



SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT



Black Water Tiger
\$BWT



13/01/2022



TABLE OF CONTENTS

- 1 DISCLAIMER
- 2 INTRODUCTION
- 3-4 WEBSITE DIAGNOSTIC
- 5-6 AUDIT OVERVIEW
- 7-8 OWNER PRIVILEGES
- 9 CONCLUSION AND ANALYSIS
- 10 TOKEN DETAILS
- 11 BLACK WATER TIGER TOKEN DISTRIBUTION & TOP 10 TOKEN HOLDERS
- 12 TECHNICAL DISCLAIMER



DISCLAIMER

The information provided on this analysis document is only for general information and should not be used as a reason to invest.

FreshCoins Team will take no payment for manipulating the results of this audit.

The score and the result will stay on this project page information on our website <https://freshcoins.io>

FreshCoins Team does not guarantees that a project will not sell off team supply, or any other scam strategy (RUG or Honeypot etc)



INTRODUCTION

FreshCoins (Consultant) was contracted by Black Water Tiger (Customer) to conduct a Smart Contract Code Review and Security Analysis.

0x939054d37655974B363954C0FDDEe6104c4436a5

Network: Binance Smart Chain (BSC)

This report presents the findings of the security assessment of Customer's smart contract and its code review conducted on 13/01/2022



WEBSITE DIAGNOSTIC

<https://blackwatertigercoin.com/>



0-49



50-89



90-100



Performance



Accessibility



Best Practices



SEO



Progressive
Web App

Metrics



First Contentful Paint

2.2 s



Time to interactive

5.0 s



Speed Index

3.6 s



Total Blocking Time

90 ms



Large Contentful Paint

5.2 s



Cumulative Layout Shift

0

WEBSITE IMPROVEMENTS

Eliminate render-blocking resources

Reduce unused CSS

Reduce unused JavaScript

Ensure text remains visible during webfont load

Document doesn't have a <title> element

Reduce initial server response time

AUDIT OVERVIEW



Security Score



Static Scan
Automatic scanning for common vulnerabilities



ERC Scan
Automatic checks for ERC's conformance

0 **High**

0 **Medium**

0 **Low**

0 **Optimizations**

0 **Informational**



| No. | Issue description | Checking Status |
|-----|--------------------------------|-----------------|
| 1 | Compiler Errors / Warnings | Passed |
| 2 | Reentrancy and Cross-function | Passed |
| 3 | Front running | Passed |
| 4 | Timestamp dependence | Passed |
| 5 | Integer Overflow and Underflow | Passed |
| 6 | Reverted DoS | Passed |
| 7 | DoS with block gas limit | Passed |
| 8 | Methods execution permissions | Passed |
| 9 | Exchange rate impact | Passed |
| 10 | Malicious Event | Passed |
| 11 | Scoping and Declarations | Passed |
| 12 | Uninitialized storage pointers | Passed |
| 13 | Design Logic | Passed |
| 14 | Safe Zeppelin module | Passed |

OWNER PRIVILEGES

Contract owner can't mint tokens after initial contract deploy.

Contract owner can't exclude an address from transactions.

Contract owner can exclude/include wallet/multiple wallets from fee

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

    emit ExcludeFromFees(account, excluded);
}

.

.

.

function excludeMultipleAccountsFromFees(
    address[] calldata accounts,
    bool excluded
) public onlyOwner {
    for (uint256 i = 0; i < accounts.length; i++) {
        _isExcludedFromFees[accounts[i]] = excluded;
    }

    emit ExcludeMultipleAccountsFromFees(accounts, excluded);
}
```

Contract owner can renounce ownership

```
function renounceOwnership() public virtual onlyOwner {
    _setOwner(address(0));
}
```

Contract owner can transfer ownership

```
function transferOwnership(address newOwner) public virtual onlyOwner {
    require(newOwner != address(0), "Ownable: new owner is the zero address");
    _setOwner(newOwner);
}
```

Contract owner can change max tx amount

```
function setSwapTokensAtAmount(uint256 amount) external onlyOwner {
    swapTokensAtAmount = amount;
}
```

Contract owner can change the fees

```
function setTokenRewardsFee(uint256 value) external onlyOwner {
    tokenRewardsFee = value;
    totalFees = tokenRewardsFee.add(liquidityFee).add(marketingFee);
    require(totalFees <= 25, "Total fee is over 25%");
}

function setLiquidityFee(uint256 value) external onlyOwner {
    liquidityFee = value;
    totalFees = tokenRewardsFee.add(liquidityFee).add(marketingFee);
    require(totalFees <= 25, "Total fee is over 25%");
}

function setMarketingFee(uint256 value) external onlyOwner {
    marketingFee = value;
    totalFees = tokenRewardsFee.add(liquidityFee).add(marketingFee);
    require(totalFees <= 25, "Total fee is over 25%");
}
```

Contract owner can change marketing wallet address

```
function setMarketingWallet(address payable wallet) external onlyOwner {
    _marketingWalletAddress = wallet;
}
```

CONCLUSION AND ANALYSIS



Smart Contracts within the scope were manually reviewed and analyzed with static tools.



Audit report overview contains all found security vulnerabilities and other issues in the reviewed code.



Found no issue during the first review.

TOKEN DETAILS

Details

Buy fees: 13%

Sell fees: 13%

Max TX: N/A

Max Sell: N/A

Honeypot Risk

Ownership: Owned

Blacklist: Not detected

Modify Max TX: Detected

Modify Max Sell: Not detected

Disable Trading: Not detected

Rug Pull Risk

Liquidity: N/A

Holders: Clean



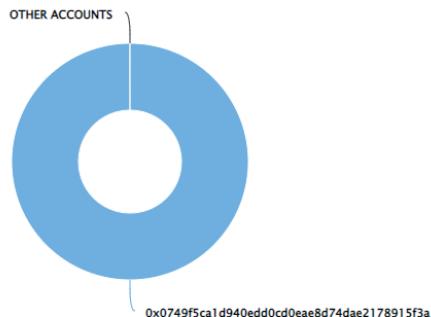
BLACK WATER TIGER TOKEN DISTRIBUTION & TOP 10 TOKEN HOLDERS

The top 10 holders collectively own 100.00% (202,220,222,022,202.00 Tokens) of Black Water Tiger

Token Total Supply: 202,220,222,022,202.00 Token | Total Token Holders: 1

Black Water Tiger Top 10 Token Holders

Source: BscScan.com



(A total of 202,220,222,022,202.00 tokens held by the top 10 accounts from the total supply of 202,220,222,022,202.00 token)

| Rank | Address | Quantity (Token) | Percentage |
|------|--|---------------------|------------|
| 1 | 0x0749f5ca1d940edd0cd0eae8d74dae2178915f3a | 202,220,222,022,202 | 100.0000% |

TECHNICAL DISCLAIMER

Smart contracts are deployed and executed on the blockchain platform. The platform, its programming language, and other software related to the smart contract can have its vulnerabilities that can lead to hacks. The audit can't guarantee the explicit security of the audited project / smart contract.

