Wall Street Memes 2.0 Token

RugfreeCoins Verified on October 5th, 2023

Overview

✅ No mint function found, the owner cannot mint tokens after initial deployment.

✅ The owner can't set a max transaction limit

✅ The owner can't pause trading once it's enabled

❌ The owner must enable trade for the holders, if trading remains disabled, no one would be able to buy and sell.

✅ The owner can't change fees over 20%.

✅ The owner can't blacklist wallets.

✅ The owner can't set a max wallet limit

✅ The owner can't claim the contract's balance of its own token.

❗ HIGH SEVERITY ISSUES

The owner must enable trade for the holders, if trading remains disabled, no one would be able to buy and sell.

```solidity
function enableTrading() public onlyOwner {
    require(!tradingEnabled, "Trading already enabled!");
    require(hasLiqBeenAdded, "Liquidity must be added.");
    tradingEnabled = true;
    swapEnabled = true;
    allowedPresaleExclusion = false;
}
```

Contents

Audit details

Audited project

Wall Street Memes 2.0 Token

Contract Address

0x2fe2aA3edf261B925C10D96D6383B9663497261C

Client contact

Wall Street Memes 2.0 Token Team

Blockchain

Binance Smart Chain

Project website

https://wallstreetmeme20.xyz/

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.**

Background

RugfreeCoins was commissioned by the **Wall Street Memes 2.0 Token Team** to perform an audit of the smart contract.

https://bscscan.com/address/0x2fe2aa3edf261b925c10d96d6383b9663497261c

This audit focuses on verifying 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, and long-term sustainability, and as a guide to improving the smart contract's security posture by remediating the identified issues.

Tokenomics

▲ 5% tax when buying & selling

5% of trade goes to the Dev wallet in BNB

Target market and the concept

- 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 in the Wall Street Memes 2.0 token ecosystem.
- Anyone who's interested in taking part in the future plans of Wall Street Memes 2.0 Token.
- Anyone who's interested in making financial transactions with any other party using Wall Street Memes 2.0 Token as the currency.

Potential to grow with score points

⚡ Project efficiency	**8** / 10
🌟 Project uniqueness	**7** / 10
📊 Information quality	**8** / 10
💧 Service quality	**8** / 10
💻 System quality	**8** / 10
🌍 Impact on the community	**8** / 10
💼 Impact on the business	**9** / 10
🔮 Preparing for the future	**8** / 10
🔒 Smart contract security	**9** / 10
🛠️ Smart contract functionality assessment	**8** / 10
🏆 **Total Score**	**8.1**/ **10**

Contract details

Token contract details for 5th of October 2023

Contract name	**Wall Street Memes 2.0**
Contract address	**0x2fe2aA3edf261B925C10D96D6383B9663497261C**
Token supply	**2,000,000,000**
Token ticker	**WSM 2.0**
Decimals	**18**
Token holders	**1**
Transaction count	**1**
Contract deployer address	**0xD904f646d5B94dbdCb89ED105C0B675F72D93Ea9**
Contract's current owner address	**0xD904f646d5B94dbdCb89ED105C0B675F72D93Ea9**

Contract code function details

№	Category	Item	Result
1	Coding conventions	ERC20 Token standards	PASS
		Compile errors	PASS
		Compiler version security	PASS
		Visibility specifiers	PASS
		Gas consumption	LOW
		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
		Self destruct function security	PASS
3	Business security & centralisation	Access control of owners	HIGH
		Business logics	PASS
		Business implementation	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

The 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 their visibility and mutability.

Contract	Type	Bases		
L	Function Name	Visibility	Mutability	Modifiers
SafeMath	**Library**			
L	add	Internal 🔒		
L	sub	Internal 🔒		
L	sub	Internal 🔒		
L	mul	Internal 🔒		
L	div	Internal 🔒		
L	div	Internal 🔒		
IERC20	**Interface**			
L	totalSupply	External ❗		NO ❗
L	decimals	External ❗		NO ❗
L	symbol	External ❗		NO ❗
L	name	External ❗		NO ❗
L	getOwner	External ❗		NO ❗
L	balanceOf	External ❗		NO ❗

L	transfer	External ❗	🛑	NO ❗
L	allowance	External ❗		NO ❗
L	approve	External ❗	🛑	NO ❗
L	transferFrom	External ❗	🛑	NO ❗
IFactoryV2	**Interface**			
L	getPair	External ❗		NO ❗
L	createPair	External ❗	🛑	NO ❗
IV2Pair	**Interface**			
L	factory	External ❗		NO ❗
L	getReserves	External ❗		NO ❗
L	sync	External ❗	🛑	NO ❗
IRouter01	**Interface**			
L	factory	External ❗		NO ❗
L	WETH	External ❗		NO ❗
L	addLiquidityETH	External ❗	💵	NO ❗
L	addLiquidity	External ❗	🛑	NO ❗
L	swapExactETHForTokens	External ❗	💵	NO ❗
L	getAmountsOut	External ❗		NO ❗

L	getAmountsIn	External ❗		NO ❗
IRouter02	**Interface**	**IRouter01**		
L	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
L	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗	💵	NO ❗
L	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
L	swapExactTokensForTokens	External ❗	🛑	NO ❗
Wallstreet meme20Token	**Implementation**	**IERC20**		
L		Public ❗	💵	NO ❗
L		External ❗	💵	NO ❗
L	totalSupply	External ❗		NO ❗
L	decimals	External ❗		NO ❗
L	symbol	External ❗		NO ❗
L	name	External ❗		NO ❗
L	getOwner	External ❗		NO ❗
L	allowance	Public ❗		NO ❗
L	balanceOf	Public ❗		NO ❗
L	transfer	Public ❗	🛑	NO ❗
L	approve	External ❗	🛑	NO ❗
L	_approve	Internal 🔒	🛑	
L	approveContractContingency	External ❗	🛑	onlyOwner

14

L	transferFrom	External ❗	🛑	NO ❗
L	_hasLimits	Internal 🔒		
L	_transfer	Internal 🔒	🛑	
L	finalizeTransfer	Internal 🔒	🛑	
L	_transferAmount	Internal 🔒	🛑	
L	_checkLiquidityAdd	Internal 🔒	🛑	
L	swapTokensForEth	Private 🔐	🛑	lockThe Swap
L	sendETHToFee	Private 🔐	🛑	
L	manualswap	External ❗	🛑	NO ❗
L	manualsend	External ❗	🛑	NO ❗
L	transferOwner	External ❗	🛑	onlyOwner
L	renounceOwnership	External ❗	🛑	onlyOwner
L	excludePresaleAddresses	External ❗	🛑	onlyOwner
L	setDevAddress	External ❗	🛑	onlyOwner
L	excludeMultipleAccountsFromFees	Public ❗	🛑	onlyOwner
L	enableTrading	Public ❗	🛑	onlyOwner
L	setFee	Public ❗	🛑	onlyOwner
L	updateSwapEnabled	External ❗	🛑	onlyOwner

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

Inheritance Hierarchy
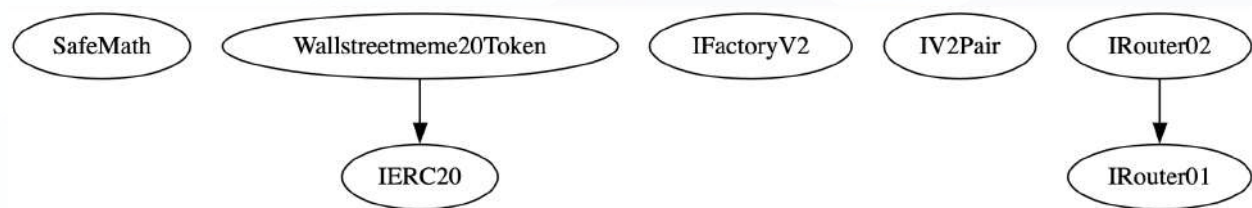
Security issue checking status

❖ High severity issues

The owner must enable trade for the holders, if trading remains disabled, no one would be able to buy and sell.

```
function enableTrading() public onlyOwner {
    require(!tradingEnabled, "Trading already enabled!");
    require(hasLiqBeenAdded, "Liquidity must be added.");
    tradingEnabled = true;
    swapEnabled = true;
    allowedPresaleExclusion = false;
}
```

❖ Medium severity issues

No medium severity issues found

❖ Low severity issues

It verifies the sender's balance after the swap, and if the user does not possess the necessary balance, the transaction will fail, leading to a loss of gas. Hence, it is advisable to conduct the balance check at the outset of the function.

```
uint256 fromBalance = _balances[from];
require(
    fromBalance >= amount,
    "ERC20: transfer amount exceeds balance"
);
```

When excluding multiple wallets from fees, the owner can input any number of wallets simultaneously. However, it's important to note that if the owner enters a large array of wallets, the transaction may fail due to the block gas limit.

```solidity
function excludeMultipleAccountsFromFees(
    address[] calldata _accounts,
    bool _excluded
) public onlyOwner {
    for (uint256 i = 0; i < _accounts.length; i++) {
        _isExcludedFromFee[_accounts[i]] = _excluded;
    }
}
```

Owner privileges

❖ Owner and dev wallet can manually swap and send BNB to dev wallet

```
function manualswap() external {
    require(msg.sender == devAddress || msg.sender == _owner);
    uint256 contractBalance = balanceOf(address(this));
    swapTokensForEth(contractBalance);
}

function manualsend() external {
    require(msg.sender == devAddress || msg.sender == _owner);
    uint256 contractETHBalance = address(this).balance;
    sendETHToFee(contractETHBalance);
}
```

❖ Owner can enable/disable swapping

```
function updateSwapEnabled(bool _enabled) external onlyOwner {
    swapEnabled = _enabled;
}
```

❖ Owner can transfer and renounce the ownership

```solidity
function transferOwner(address _newOwner) external onlyOwner {
    require(
        _newOwner != address(0),
        "Call renounceOwnership to transfer owner to the zero address."
    );
    require(
        _newOwner != DEAD,
        "Call renounceOwnership to transfer owner to the zero address."
    );
    if (balanceOf(_owner) > 0) {
        finalizeTransfer(_owner, _newOwner, balanceOf(_owner));
    }

    address oldOwner = _owner;
    _owner = _newOwner;
    _isExcludedFromFee[_owner] = true;

    emit OwnershipTransferred(oldOwner, _newOwner);
}

function renounceOwnership() external onlyOwner {
    address oldOwner = _owner;
    _owner = address(0);
    emit OwnershipTransferred(oldOwner, address(0));
}
```

❖ Owner can change dev wallet address

```solidity
function setDevAddress(address _newAddress) external onlyOwner {
    require(devAddress != address(0), "address cannot be 0");
    devAddress = payable(_newAddress);
}
```

❖ Owner can exclude pre-sale address from contract before enabling trading

```solidity
function excludePresaleAddresses(
    address _router,
    address _presale
) external onlyOwner {
    require(allowedPresaleExclusion);
    require(
        _router != address(this) &&
            _presale != address(this) &&
            lpPair != _router &&
            lpPair != _presale,
        "Just don't."
    );
    if (_router == _presale) {
        _liquidityHolders[_presale] = true;
    } else {
        _liquidityHolders[_router] = true;
        _liquidityHolders[_presale] = true;
    }
}
```

❖ Owner can include/exclude multiple wallets from fees

```solidity
function excludeMultipleAccountsFromFees(
    address[] calldata _accounts,
    bool _excluded
) public onlyOwner {
    for (uint256 i = 0; i < _accounts.length; i++) {
        _isExcludedFromFee[_accounts[i]] = _excluded;
    }
}
```

❖ Owner can enable trading, once enabled can not disable again

```solidity
function enableTrading() public onlyOwner {
    require(!tradingEnabled, "Trading already enabled!");
    require(hasLiqBeenAdded, "Liquidity must be added.");
    tradingEnabled = true;
    swapEnabled = true;
    allowedPresaleExclusion = false;
}
```

❖ Owner can change buy and sell fees maximum up-to 20%

```solidity
function setFee(uint256 _taxFeeOnBuy, uint256 _taxFeeOnSell) public onlyOwner {
    require(_taxFeeOnBuy <= 100, "Tax cannot be more than 10.");
    require(_taxFeeOnSell <= 100, "Tax cannot be more than 10.");
    taxFeeOnBuy = _taxFeeOnBuy;
    taxFeeOnSell = _taxFeeOnSell;
}
```

Audit conclusion

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

Smart contract functional Status:	PASS ▾
Smart contract security Status:	HIGH ISSUES ▾
Number of risk issues:	3
Solidity code functional issue level:	PASS ▾
Number of owner privileges:	8
Centralization risk correlated to the active owner:	HIGH ▾
Smart contract active ownership:	ACTIVE ▾