RugFreeCoins Audit

ETHFan Burn Token

Smart Contract Security Audit

February 26, 2022

Contents

Audit details

Audited project
ETHFan burn Token

Contract Address

0x002d3C280f719c4B0444680A8C4B1785b6cC2A59

Client contact

Meta ETHFan Burn Team

Blockchain

Binance smart chain

Project website

https://etb.ethfan.club

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 BNBBox Team to perform an audit of the smart contract.

https://bscscan.com/token/0x002d3C280f719c4B0444680A8C4B1785b6cC2A59

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 improving the security posture of the smart contract by remediating the issues that were identified.
.

About the project

With the benefits of the holders in mind, ETH Fan Burn Token (EFB) is a token created to complete the ETH Fan Token Ecosystem. It allows EFT Ecosystem to create an Infinite-Compounding-Loop mechanism of rewarding and earning that has never been seen before.

EFB is designed to use a certain portion of the tax from each transaction to buy back EFT and reward EFB holders with EFT – which in turn EFT will reward Binance Pegged ETH

EFB is also designed to use a certain portion of the tax from each transaction to buy back and sent EFT to the dead wallet – which in turn will reduce the EFT circulating supply permanently.

Since the buyback is done on an open market, it will directly impact the price as well as reduce the circulating supply of EFT. Furthermore, because of how the EFT reward mechanism works, the reduced circulating supply as well as EFT rewarded will make the EFT holder's ETH reward portion proportionally larger over time.

By introducing EFB to the ecosystem, EFT holders will now have the option to compound their ETH rewards into EFB and help to actively reduce EFT supply as well as earn EFT and ETH at the same time. ETH Fan Token Ecosystem is a community-driven project and by introducing EFB to the ecosystem, The community will be actively participating to help EFT reach its maximum potential.

Features

- The **ETHFan Burn rewards** will be distributed in ETF tokens among every holder proportional to how many tokens each individual holds in values of **4% when buying and selling.**

- The **sustainability fee of 3% when buying and selling for marketing** is what allows ETHFan Burn to hold the aforementioned promise. Tokens will be swapped into BNB and will be sent to a marketing wallet. This way, ETHFan Burn Token will have enough funds to promote the coin and spend for future development and marketing without selling tokens as the traditional way.

- **The fee of 1% when buying and selling** is used for buyback ETF tokens and burn to sustain the ETH Fan Token Ecosystem.

- The additional component included under the sustainability section is a **liquidity fee of 1% when buying and selling**, which is a redistribution mechanism that ensures the trading pool always has sufficient liquidity.

Tokenomics

9% fee when buying and selling

- 4% of trade goes to holders pockets in EFT token rewards.
- 3% of trade goes to the development wallet.
- 1% of trade goes to the buyback and burn of EFT.
- 1% of trade goes to the liquidity pool

Target market and the concept

Target market

- Anyone who’s interested in the Crypto space with long-term investment plans.
- Anyone who’s ready to earn a passive income by holding tokens.
- Anyone who’s interested in trading tokens.
- Anyone who’s interested in taking part with staking.
- Anyone who’s interested in taking part in decision-making.
- Anyone who’s interested in taking part in the future plans of the ETHFan Burn token.
- Anyone who’s interested in making financial transactions with any other party using ETHFan Burn as the currency.

Core concept

The ETHFan Burn reward system

4% of each transaction when buying and selling gets converted to EFT tokens and is split amongst all holders. Holders will be eligible to receive tokens in each transaction and rewards are proportional to how many tokens each individual holds.

Sustainable mechanism

The **sustainability fee of 4% when buying and selling for dev and marketing** is what allows ETHFan Burn 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 wallet and dev wallet. This way, ETHFan Burn 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 1% when buying and selling, is a redistribution mechanism that ensures the trading pool always has sufficient liquidity.

The fee of 1% when buying and selling is used for buyback ETF tokens and burn to sustain the ETH Fan Token Ecosystem.

Potential to grow with score points

1.	Project efficiency	9/10
2.	Project uniqueness	9/10
3	Information quality	9/10
4	Service quality	10/10
5	System quality	10/10
6	Impact on the community	10/10
7	Impact on the business	10/10
8	Preparing for the future	10/10
Total Points		**9.625/10**

Contract details

Token contract details for 26th February 2022

Contract name	ETHFan Burn
Contract address	0x002d3C280f719c4B0444680A8C4B1785b6cC2A59
Token supply	1,000,000,000,000
Token ticker	$EFB
Decimals	9
Token holders	1
Transaction count	2
Dividend tracker	0xa0eae1d9a20cab8544f929e72f0236397a13db86
Marketing fee receiver	0x23c37ace44ab62da0ab74a371a6ec59d94cb1c33
Contract deployer address	0x97361F6979756A647458DC3EAAB802Db416997a3
Contract’s current owner address	0xfc098a2ec7f9fd885c172aff0ce3a9a0eeaf03e6

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
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❗
SafeMath	**Library**			
L	tryAdd	Internal 🔒		
L	trySub	Internal 🔒		
L	tryMul	Internal 🔒		
L	tryDiv	Internal 🔒		
L	tryMod	Internal 🔒		
L	add	Internal 🔒		
L	sub	Internal 🔒		
L	mul	Internal 🔒		
L	div	Internal 🔒		
L	mod	Internal 🔒		
L	sub	Internal 🔒		
L	div	Internal 🔒		
L	mod	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❗
IDividend Distributor	**Interface**			
L	setDistributionCriteria	External ❗	🛑	NO❗
L	setShare	External ❗	🛑	NO❗
L	deposit	External ❗	🛑	NO❗
L	process	External ❗	🛑	NO❗
L	purge	External ❗	🛑	NO❗
DividendDistributor	**Implementation**	**IDividend Distributor**		
L		Public ❗	🛑	NO❗
L		External ❗	💵	NO❗
L	setDistributionCriteria	External ❗	🛑	onlyToken
L	purge	External ❗	🛑	onlyToken
L	setShare	External ❗	🛑	onlyToken
L	deposit	External ❗	🛑	onlyToken
L	process	External ❗	🛑	onlyToken

L	shouldDistribute	Internal 🔒		
L	distributeDividend	Internal 🔒	🛑	
L	claimDividend	External ❗	🛑	NO❗
L	getUnpaidEarnings	Public ❗		NO❗
L	getHolderDetails	Public ❗		NO❗
L	getCumulativeDividends	Internal 🔒		
L	getLastProcessedIndex	External ❗		NO❗
L	getNumberOfTokenHolders	External ❗		NO❗
L	getShareHoldersList	External ❗		NO❗
L	totalDistributedRewards	External ❗		NO❗
L	addShareholder	Internal 🔒	🛑	
L	removeShareholder	Internal 🔒	🛑	
Context	**Implementation**			
L	_msgSender	Internal 🔒		
L	_msgData	Internal 🔒		
Ownable	**Implementation**	**Context**		
L		Public ❗	🛑	NO❗
L	owner	Public ❗		NO❗
L	renounceOwnership	Public ❗	🛑	onlyOwner

L	transferOwnership	Public ❗	🛑	onlyOwner
L	_transferOwnership	Internal 🔒	🛑	
ETHFanBurn	**Implementation**	**IERC20**, **Ownable**		
L		Public ❗	🛑	NO❗
L		External ❗	💶	NO❗
L	totalSupply	External ❗		NO❗
L	name	Public ❗		NO❗
L	symbol	Public ❗		NO❗
L	decimals	Public ❗		NO❗
L	balanceOf	Public ❗		NO❗
L	getHolderDetails	Public ❗		NO❗
L	getLastProcessedIndex	Public ❗		NO❗
L	getNumberOfTokenHolders	Public ❗		NO❗
L	totalDistributedRewards	Public ❗		NO❗
L	allowance	External ❗		NO❗
L	approve	Public ❗	🛑	NO❗
L	_approve	Internal 🔒	🛑	
L	approveMax	External ❗	🛑	NO❗
L	transfer	External ❗	🛑	NO❗
L	transferFrom	External ❗	🛑	NO❗

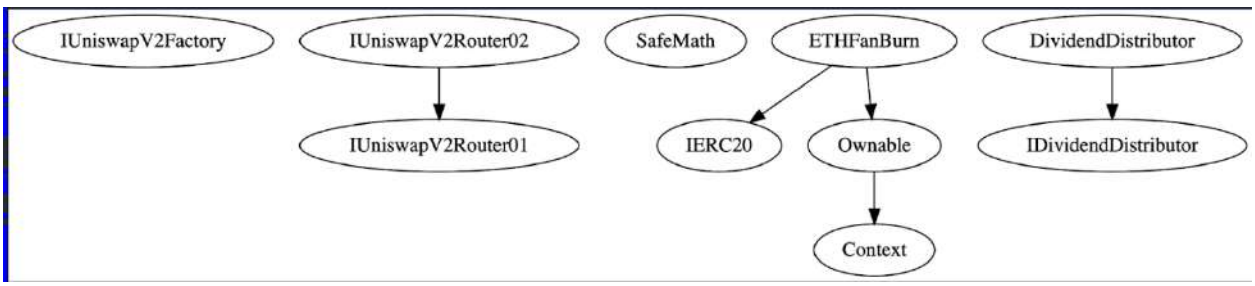
L	_transferFrom	Internal 🔒	●	
L	_basicTransfer	Internal 🔒	●	
L	shouldTakeFee	Internal 🔒		
L	takeFee	Internal 🔒	●	
L	shouldSwapBack	Internal 🔒		
L	clearStuckBalance	External ❗	●	onlyOwner
L	updateBuyFees	Public ❗	●	onlyOwner
L	updateSellFees	Public ❗	●	onlyOwner
L	updateSwapPercentages	Public ❗	●	onlyOwner
L	tradingStatus	Public ❗	●	onlyOwner
L	whitelistPreSale	Public ❗	●	onlyOwner
L	___claimRewards	Public ❗	●	NO❗
L	claimProcess	Public ❗	●	NO❗
L	swapBackInBnb	Internal 🔒	●	swapping
L	swapAndLiquify	Private 🔐	●	
L	swapTokensForEth	Private 🔐	●	
L	swapTokensForTokens	Private 🔐	●	
L	addLiquidity	Private 🔐	●	
L	setIsDividendExempt	External ❗	●	onlyOwner
L	setIsFeeExempt	External ❗	●	onlyOwner
L	setIsMaxTxExempt	External ❗	●	onlyOwner

L	setIsMaxWalletExempt	External ❗	🛑	onlyOwner
L	addAuthorizedWallets	External ❗	🛑	onlyOwner
L	setFeeReceivers	External ❗	🛑	onlyOwner
L	setStakePoolAddress	External ❗	🛑	onlyOwner
L	setMaxTxAmount	External ❗	🛑	onlyOwner
L	setMaxWalletToken	External ❗	🛑	onlyOwner
L	setSwapBackSettings	External ❗	🛑	onlyOwner
L	setDistributionCriteria	External ❗	🛑	onlyOwner
L	setDistributorSettings	External ❗	🛑	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 change all buy and sell fees without any maximum limit

```
ftrace | funcSig
function updateBuyFees(
    uint256 reward,
    uint256 marketing,
    uint256 liquidity,
    uint256 burn,
    uint256 staking
) public onlyOwner {
    buyRewardFee = reward;
    buyMarketingFee = marketing;
    buyLiquidityFee = liquidity;
    buyBurnFee = burn;
    buyStakePoolFee = staking;
    buyTotalFees = reward.add(marketing).add(liquidity).add(burn).add(
        staking
    );
}

ftrace | funcSig
function updateSellFees(
    uint256 reward,
    uint256 marketing,
    uint256 liquidity,
    uint256 burn,
    uint256 staking
) public onlyOwner {
    sellRewardFee = reward;
    sellMarketingFee = marketing;
    sellLiquidityFee = liquidity;
    sellBurnFee = burn;
    sellStakePoolFee = staking;
    sellTotalFees = reward.add(marketing).add(liquidity).add(burn).add(
        staking
    );
}
```

- The owner can enable/disable trading any time.

```
// switch Trading
ftrace | funcSig
function tradingStatus(bool _status) public onlyOwner {
    tradingOpen = _status;
}
```

❖ The owner can change max transaction amount without minimum limit

```
ftrace | funcSig
function setMaxTxAmount(uint256 amount) external onlyOwner {
    maxTxAmount = amount * (10**9);
}

ftrace | funcSig
function setMaxWalletToken(uint256 amount) external onlyOwner {
    maxWalletTokens = amount * (10**9);
}
```

Owner privileges

- The Owner can change all buy and sell fees

```
ftrace | funcSig
function updateBuyFees(
    uint256 reward,
    uint256 marketing,
    uint256 liquidity,
    uint256 burn,
    uint256 staking
) public onlyOwner {
    buyRewardFee = reward;
    buyMarketingFee = marketing;
    buyLiquidityFee = liquidity;
    buyBurnFee = burn;
    buyStakePoolFee = staking;
    buyTotalFees = reward.add(marketing).add(liquidity).add(burn).add(
        staking
    );
}

ftrace | funcSig
function updateSellFees(
    uint256 reward,
    uint256 marketing,
    uint256 liquidity,
    uint256 burn,
    uint256 staking
) public onlyOwner {
    sellRewardFee = reward;
    sellMarketingFee = marketing;
    sellLiquidityFee = liquidity;
    sellBurnFee = burn;
    sellStakePoolFee = staking;
    sellTotalFees = reward.add(marketing).add(liquidity).add(burn).add(
        staking
    );
}
```

- The owner can get BNB in contract to the owner wallet

```
ftrace | funcSig
function clearStuckBalance(uint256 amountPercentage) external onlyOwner {
    uint256 amountBNB = address(this).balance;
    payable(msg.sender).transfer((amountBNB * amountPercentage) / 100);
}
```

- The owner can change all swap percentages

```
// update swap percentages
ftrace | funcSig
function updateSwapPercentages(
    uint256 reward,
    uint256 marketing,
    uint256 liquidity,
    uint256 burn,
    uint256 staking
) public onlyOwner {
    rewardSwap = reward;
    marketingSwap = marketing;
    liquiditySwap = liquidity;
    burnSwap = burn;
    swapForStake = staking;
    totalSwap = reward.add(marketing).add(liquidity).add(burn).add(staking);
}
```

- The owner can enable/disable trading

```
// switch Trading
ftrace | funcSig
function tradingStatus(bool _status) public onlyOwner {
    tradingOpen = _status;
}
```

- The owner can whitelist pre-sale address

```
ftrace | funcSig
function whitelistPreSale(address _preSale) public onlyOwner {
    isFeeExempt[_preSale] = true;
    isDividendExempt[_preSale] = true;
    isAuthorized[_preSale] = true;
    isMaxTxExempt[_preSale] = true;
    isMaxWalletExempt[_preSale] = true;
}
```

- The owner can exclude wallets from rewards

```
ftrace | funcSig
function setIsDividendExempt(address holder, bool exempt)
    external
    onlyOwner
{
    require(holder != address(this) && holder != pair);
    isDividendExempt[holder] = exempt;
    if (exempt) {
        dividendTracker.setShare(holder, 0);
    } else {
        dividendTracker.setShare(holder, _balances[holder]);
    }
}
```

- The owner can include/exclude wallets from fee/max transactions and max wallets

```
ftrace | funcSig
function setIsFeeExempt(address holder, bool exempt) external onlyOwner {
    isFeeExempt[holder] = exempt;
}

ftrace | funcSig
function setIsMaxTxExempt(address holder, bool exempt) external onlyOwner {
    isMaxTxExempt[holder] = exempt;
}

ftrace | funcSig
function setIsMaxWalletExempt(address holder, bool exempt)
    external
    onlyOwner
{
    isMaxWalletExempt[holder] = exempt;
}
```

- The owner can add/remove authorized wallets

```
ftrace | funcSig
function addAuthorizedWallets(address holder, bool exempt)
    external
    onlyOwner
{
    isAuthorized[holder] = exempt;
}
```

- The owner can change marketing wallet and stake pool address

```
ftrace | funcSig
function setFeeReceivers(address _marketingFeeReceiver) external onlyOwner {
    marketingFeeReceiver = _marketingFeeReceiver;
}

ftrace | funcSig
function setStakePoolAddress(address _stakePool) external onlyOwner {
    stakePoolAddress = _stakePool;
}
```

- The owner can change max transaction amount and max wallet amount

```
ftrace | funcSig
function setMaxTxAmount(uint256 amount) external onlyOwner {
    maxTxAmount = amount * (10**9);
}

ftrace | funcSig
function setMaxWalletToken(uint256 amount) external onlyOwner {
    maxWalletTokens = amount * (10**9);
}
```

- The owner can enable/disable swapping and can change swap point

```
ftrace | funcSig
function setSwapBackSettings(bool _enabled, uint256 _amount)
    external
    onlyOwner
{
    swapEnabled = _enabled;
    swapThreshold = _amount;
}
```

- The owner can change minimum distribution token amount and minimum distribution period

```
ftrace | funcSig
function setDistributionCriteria(
    uint256 _minPeriod,
    uint256 _minDistribution
) external onlyOwner {
    dividendTracker.setDistributionCriteria(_minPeriod, _minDistribution);
}
```

Audit conclusion

While conducting the audit of the ETHFan Burn Token smart contract, it was observed that there is nothing alarming with the code.