L	withdrawableDividendOf	External ⬚		NO⬚
L	withdrawnDividendOf	External ⬚		NO⬚
L	accumulativeDividendOf	External ⬚		NO⬚
ERC20	**Implementation**	**Context, IERC20, IERC20Metadata**		
L		Public ⬚	⬤	NO⬚
L	name	Public ⬚		NO⬚
L	symbol	Public ⬚		NO⬚
L	decimals	Public ⬚		NO⬚
L	totalSupply	Public ⬚		NO⬚
L	balanceOf	Public ⬚		NO⬚
L	transfer	Public ⬚	⬤	NO⬚
L	allowance	Public ⬚		NO⬚
L	approve	Public ⬚	⬤	NO⬚
L	transferFrom	Public ⬚	⬤	NO⬚
L	increaseAllowance	Public ⬚	⬤	NO⬚
L	decreaseAllowance	Public ⬚	⬤	NO⬚
L	_transfer	Internal 🔒	⬤	

L	_mint	Internal 🔒	⬤	
L	_burn	Internal 🔒	⬤	
L	_approve	Internal 🔒	⬤	
L	_beforeTokenTransfer	Internal 🔒	⬤	
IERC20	**Interface**			
L	totalSupply	External ▯		NO▯
L	balanceOf	External ▯		NO▯
L	transfer	External ▯	⬤	NO▯
L	allowance	External ▯		NO▯
L	approve	External ▯	⬤	NO▯
L	transferFrom	External ▯	⬤	NO▯
IERC20Metadata	**Interface**	**IERC20**		
L	name	External ▯		NO▯
L	symbol	External ▯		NO▯
L	decimals	External ▯		NO▯
IterableMapping	**Library**			

∟	get	Public ▮		NO▮
∟	getIndexOfKey	Public ▮		NO▮
∟	getKeyAtIndex	Public ▮		NO▮
∟	size	Public ▮		NO▮
∟	set	Public ▮	⬤	NO▮
∟	remove	Public ▮	⬤	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▮
IUniswapV2Pair	**Interface**			
∟	name	External ▮		NO▮

21

└	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▯

22

L	price1CumulativeLast	External ▯		NO▯
L	kLast	External ▯		NO▯
L	mint	External ▯	⬤	NO▯
L	burn	External ▯	⬤	NO▯
L	swap	External ▯	⬤	NO▯
L	skim	External ▯	⬤	NO▯
L	sync	External ▯	⬤	NO▯
L	initialize	External ▯	⬤	NO▯
IUniswapV2Router01	**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	swapExactTokens ForTokens	External ꟻ	⬤	NO ꟻ
L	swapTokensForEx actTokens	External ꟻ	⬤	NO ꟻ
L	swapExactETHFor Tokens	External ꟻ	💵	NO ꟻ
L	swapTokensForEx actETH	External ꟻ	⬤	NO ꟻ
L	swapExactTokens ForETH	External ꟻ	⬤	NO ꟻ
L	swapETHForExact Tokens	External ꟻ	💵	NO ꟻ
L	quote	External ꟻ		NO ꟻ
L	getAmountOut	External ꟻ		NO ꟻ
L	getAmountIn	External ꟻ		NO ꟻ
L	getAmountsOut	External ꟻ		NO ꟻ
L	getAmountsIn	External ꟻ		NO ꟻ
IUniswapV2Router02	**Interface**	**IUniswapV2Router01**		
L	removeLiquidityET HSupportingFeeO nTransferTokens	External ꟻ	⬤	NO ꟻ
L	removeLiquidityET HWithPermitSupp ortingFeeOnTransf erTokens	External ꟻ	⬤	NO ꟻ
L	swapExactTokens ForTokensSupport ingFeeOnTransfer Tokens	External ꟻ	⬤	NO ꟻ

L	swapExactETHFor TokensSupporting FeeOnTransferTo kens	External 〗	🔳	NO〗
L	swapExactTokens ForETHSupporting FeeOnTransferTo kens	External 〗	⬤	NO〗
Ownable	**Implementation**	**Context**		
L		Public 〗	⬤	NO〗
L	owner	Public 〗		NO〗
L	renounceOwnershi p	Public 〗	⬤	onlyOwn er
L	transferOwnership	Public 〗	⬤	onlyOwn er
SafeMath	**Library**			
L	add	Internal 🔒		
L	sub	Internal 🔒		
L	sub	Internal 🔒		
L	mul	Internal 🔒		
L	div	Internal 🔒		
L	div	Internal 🔒		
L	mod	Internal 🔒		

L	mod	Internal 🔒		
SafeMathInt	**Library**			
L	mul	Internal 🔒		
L	div	Internal 🔒		
L	sub	Internal 🔒		
L	add	Internal 🔒		
L	abs	Internal 🔒		
L	toUint256Safe	Internal 🔒		
SafeMathUint	**Library**			
L	toInt256Safe	Internal 🔒		

Legend

Symbol	Meaning
🛑	**Function can modify state**
💵	**Function is payable**

26

Inheritance Hierarchy
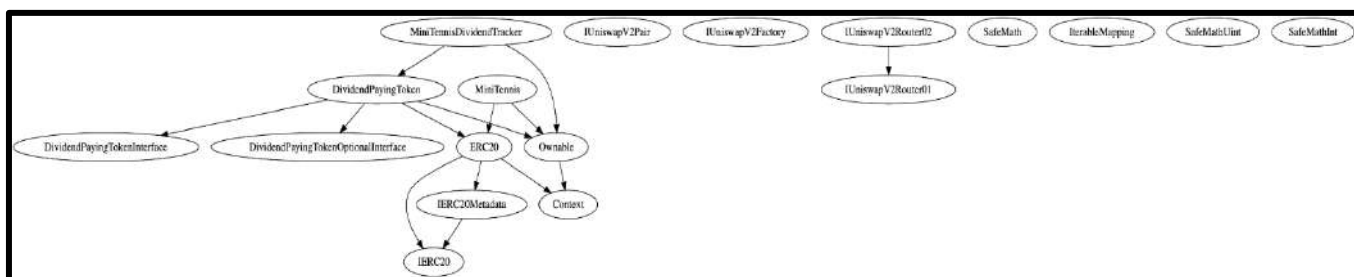
Security issue checking status

❖ **High severity issues**
No high severity issues found.

❖ **Medium severity issues**
No medium severity issues found.

❖ **Low severity issues**
The addLiquidity function calls the uniswapV2Router.addLiquidityETH function with the address specified as owner() for acquiring the generated LP tokens from the Mini Tennis Token-BNB pool. As a result, over time the _owner address will accumulate a significant portion of LP tokens. If the _owner is an EOA(Externally Owned Account), mishandling of its private key can have devastating consequences to the project as a whole.

Recommendation
We advise the address of the uniswapV2Router.addLiquidityETH function call to be replaced by the contract itself.

```solidity
function addLiquidity(uint256 tokenAmount, uint256 bnbAmount) private {

    // Approve token transfer to cover all possible scenarios
    _approve(address(this), address(uniswapV2Router), tokenAmount);

    // Add the liquidity
    uniswapV2Router.addLiquidityETH{value: bnbAmount}(
        address(this),
        tokenAmount,
        0, // Slippage is unavoidable
        0, // Slippage is unavoidable
        owner(),
        block.timestamp
    );

    emit SwapAndLiquify(tokenAmount, bnbAmount, tokenAmount);
}
```

Owner privileges

❖ The owner can update the dividend tracker.

```
ftrace | funcSig
function updateDividendTracker(address newAddress↑) public onlyOwner {
    require(
        newAddress↑ != address(dividendTracker),
        "MiniTennis: The dividend tracker already has that address"
    );

    MiniTennisDividendTracker newDividendTracker = MiniTennisDividendTracker(
        payable(newAddress↑)
    );

    require(
        newDividendTracker.owner() == address(this),
        "MiniTennis: The new dividend tracker must be owned by the MiniTennis token contract"
    );

    newDividendTracker.excludeFromDividends(address(newDividendTracker));
    newDividendTracker.excludeFromDividends(address(this));
    newDividendTracker.excludeFromDividends(owner());
    newDividendTracker.excludeFromDividends(deadWallet);
    newDividendTracker.excludeFromDividends(address(uniswapV2Router));

    emit UpdateDividendTracker(newAddress↑, address(dividendTracker));

    dividendTracker = newDividendTracker;
}
```

❖ The owner can update the router address.

```
ftrace | funcSig
function updateUniswapV2Router(address newAddress↑) public onlyOwner {
    require(
        newAddress↑ != address(uniswapV2Router),
        "MiniTennis: The router already has that address"
    );
    emit UpdateUniswapV2Router(newAddress↑, address(uniswapV2Router));
    uniswapV2Router = IUniswapV2Router02(newAddress↑);
    address _uniswapV2Pair = IUniswapV2Factory(uniswapV2Router.factory())
        .createPair(address(this), uniswapV2Router.WETH());
    uniswapV2Pair = _uniswapV2Pair;
}
```

28

❖ The owner can exclude wallets from fees.

```
ftrace | funcSig
function excludeFromFees(address account↑, bool excluded↑) public onlyOwner {
    require(
        _isExcludedFromFees[account↑] != excluded↑,
        "MiniTennis: Account is already the value of 'excluded'"
    );
    _isExcludedFromFees[account↑] = excluded↑;

    emit ExcludeFromFees(account↑, excluded↑);
}
```

❖ The owner can change the swap point.

```
ftrace | funcSig
function setSwapTokensAtAmount(uint256 amount↑) external onlyOwner {
    swapTokensAtAmount = amount↑;
}
```

❖ The owner can change the maximum swap token.

```
ftrace | funcSig
function setMaxTokensToSwap(uint256 amount↑) external onlyOwner {
    maxTokensToSwap = amount↑;
}
```

❖ The owner can update the marketing wallet.

```
ftrace | funcSig
function setMarketingWallet(address payable wallet↑) external onlyOwner {
    marketingWalletAddress = wallet↑;
}
```

❖ The owner can change the airdrop address.

```
ftrace | funcSig
function setAirdropAddress(address wallet↑) external onlyOwner {
    airdropAddress = wallet↑;
}
```

❖ The owner can waive off purchase fees.

```
ftrace | funcSig
function setWaivePurchaseFees(bool value↑) external onlyOwner {
    waivePurchaseFees = value↑;
}
```

❖ The owner can change all fees.

```
ftrace | funcSig
function setSellFee(uint256 value↑) external onlyOwner {
    sellFee = value↑;
}

ftrace | funcSig
function setDividendsFee(uint256 value↑) external onlyOwner {
    dividendsFee = value↑;
    calculateTotalFees();
}

ftrace | funcSig
function setMarketingFee(uint256 value↑) external onlyOwner {
    marketingFee = value↑;
    calculateTotalFees();
}

ftrace | funcSig
function setLiquiditFee(uint256 value↑) external onlyOwner {
    liquidityFee = value↑;
    calculateTotalFees();
}
```

❖ The owner can change the max transaction amount.

```
ftrace | funcSig
function setMaxTxAmount(uint256 value↑) external onlyOwner {
    maxTxAmount = value↑;
}
```

30

❖ The owner can update the claim wait.

```
ftrace | funcSig
function updateClaimWait(uint256 claimWait↑) external onlyOwner {
    dividendTracker.updateClaimWait(claimWait↑);
}
```

❖ The owner can exclude wallets from dividends.

```
ftrace | funcSig
function excludeFromDividends(address account↑) external onlyOwner {
    dividendTracker.excludeFromDividends(account↑);
}
```

❖ The owner can manually send rewards to wallet.

```
ftrace | funcSig
function claimAddress(address claimee↑) external onlyOwner {
    dividendTracker.processAccount(payable(claimee↑), false);
}
```

❖ The owner can manually set last processed index in reward.

```
ftrace | funcSig
function setLastProcessedIndex(uint256 index↑) external onlyOwner {
    dividendTracker.setLastProcessedIndex(index↑);
}
```

❖ The owner manually can swap the tokens.

```
ftrace | funcSig
function swapTokensOnDemand(
    bool liquify↑,
    bool sendToFee↑,
    bool sendDividends↑
) external onlyOwner {
    swapping = true;

    uint256 contractTokenBalance = balanceOf(address(this));
    if (contractTokenBalance > maxTokensToSwap) {
        contractTokenBalance = maxTokensToSwap;
    }

    // Determine the amount of tokens to swap
    uint256 tokensToSwap = 0;
    uint256 liquidityTokens = 0;
    uint256 feeTokens = 0;
    uint256 coreTokens = 0;
    uint256 dividendTokens = 0;
    if (liquify↑) {
        liquidityTokens = contractTokenBalance.mul(liquidityFee).div(
            totalFees
        );
        tokensToSwap += liquidityTokens.div(2);
    }

    if (sendToFee↑) {
        feeTokens = contractTokenBalance.mul(marketingFee).div(totalFees);
        coreTokens = contractTokenBalance.mul(coreFee).div(totalFees);
        tokensToSwap += feeTokens + coreTokens;
    }

    if (sendDividends↑) {
        dividendTokens = contractTokenBalance.mul(dividendsFee).div(
            totalFees
        );
        tokensToSwap += dividendTokens;
    }

    // Swap and magic
    if (tokensToSwap > 0) {
        uint256 bnbReceived = swapTokensForBnb(tokensToSwap);

        if (liquify↑) {
            uint256 half = liquidityTokens.div(2);
            uint256 bnbAmount = bnbReceived.mul(half).div(tokensToSwap);
            addLiquidity(liquidityTokens.sub(half), bnbAmount);
        }

        if (sendToFee↑) {
            uint256 bnbAmount = bnbReceived.mul(feeTokens).div(
                tokensToSwap
            );
            marketingWalletAddress.transfer(bnbAmount);
            bnbAmount = bnbReceived.mul(coreTokens).div(tokensToSwap);
            if (bnbAmount > 0) {
                coreWalletAddress.transfer(bnbAmount);
            }
        }

        if (sendDividends↑) {
            uint256 bnbAmount = bnbReceived.mul(dividendTokens).div(
```

Audit conclusion

While conducting the audit of the Mini Tennis Token smart contract, it was observed that there is nothing alarming with the code and it only contains a low severity issue.

.