freshcoins

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

XProfit

$XProfit

08/04/2024

TOKEN OVERVIEW

Fees

- Buy fees: 7%
- Sell fees: 7%

Fees privileges

- Can change fees up to 25%

Ownership

- Owned

Minting

- No mint function

Max Tx Amount / Max Wallet Amount

- Can’t change max tx amount and / or max wallet amount

Blacklist

- Blacklist function not detected

Other privileges

- Can exclude / include from fees
- The contract has several points of centralization and external dependencies that pose risks

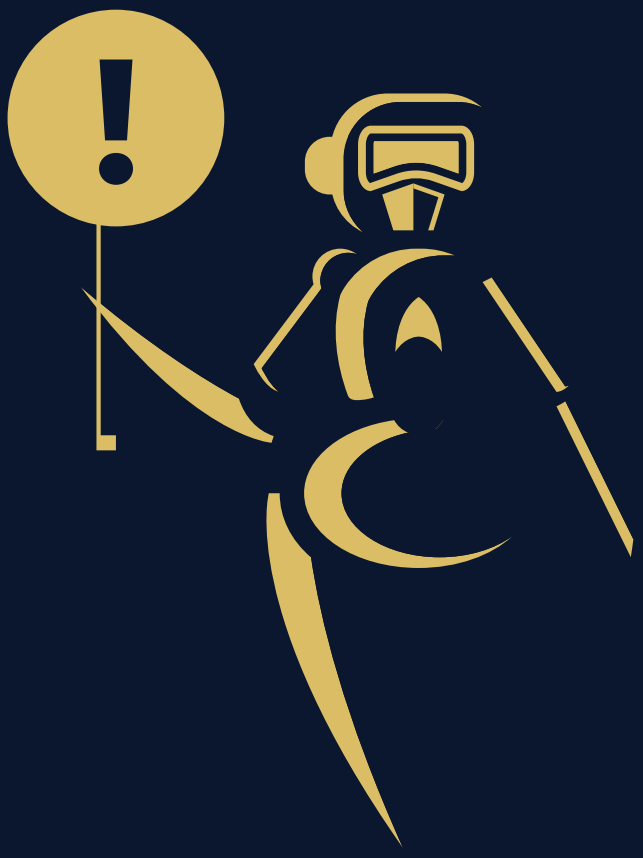
TABLE OF CONTENTS

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 XProfit (Customer) to conduct a Smart Contract Code Review and Security Analysis.

0x6f71052CB28864140Fa7A48F9dd50aDd8794e9cd

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 08/04/2024

WEBSITE DIAGNOSTIC

https://xprofit.app/

0-49 50-89 90-100

Performance Accessibility Best Practices SEO Progressive Web App

Socials

Twitter

https://twitter.com/realxprofitdev

Telegram

https://t.me/realxprofit

AUDIT OVERVIEW

65

Security Score
HIGH RISK
Audit FAIL

93

Static Scan
Automatic scanning for common vulnerabilities

62

ERC Scan
Automatic checks for ERC's conformance

3 High

1 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	Low
4	Timestamp dependence	Passed
5	Integer Overflow and Underflow	Passed
6	Reverted DoS	Passed
7	DoS with block gas limit	Low
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 addresses from transactions

- Contract owner can change _marketCapMultiplier value (without threshold)

Setting the marketCapMultiplier to an extremely high value, it would significantly inflate the target market capitalization (targetMarketCap). This could lead to unrealistic market cap goals, potentially distorting market dynamics and investor expectations. Additionally, it might introduce instability and volatility into the token's price, as achieving such a high market cap could be practically unattainable and unsustainable in the long term

Current values:
uint256 public marketCapMultiplier = 10;
uint256 public marketCapIndex = 1;

```
function setMarketCapMultiplier(uint256 _marketCapMultiplier, uint256 _marketCapIndex, uint256 setMar-
ketCap) external onlyOwner {
    marketCapMultiplier = _marketCapMultiplier;
    marketCapIndex = _marketCapIndex;
    if (setMarketCap == 1) {
        targetMarketCap = ((initialMarketCap * (marketCapIndex * marketCapMultiplier)) * 10) / 10;
    }
}
```

- Contract owner can change multiplier value (without threshold)

Setting the multiplier to an extremely high value, it would significantly amplify the calculation of presale values during token transfers. This could lead to unrealistic and unsustainable levels of token distribution during presales, potentially causing inflationary pressures and devaluing the token over time. Additionally, such a high multiplier may disrupt the intended balance between token rewards, project sustainability, and investor returns, posing significant risks to the project's viability and long-term success

Current values:
uint256 public multiplier = 10752;

```
function setInitFees(uint256 _multiplier) external onlyOwner {
    multiplier = _multiplier;
}
```

Contract owner can change trading timer values (without threshold)

Setting the fullTradingOffTimer or claimOffTimer to an extremely high value, it would result in excessively long periods for full trading to be disabled or for claiming rewards to be turned off. This could disrupt the project's intended market dynamics and reward distribution mechanisms, potentially leading to unfair advantages for certain participants and hampering overall project sustainability.

Current values:

```
uint256 public fullTradingOffTimer = 48 hours;
uint256 public claimOffTimer = 24 hours;
```

```
function setTradingTimer(uint256 _fullTradingOffTimer, uint256 _claimOffTimer) external onlyOwner {
    fullTradingOffTimer = _fullTradingOffTimer;
    claimOffTimer = _claimOffTimer;
}
```

Contract owner can change fees up to 25%

Current values:

```
uint256 public feeDenominator = 10000;
```

```
function setFees(uint256 _projectFee, uint256 _rewardFee) external onlyOwner {
    rewardFee = _rewardFee;
    projectFee = _projectFee;
    totalFee = _projectFee + _rewardFee;
    require(totalFee < (feeDenominator/4), "Fee > 25%");
}
```

Contract owner can exclude/include wallet from tax

```
function setIsFeeExempt(address holder, uint256 exempt) external onlyOwner {
    isFeeExempt[holder] = exempt;
}
```

Contract owner can change projectFeeReceiver address

Current values:

projectFeeReceiver : 0x1CA34FD6a444c58ccdAacF3B7c28F9905A71263c

```
function setProjectFeeReceiver(address _projectFeeReceiver) external onlyOwner {
    projectFeeReceiver = _projectFeeReceiver;
}
```

Contract owner can change swap settings (without threshold)

```
function setSwapAgainstPair(uint256 _swapAgainstPair) external onlyOwner {
    swapAgainstPair = _swapAgainstPair;
}

function setSwapPercent(uint256 _swapThresholdPercent, uint256 _swapMaxPercent) external onlyOwner {
    swapThreshold = TOTAL_SUPPLY / _swapThresholdPercent;
    swapThresholdPercent = _swapThresholdPercent;
    swapMax = TOTAL_SUPPLY / _swapMaxPercent;
    swapMaxPercent = _swapMaxPercent;
}
```

Contract owner can transfer ownership

```
function transferOwnership(address payable adr) public onlyOwner {
    owner = adr;
    emit OwnershipTransferred(adr);
}
```

A lock is implemented on sales that exceed an investor's initial investment until the current target market cap is reached. During this lock period, investors are only permitted to sell XProfit equivalent to their initial investment. To enforce this restriction, the smart contract tracks the total amount invested by each wallet and only allows the sale of tokens valued at or below that amount. For instance, if a wallet initially invested $100 in XProfit, any attempt to sell tokens exceeding $100 during the lock period will be rejected. Once XProfit reaches its current target market cap, the contract will disable the lock through Chain Link automation. The lock remains disabled for 24 hours, during which any amount of XProfit can be sold without restrictions. After this period, the lock is reinstated until the next milestone is reached.

Diamond holders can also claim their rewards during the lock period. The cycle repeats, with each milestone resulting in a higher market cap than the previous one. For instance, at launch, the target market cap will be 10 times the initial market cap. Subsequently, the next milestones will be set at 20 times, and so forth. The target market caps will follow a pattern such as 10, 20...100, 100, 200... 1000, 1000, 2000... 10000, and so on, reflecting an incremental increase with each milestone.

In summary, the contract has several points of centralization and external dependencies that pose risks

Recommendation:

The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. The risk can be prevented by temporarily locking the contract or renouncing ownership.

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 3 HIGH issues during the first review.

TOKEN DETAILS

Details

Buy fees:	7%
Sell fees:	7%
Max TX:	N/A
Max Wallet:	N/A

Honeypot Risk

Ownership:	Owned
Blacklist:	Not detected
Modify Max TX:	Not detected
Modify Max Sell:	Not detected
Disable Trading:	Not detected

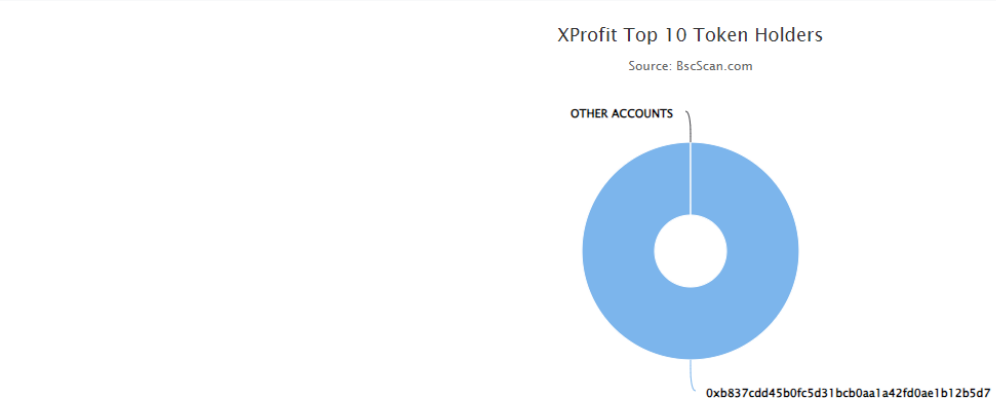
Rug Pull Risk

Liquidity:	N/A
Holders:	100% unlocked tokens

XPROFIT TOKEN ANALYTICS & TOP 10 TOKEN HOLDERS

The top 10 holders collectively own 100.00% (21,000,000,000.00 Tokens) of XProfit

Token Total Supply: 21,000,000,000.00 Token | Total Token Holders: 1



(A total of 21,000,000,000.00 tokens held by the top 10 accounts from the total supply of 21,000,000,000.00 token)

Rank	Address	Quantity (Token)	Percentage
1	0xB837cdD4...e1B12b5d7	21,000,000,000	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.