



RugFreeCoins Audit



HOWLX Token Audit

Smart Contract Security Audit

December 2, 2021

Contents

Audit details	1
Disclaimer	2
Background	3
About the project	4
Target market and the concept	6
Potential to grow with score points	11
Total Points	11
Contract details	12
Token distribution	13
Contract code function details	14
Contract description table	15
Security issue checking status	28
Owner privileges	30
Audit conclusion	36

Audit details



Audited project

HOWLX Token



Contract Address

0x1D49ccE9C96c882737aD835F2dF2706A42E74116



Client contact

HOWLX Team



Blockchain

Binance smart chain



Project website

<https://howlxtoken.com/>

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 HOWLX to perform an audit of the smart contract.

<https://bscscan.com/token/0x1D49ccE9C96c882737aD835F2dF2706A42E74116>

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

HOWLX is a token built on the Binance Smart Chain that consists of two free-to-play + play-to-earn games with a profit-sharing model and a Metaverse with integrated cross-chain NFT interoperability. It also has a limited series of comic books (NFTs), a staking platform, and a 2d/3d Cartoon animation series (to be featured on Netflix). Each transaction, purchase and sell incur a 10% fee.

Features

- ❖ The **automatic BNB rewards** will be distributed among every holder proportional to how many tokens each individual holds in values of 4% when buying and selling.
- ❖ The **sustainability fee of 1% when buying and selling for marketing** is what allows HOWLX Token to hold the aforementioned promise. Tokens will be swapped into BNB and will be sent to a marketing wallet per transaction. This way, HOWLX Token will have enough funds to promote the coin and spend for future development without selling tokens as the traditional way.
- ❖ The additional component included under the sustainability section is a **liquidity fee of 2%**, which is a redistribution mechanism that ensures the trading pool always has sufficient liquidity. This is a key element for decentralized exchanges like Pancakeswap.
- ❖ The **gaming pool fee of 3% when buying and selling** is what allows HOWLX to allocate funds for the development of the games.

Tokenomics

10% fee when buying & selling

- ❖ 4% of trade goes to holders pockets in BNB.
- ❖ 2% of trade goes to the liquidity pool.
- ❖ 3% of trade goes to the game pool.
- ❖ 1% of trade goes to the marketing wallet.

Roadmap

Q4 2021

- ❖ Presale
- ❖ Pancake listing
- ❖ Worldwide marketing
- ❖ NFT Sale
- ❖ CG Listing
- ❖ CMC Listing
- ❖ More Listings
- ❖ A bunch of character reveal

Q1 2022

- ❖ Game 2 beta reveal
- ❖ Reward platform reveal
- ❖ Metaverse & Movie Update
- ❖ Game 2 Alpha test
- ❖ First CEX listing
- ❖ Certik Audit

Q2 2022

- ❖ New website
- ❖ Game 2 launch
- ❖ Game 2 Public Tournament
- ❖ Reward platform launch
- ❖ Game 1 Beta & Alpha test
- ❖ Metaverse land sale
- ❖ Update on Movie

Target market and the concept

Target market

- ❖ Anyone who's interested in Crypto space with long term investment plans.
- ❖ Anyone who's ready to earn a passive income in BNB by holding tokens.
- ❖ Anyone who's interested in owning Digital comic book series NFTs.
- ❖ Any casual or hardcore gamers out there play games and win rewards.
- ❖ Anyone who's interested in collecting NFTs or trading NFTs.
- ❖ Anyone who's interested in trading tokens.
- ❖ Anyone who's interested in taking part with the future plans of the HOWLX token.
- ❖ Anyone who's interested in making financial transactions with any other party using HOWLX or BNB as the currency.

Core concept

The BNB reward system

4% of each transaction when selling gets sent amongst all holders in BNB rewards. The holders will be eligible to receive BNB, in every one hour, and rewards are proportional to how many tokens each individual holds.

Sustainable mechanism

The **sustainability fee of 4% when buying and selling for marketing and development of game** is what allows HOWLX Token to promote the token and use funds to further 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, HOWLX 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.

WHY HAVE 1 GAME WHEN YOU CAN HAVE TWO BLOCKBUSTER GAMES?!!!!

GAME 1 - 5 vs 5 (PVP/PVE mode)

GAME 2 - 1 vs 1 (PVP/PVE mode) - Contains some cross over characters from Game 1.

GAME 1

Each player's character can be equipped with multiple weapons and a drone. The drone has a unique function to assist the player and can be set to different combat modes; mainly full assault or strategic defensive mode when engaging in gun fights.

The game will also have multiple seasons. Each season's storyline will be based off the comics and Netflix cartoon.



GAME 2

You will have the ability to design your own character and create your very own ultimate fighting machine. We have already created some interesting characters, however you can also modify them to your taste. You will earn points from beating people up and ultimately the more points you get the more token rewards you will be eligible for.



THE METAVERSE

THE HOWLX METAVERSE - MALLIAN ISLANDS

During the height of the pandemic the idea of the metaverse replacing the internet/ websites, became more of a reality than ever before.

Our goal is not just to create another wanna be Metaverse where people can build whatever they want or do what ever they like without having positive contribution to the real world. Our goal is to create a Metaverse that addresses real life problems and ultimately improves people's life.

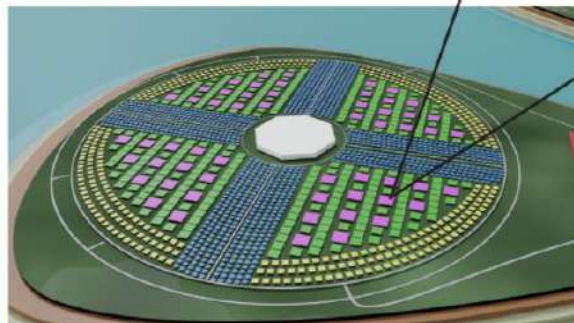
We have designed 5 identical Islands with the same layout and structure. (The reason we did this is to ensure fairness to all land owners). Each Island will have a town centre that will have a focus. For example; an Island's town centre can be focused on sports and essentially we will empower the Island to collaborate with industry leading entities like the NFL, NBA FIEA etc to create value and solve real life problems.

WHAT CAN LAND OWNERS DO IN THIS METAVERSE:

You will be able to build anything you want on the land. Our ambition is to make the metaverse plug and play; this is to allow anyone to easily design structures without any coding or experience.

Each Island will essentially have its own core functions and will be aimed at collaborating with industry leading entities to tackle real life problems.

***The land owners will have the ability to vote to decide what they want their city centre to focus on.**



BACKGROUND STORY OF THE MALLIAN ISLANDS

The Mallian Islands is a location that features in one of the missions in the HOWLX game. The six mysterious looking Islands were discovered by the founders of Mallian corp (The company responsible for creating

CLASSIFIED

The center Island was the Headquarters of Mallian corp. The other 5 Islands were converted into resorts/ holiday Island. However the 6 Islands were later seized by the authorities when a journalist infiltrated the HQ and exposed the secrets of the company by releasing a video footage of

CLASSIFIED

CLASSIFIED



NFTs



Aside from our game assets, we will be auctioning off our 5 limited series digital comic book on Opensea!



We are going big into the TV screens!!
NETFLIX ANIMATION MOVIE AND CARTOON SERIES

Potential to grow with score points

1.	Project efficiency	9/10
2.	Project uniqueness	9/10
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		9/10

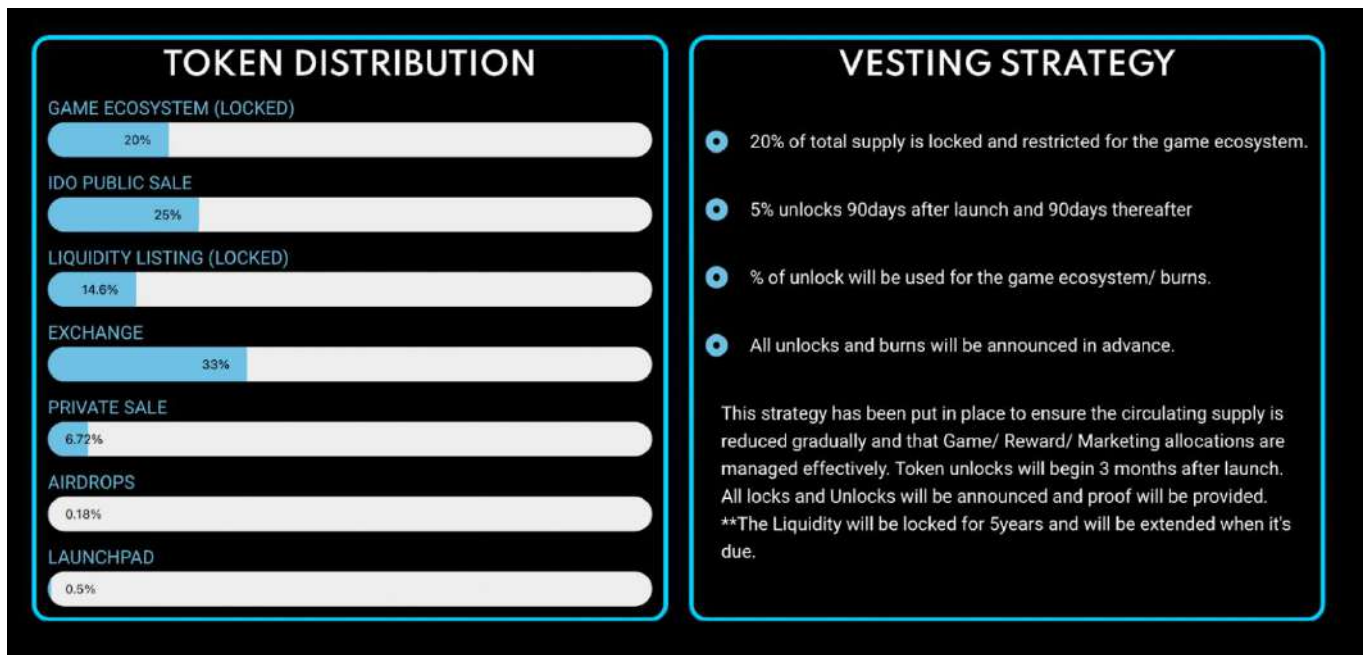
Contract details

Token contract details for 02nd December 2021

Contract name	HOWLX TOKEN
Contract address	0x1D49ccE9C96c882737aD835F2dF2706A42E74116
Token supply	100,000,000,000,000
Token ticker	HWXT
Decimals	9
Token holders	1
Transaction count	1
Dividend tracker	0x33483b0de44ea09d9df148cc5f772368e870d4d4
Game wallet	0x8abdf276fda153fcacc609f78536e71d1959bbd7
Marketing wallet	0x01186047926b31b4dafcf3afee749ae58628af60
Contract deployer address	0x6269eb79255e366CC2a4bDb4258301EFFB02A231
Contract's current owner address	0xf43b8dcf52ab41783d950d843c6d413960caa054

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	pass
		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
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 








Context	Implementation			
L	_msgSender	Internal 		
L	_msgData	Internal 		
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 		
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 		
SafeMathUint	Library			
L	toInt256Safe	Internal 		












SafeMathInt	Library			
L	mul	Internal 		
L	div	Internal 		
L	sub	Internal 		
L	add	Internal 		
L	abs	Internal 		
L	toUint256Safe	Internal 		
DividendPayingTokenInterface	Interface			
L	dividendOf	External 		NO 
L	distributeDividends	External 		NO 
L	withdrawDividend	External 		NO 
DividendPayingTokenOptionalInterface	Interface			
L	withdrawableDividendOf	External 		NO 
L	withdrawnDividendOf	External 		NO 
L	accumulativeDividendOf	External 		NO 














DividendPayingToken	Implementation	ERC20, DividendPayingTo kenInterface, DividendPayingTo kenOptionalInterfa ce		
L		Public 		ERC20
L		External 		NO 
L	distributeDividend s	Public 		NO 
L	withdrawDividend	Public 		NO 
L	_withdrawDividend OfUser	Internal 		
L	dividendOf	Public 		NO 
L	withdrawableDivid endOf	Public 		NO 
L	withdrawnDividend Of	Public 		NO 
L	accumulativeDivid endOf	Public 		NO 
L	_transfer	Internal 		
L	_mint	Internal 		
L	_burn	Internal 		
L	_setBalance	Internal 		
IterableMapping	Library			
L	get	Public 		NO 





































L	getIndexOfKey	Public ¶		NO¶
L	getKeyAtIndex	Public ¶		NO¶
L	size	Public ¶		NO¶
L	set	Public ¶	⦿	NO¶
L	remove	Public ¶	⦿	NO¶
Ownable	Implementation	Context		
L		Public ¶	⦿	NO¶
L	owner	Public ¶		NO¶
L	renounceOwnership	Public ¶	⦿	onlyOwner
L	transferOwnership	Public ¶	⦿	onlyOwner
IUniswapV2Pair	Interface			
L	name	External ¶		NO¶
L	symbol	External ¶		NO¶
L	decimals	External ¶		NO¶
L	totalSupply	External ¶		NO¶
L	balanceOf	External ¶		NO¶
L	allowance	External ¶		NO¶

L	approve	External ¶		NO¶
L	transfer	External ¶		NO¶
L	transferFrom	External ¶		NO¶
L	DOMAIN_SEPARATOR	External ¶		NO¶
L	PERMIT_TYPEHASH	External ¶		NO¶
L	nonces	External ¶		NO¶
L	permit	External ¶		NO¶
L	MINIMUM_LIQUIDITY	External ¶		NO¶
L	factory	External ¶		NO¶
L	token0	External ¶		NO¶
L	token1	External ¶		NO¶
L	getReserves	External ¶		NO¶
L	price0CumulativeLast	External ¶		NO¶
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¶
IUniswapV2Factory	Interface			
L	feeTo	External ¶		NO¶
L	feeToSetter	External ¶		NO¶
L	getPair	External ¶		NO¶
L	allPairs	External ¶		NO¶
L	allPairsLength	External ¶		NO¶
L	createPair	External ¶		NO¶
L	setFeeTo	External ¶		NO¶
L	setFeeToSetter	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	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 ¶
HOWLX	Implementation	ERC20, Ownable		
L		Public ¶		ERC20
L		External ¶		NO ¶
L	updateDividendTracker	Public ¶		onlyOwner
L	updateUniswapV2Router	Public ¶		onlyOwner
L	excludeFromFees	Public ¶		onlyOwner
L	excludeMultipleAccountsFromFees	Public ¶		onlyOwner
L	excludeFromRewards	Public ¶		onlyOwner
L	allowAddLiquidity	Public ¶		onlyOwner
L	setAutomatedMarketMakerPair	Public ¶		onlyOwner

L	_setAutomatedMarketMakerPair	Private 		
L	updateMaxBuyTransactionAmount	Public 		onlyOwner
L	updateMaxSellTransactionAmount	Public 		onlyOwner
L	updateMaxWalletToken	Public 		onlyOwner
L	updateSwapTokensAtAmount	Public 		onlyOwner
L	updateMarketingWallet	Public 		onlyOwner
L	startTrading	Public 		onlyOwner
L	liftAllLimits	Public 		onlyOwner
L	updateGasForProcessing	Public 		onlyOwner
L	updateClaimWait	External 		onlyOwner
L	updateGameFeeReceiver	External 		onlyOwner
L	updateBuyFees	External 		onlyOwner
L	updateSellFees	External 		onlyOwner
L	getClaimWait	External 		NO 
L	getTotalDividendsDistributed	External 		NO 
L	isExcludedFromFees	Public 		NO 
L	withdrawableDividendOf	Public 		NO 
L	dividendTokenBalanceOf	Public 		NO 

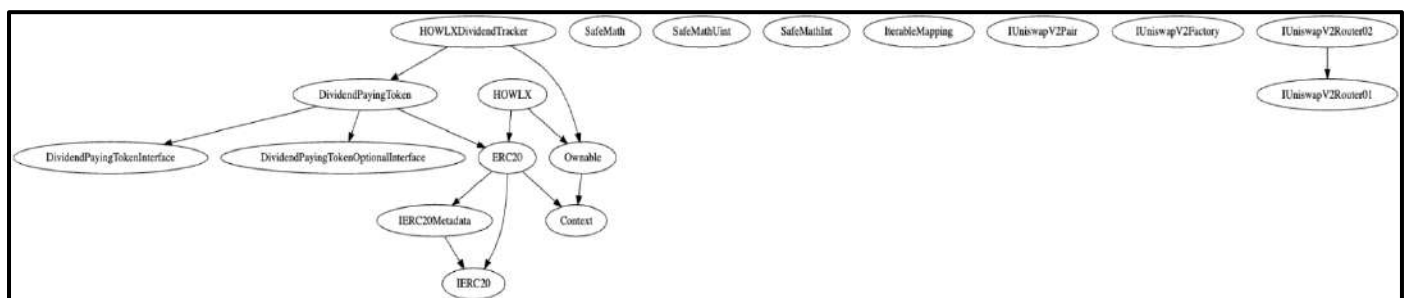
L	getAccountDividendsInfo	External 🔒		NO 🔒
L	getAccountDividendsInfoAtIndex	External 🔒		NO 🔒
L	processDividendTracker	External 🔒	⬢	NO 🔒
L	claim	External 🔒	⬢	NO 🔒
L	getLastProcessedIndex	External 🔒		NO 🔒
L	getNumberOfDividendTokenHolders	External 🔒		NO 🔒
L	getLiquidityIsAdded	Public 🔒		NO 🔒
L	setLiquidityIsAdded	External 🔒	⬢	onlyOwner
L	_transfer	Internal 🔒	⬢	
L	swapAndLiquify	Private 🔒	⬢	
L	swapTokensForEth	Private 🔒	⬢	
L	addLiquidity	Private 🔒	⬢	
L	swapAndSendDividends	Private 🔒	⬢	
HOWLXDividendTracker	Implementation	DividendPayingToken, Ownable		
L		Public 🔒	⬢	DividendPayingToken
L	_transfer	Internal 🔒	⬢	
L	withdrawDividend	Public 🔒	⬢	NO 🔒

L	excludeFromDividends	External ¶		onlyOwner
L	updateClaimWait	External ¶		onlyOwner
L	getLastProcessedIndex	External ¶		NO¶
L	getNumberOfTokenHolders	External ¶		NO¶
L	getAccount	Public ¶		NO¶
L	getAccountAtIndex	Public ¶		NO¶
L	canAutoClaim	Private		
L	setBalance	External ¶		onlyOwner
L	process	Public ¶		NO¶
L	processAccount	Public ¶		onlyOwner

Legend

Symbol	Meaning
	Function can modify state
	Function is payable

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

No low severity issues found.

❖ Informational

- The owner can enable/disable trading.

```
ftrace | funcSig
function startTrading(bool _status↑) public onlyOwner {
    tradingIsEnabled = _status↑;
}
```

- The owner can update all buy and sell fees.

```
// update total buy fees
ftrace | funcSig
function updateBuyFees(
    uint256 rewardFee↑,
    uint256 liquidityFee↑,
    uint256 marketingFee↑,
    uint256 gameFee↑
) external onlyOwner {
    buyRewardFee = rewardFee↑;
    buyLiquidityFee = liquidityFee↑;
    buyMarketingFee = marketingFee↑;
    buyGameFee = gameFee↑;

    // calculate total buy fees
    totalBuyFees = buyRewardFee
        .add(buyLiquidityFee)
        .add(buyMarketingFee)
        .add(buyGameFee);
}

// update total sell fees
ftrace | funcSig
function updateSellFees(
    uint256 rewardFee↑,
    uint256 liquidityFee↑,
    uint256 marketingFee↑,
    uint256 gameFee↑
) external onlyOwner {
    sellRewardFee = rewardFee↑;
    sellLiquidityFee = liquidityFee↑;
    sellMarketingFee = marketingFee↑;
    sellGameFee = gameFee↑;

    // calculate total sell fees
    totalSellFees = sellRewardFee
        .add(sellLiquidityFee)
        .add(sellMarketingFee)
        .add(sellGameFee);
}
```

Owner privileges

- ❖ The owner can update the dividend tracker.

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

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

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

    newDividendTracker.excludeFromDividends(address(newDividendTracker));
    newDividendTracker.excludeFromDividends(address(this));
    newDividendTracker.excludeFromDividends(owner());
    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),
        "HOWLX: The router already has that address"
    );

    emit UpdateUniswapV2Router(newAddress↑, address(uniswapV2Router));
    uniswapV2Router = IUniswapV2Router02(newAddress↑);
}
```

- ❖ The owner can exclude wallets from fees.

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

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

- ❖ The owner can exclude wallets from rewards.

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

- ❖ The owner can allow wallets to add liquidity when trading disable.

```
ftrace | funcSig
function allowAddLiquidity(address account↑) public onlyOwner {
    canTransferBeforeLiquidityIsEnabled[account↑] = true;
}
```

- ❖ The owner can add/remove marker pair.

```
ftrace | funcSig
function setAutomatedMarketMakerPair(address pair↑, bool value↑)
    public
    onlyOwner
{
    require(
        pair↑ != uniswapV2Pair,
        "HOWLX: The PancakeSwap pair cannot be removed from automatedMarketMakerPairs"
    );

    _setAutomatedMarketMakerPair(pair↑, value↑);
}
```


- ❖ The owner can update transaction limit and max wallet token amount.

```
ftrace | funcSig
function updateMaxBuyTransactionAmount(uint256 newMaxBuyTransactionAmount↑)
    public
    onlyOwner
{
    require(
        newMaxBuyTransactionAmount↑ != maxBuyTransactionAmount,
        "HOWLX: The max buy transaction amount is already this amount"
    );
    emit maxBuyTransactionAmountUpdated(
        newMaxBuyTransactionAmount↑,
        maxBuyTransactionAmount
    );
    maxBuyTransactionAmount = newMaxBuyTransactionAmount↑;
}

ftrace | funcSig
function updateMaxSellTransactionAmount(uint256 newMaxSellTransactionAmount↑)
    public
    onlyOwner
{
    require(
        newMaxSellTransactionAmount↑ != maxSellTransactionAmount,
        "HOWLX: The max sell transaction amount is already this amount"
    );
    emit maxSellTransactionAmountUpdated(
        newMaxSellTransactionAmount↑,
        maxSellTransactionAmount
    );
    maxSellTransactionAmount = newMaxSellTransactionAmount↑;
}

ftrace | funcSig
function updateMaxWalletToken(uint256 newMaxWalletToken↑) public onlyOwner {
    require(
        newMaxWalletToken↑ != maxWalletToken,
        "HOWLX: The max wallet token is already this amount"
    );
    emit maxWalletTokenUpdated(newMaxWalletToken↑, maxWalletToken);
    maxWalletToken = newMaxWalletToken↑;
}
```

- ❖ The owner can update swap point.

```
ftrace | funcSig
function updateswapTokensAtAmount(uint256 newswapTokensAtAmount↑)
{
    public
    onlyOwner

    require(
        newswapTokensAtAmount↑ != swapTokensAtAmount,
        "HOWLX: The swap tokens at amount is already this amount"
    );
    emit setswapTokensAtAmount(newswapTokensAtAmount↑, swapTokensAtAmount);
    swapTokensAtAmount = newswapTokensAtAmount↑;
}
```

- ❖ The owner can update marketing wallet.

```
ftrace | funcSig
function updateMarketingWallet(address newMarketingWallet↑)
{
    public
    onlyOwner

    require(
        newMarketingWallet↑ != marketingWallet,
        "HOWLX: The marketing wallet is already this address"
    );
    excludeFromFees(newMarketingWallet↑, true);
    emit MarketingWalletUpdated(newMarketingWallet↑, marketingWallet);
    marketingWallet = newMarketingWallet↑;
}
```

- ❖ The owner can enable/disable trading.

```
ftrace | funcSig
function startTrading(bool _status↑) public onlyOwner {
    tradingIsEnabled = _status↑;
}
```

- ❖ The owner can update maximum gas usage amount when sending rewards.

```
ftrace | funcSig
function updateGasForProcessing(uint256 newValue↑) public onlyOwner {
    require(
        newValue↑ >= 200000 && newValue↑ <= 500000,
        "HOWLX: gasForProcessing must be between 200,000 and 500,000"
    );
    require(
        newValue↑ != gasForProcessing,
        "HOWLX: Cannot update gasForProcessing to same value"
    );
    emit GasForProcessingUpdated(newValue↑, gasForProcessing);
    gasForProcessing = newValue↑;
}

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

- ❖ The owner can update game fee wallet.

```
// update game Fee receiver
ftrace | funcSig
function updateGameFeeReceiver(address newGameFeeReceiver↑)
    external
    onlyOwner
{
    gameWallet = newGameFeeReceiver↑;
}
```


- ❖ The owner can update all buy and sell fees.

```
// update total buy fees
ftrace | funcSig
function updateBuyFees(
    uint256 rewardFee↑,
    uint256 liquidityFee↑,
    uint256 marketingFee↑,
    uint256 gameFee↑
) external onlyOwner {
    buyRewardFee = rewardFee↑;
    buyLiquidityFee = liquidityFee↑;
    buyMarketingFee = marketingFee↑;
    buyGameFee = gameFee↑;

    // calculate total buy fees
    totalBuyFees = buyRewardFee
        .add(buyLiquidityFee)
        .add(buyMarketingFee)
        .add(buyGameFee);
}

// update total sell fees
ftrace | funcSig
function updateSellFees(
    uint256 rewardFee↑,
    uint256 liquidityFee↑,
    uint256 marketingFee↑,
    uint256 gameFee↑
) external onlyOwner {
    sellRewardFee = rewardFee↑;
    sellLiquidityFee = liquidityFee↑;
    sellMarketingFee = marketingFee↑;
    sellGameFee = gameFee↑;

    // calculate total sell fees
    totalSellFees = sellRewardFee
        .add(sellLiquidityFee)
        .add(sellMarketingFee)
        .add(sellGameFee);
}
```

Audit conclusion

While conducting the audit of the HOWLX Token smart contract, it was observed that there is nothing alarming with the code and it only contains informational concerns.