freshcoins

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

NEXTEP

$NEXTEP

31/01/2022

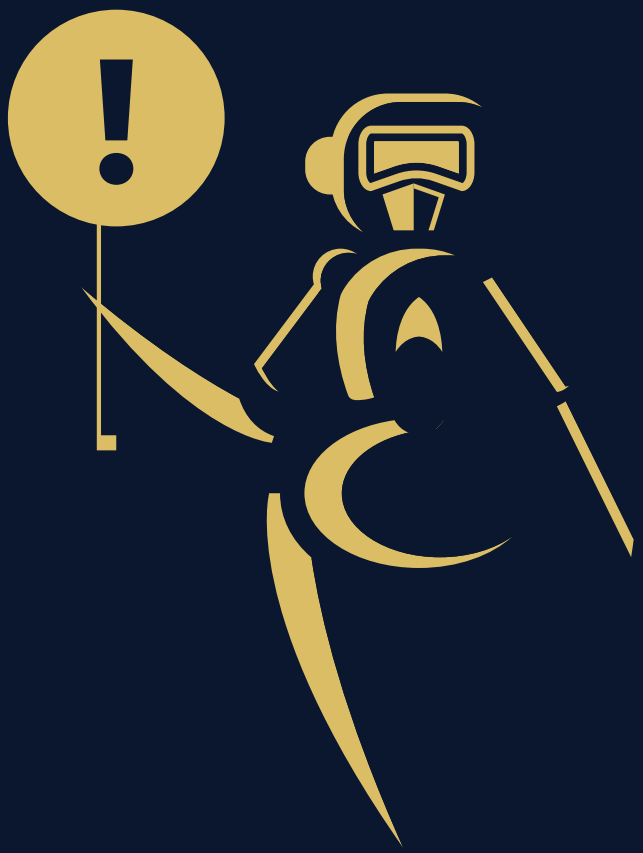
TABLE OF CONTENTS

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

0xF10770649b0b8f62BB5E87ad0da7729888A7F5C3

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 31/01/2022

AUDIT OVERVIEW

94

Security Score

92

Static Scan
Automatic scanning for common vulnerabilities

96

ERC Scan
Automatic checks for ERC's conformance

0 High

0 Medium

1 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	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.

Owner must be a contract with transparent rules for using openTrade, stopTrade functions.

Contract owner can exclude/include wallet from fee

```
function excludeFromFee(address account) public onlyOwner {
    _isExcludedFromFee[account] = true;
}

function includeInFee(address account) public onlyOwner {
    _isExcludedFromFee[account] = false;
}
```

Contract owner can change max tx amount and exclude wallet for tx amount limitations

```
function setMaxTxAmount(uint256 maxTxAmount) external onlyOwner() {
    _maxTxAmount = maxTxAmount;
}
.
.
.
function setExcludeFromMaxTxLimit(address account, bool _enabled) public onlyOwner {
    _isExcludedFromMaxTxLimit[account] = _enabled;
}
```

Contract owner is able to blacklist wallet(s)

```
function blacklistAddress(address account, bool value) external onlyOwner {
    _isBlacklisted[account] = value;
}
```

Contract owner can change sell fees

```
function setAllSaleFeesPercents(uint256 liquidityFee, uint256 marketingFee, uint256 developerFee, uint256
burnFee)  external onlyOwner {
    _saleLiquidityFee = liquidityFee;
    _saleMarketingFee = marketingFee;
    _saleDeveloperFee = developerFee;
    _saleBurnFee = burnFee;
}
```

Contract owner can renounce ownership

```
function renounceOwnership() public virtual onlyOwner {
    emit OwnershipTransferred(_owner, address(0));
    _owner = 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");
    emit OwnershipTransferred(_owner, newOwner);
    _owner = newOwner;
}
```

Contract owner can change the fees

```
function setLiquidityFeePercent(uint256 liquidityFee) external onlyOwner() {
    _liquidityFee = liquidityFee;
    _previousLiquidityFee = liquidityFee;
}

function setMarketingFeePercent(uint256 marketingFee) external onlyOwner() {
    _marketingFee = marketingFee;
    _previousMarketingFee = marketingFee;
}

function setDeveloperFeePercent(uint256 developerFee) external onlyOwner() {
    _developerFee = developerFee;
    _previousDeveloperFee = developerFee;
}

function setBurnFeePercent(uint256 burnFee) external onlyOwner() {
    _burnFee = burnFee;
    _previousBurnFee = burnFee;
}
```

Contract owner can change swap settings

```
function setSwapAndLiquifyEnabled(bool _enabled) public onlyOwner {
    swapAndLiquifyEnabled = _enabled;
    emit SwapAndLiquifyEnabledUpdated(_enabled);
}
```

Recomandation:

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 1 LOW issue during the first review.

TOKEN DETAILS

Details

Buy fees:	20%
Sell fees:	20%
Max TX:	100000000000
Max Sell:	N/A

Honeypot Risk

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

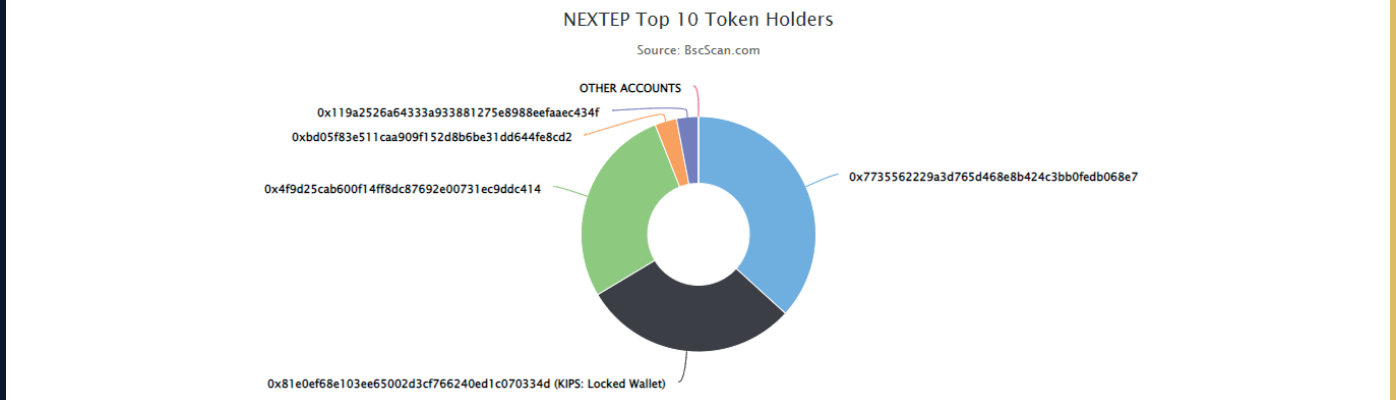
Rug Pull Risk

Liquidity:	N/A
Holders:	Clean

NEXTEP TOKEN DISTRIBUTION & TOP 10 TOKEN HOLDERS

The top 10 holders collectively own 100.00% (100,000,000,000.00 Tokens) of NEXTEP

Token Total Supply: 100,000,000,000.00 Token | Total Token Holders: 5



(A total of 100,000,000,000.00 tokens held by the top 10 accounts from the total supply of 100,000,000,000.00 token)

Rank	Address	Quantity (Token)	Percentage
1	0x7735562229a3d765d468e8b424c3bb0fedb068e7	36,740,426,000.000004	36.7404%
2	KIPS: Locked Wallet	29,700,973,999.999996	29.7010%
3	0x4f9d25cab600f14ff8dc87692e00731ec9ddc414	27,558,600,000	27.5586%
4	0xbd05f83e511caa909f152d8b6be31dd644fe8cd2	3,000,000,000	3.0000%
5	0x119a2526a64333a933881275e8988eefaaec434f	3,000,000,000	3.0000%

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.