freshcoins

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

GOLDENPIG

$PIG

24/10/2023

TOKEN OVERVIEW

Fees

• Buy fees:	3%
• Sell fees:	6%

Fees privileges

• Can change buy fees up to 4%, sell fees up to 6% and transfer fees up to 1%

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

• Contract owner has to call enableTrading function to enable trade

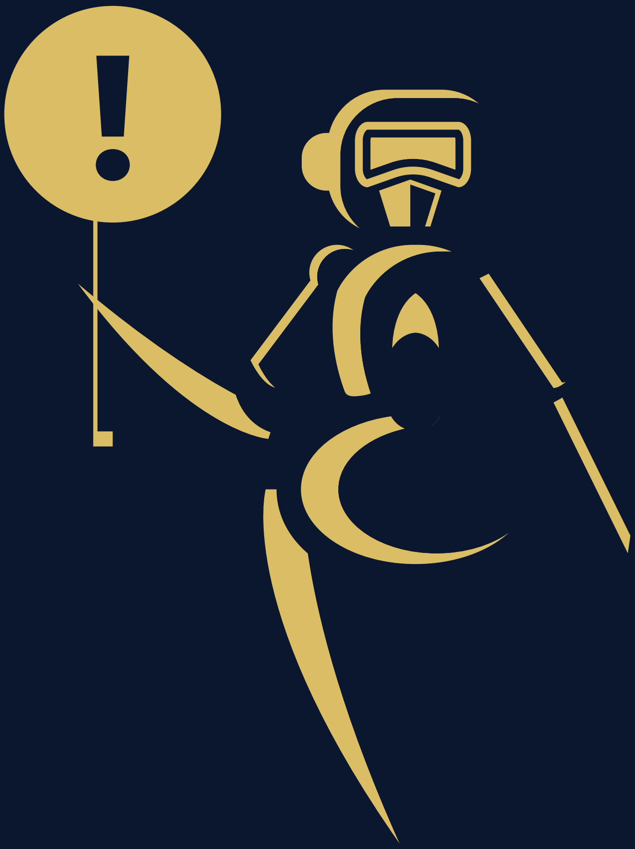
TABLE OF CONTENTS

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

0x7905bb8B4D46B74D02D3F17E5C283555DBea546D

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 24/10/2023

WEBSITE DIAGNOSTIC

https://goldenpig.site/

🟥	🟨	🟩
0-49	50-89	90-100

93	94	91	90	NA
Performance	Accessibility	Best Practices	SEO	Progressive Web App

Socials

Twitter

https://twitter.com/goldenpig_bsc

Telegram

https://t.me/goldenpig_bsc

AUDIT OVERVIEW

84

Security Score

98

Static Scan
Automatic scanning for common vulnerabilities

82

ERC Scan
Automatic checks for ERC's conformance

1	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	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 an address from transactions

● Contract owner can exclude/include wallet from tax

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

      emit ExcludeFromFees(account, excluded);
}
```

● Contract owner has to call enableTrading function to enable trade

Please note that any wallet excluded from fees retains the ability to engage in trading, even in situations where trading has been disabled

```
function enableTrading() external onlyOwner{
      require(!tradingEnabled, "Trading already enabled.");
      tradingEnabled = true;
      swapEnabled = true;
}

_transferFrom function line 703
.
.
.
require(tradingEnabled || _isExcludedFromFees[from] || _isExcludedFromFees[to], "Trading not yet enabled!");
.
.
.
```

● Contract owner can change swap settings

```
function setSwapEnabled(bool _enabled) external onlyOwner{
      require(swapEnabled != _enabled, "swapEnabled already at this state.");
      swapEnabled = _enabled;
}

function setSwapTokensAtAmount(uint256 newAmount) external onlyOwner{
      require(newAmount > totalSupply() / 1_000_000, "SwapTokensAtAmount must be greater than 0.0001%
of total supply");
      swapTokensAtAmount = newAmount;

      emit SwapTokensAtAmountUpdated(swapTokensAtAmount);
}
```

● Contract owner has ability to retrieve any token held by the contract

Native tokens excluded

```
function claimStuckTokens(address token) external onlyOwner {
      require(token != address(this), "Owner cannot claim contract's balance of its own tokens");
      if (token == address(0x0)) {
         payable(msg.sender).sendValue(address(this).balance);
         return;
      }
      IERC20 ERC20token = IERC20(token);
      uint256 balance = ERC20token.balanceOf(address(this));
      ERC20token.transfer(msg.sender, balance);
}
```

● Contract owner can change marketingWallet address

Current value:

marketingWallet: 0x5B1De8B9c763367FEb253EaaaB275EBfA40D255B

```
function changeMarketingWallet(address _marketingWallet) external onlyOwner{
      require(_marketingWallet != marketingWallet,"Marketing wallet is already that address");
      require(_marketingWallet != address(0),"Marketing wallet cannot be the zero address");
      marketingWallet = _marketingWallet;

      emit MarketingWalletChanged(marketingWallet);
}
```

● Contract owner can change buy fees up to 4%, sell fees up to 6% and transfer fees up to 1%

```
function updateBuyFees(uint256 _liquidityFeeOnBuy, uint256 _marketingFeeOnBuy) external onlyOwner {
      liquidityFeeOnBuy = _liquidityFeeOnBuy;
      marketingFeeOnBuy = _marketingFeeOnBuy;
      _totalFeesOnBuy   = liquidityFeeOnBuy + marketingFeeOnBuy;
      require(_totalFeesOnBuy <= 4, "Total Fees cannot exceed the maximum");

      emit UpdateBuyFees(liquidityFeeOnBuy, marketingFeeOnBuy);
}

function updateSellFees(uint256 _liquidityFeeOnSell, uint256 _marketingFeeOnSell) external onlyOwner {
      liquidityFeeOnSell = _liquidityFeeOnSell;
      marketingFeeOnSell = _marketingFeeOnSell;
      _totalFeesOnSell   = liquidityFeeOnSell + marketingFeeOnSell;
      require(_totalFeesOnSell <= 6, "Total Fees cannot exceed the maximum");

      emit UpdateSellFees(liquidityFeeOnSell, marketingFeeOnSell);
}

function updateWalletToWalletTransferFee(uint256 _walletToWalletTransferFee) external onlyOwner {
      require(_walletToWalletTransferFee <= 1, "Wallet to Wallet Transfer Fee cannot exceed the maximum");
      walletToWalletTransferFee = _walletToWalletTransferFee;

      emit UpdateWalletToWalletTransferFee(walletToWalletTransferFee);
}
```

● 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;
}
```

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 1 HIGH issues during the first review.

TOKEN DETAILS

Details

Buy fees:	3%
Sell fees:	6%
Max TX:	N/A
Max Sell:	N/A

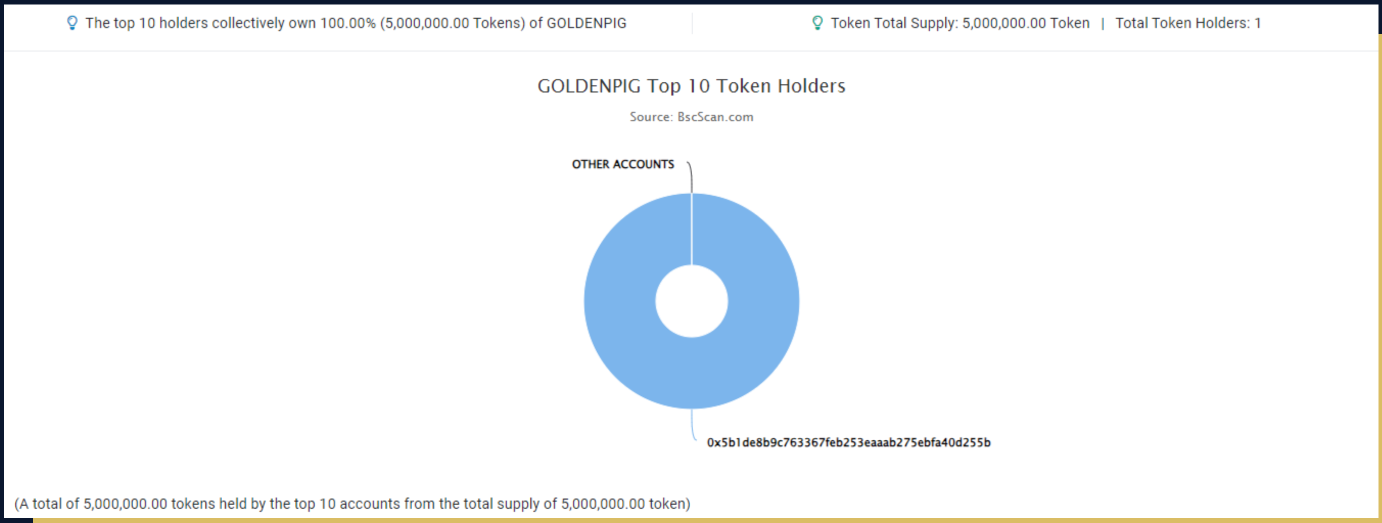
Honeypot Risk

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

PIG TOKEN ANALYTICS & TOP 10 TOKEN HOLDERS



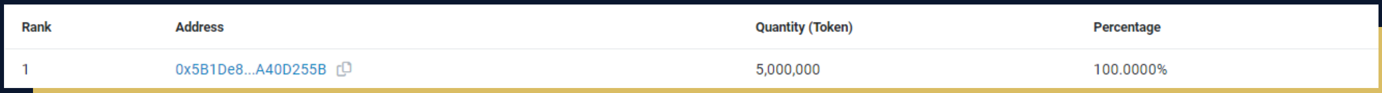
The top 10 holders collectively own 100.00% (5,000,000.00 Tokens) of GOLDENPIG

Token Total Supply: 5,000,000.00 Token | Total Token Holders: 1

GOLDENPIG Top 10 Token Holders

Source: BscScan.com

(A total of 5,000,000.00 tokens held by the top 10 accounts from the total supply of 5,000,000.00 token)



Rank	Address	Quantity (Token)	Percentage
1	0x5B1De8...A40D255B	5,000,000	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.