

Advanced Manual **Smart Contract Audit**

September 26, 2022

Audit requested by



Phuket Holiday Coin

0xd372380493213cEB1761673530b2b17Dfa3Eb5F0

Table of Contents

1. Audit Summary

- 1.1 Audit scope
- 1.2 Tokenomics
- 1.3 Source Code

2. Disclaimer

3. Global Overview

- 3.1 Informational issues
- 3.2 Low-risk issues
- 3.3 Medium-risk issues
- 3.4 High-risk issues

4. Vulnerabilities Findings

5. Contract Privileges

- 5.1 Maximum Fee Limit Check
- 5.2 Contract Pausability Check
- 5.3 Max Transaction Amount Check
- 5.4 Exclude From Fees Check
- 5.5 Ability to Mint Check
- 5.6 Ability to Blacklist Check
- 5.7 Owner Privileges Check

6. Notes

- 6.1 Notes by Coinsult
- 6.2 Notes by Phuket Holiday Coin

7. Contract Snapshot

8. Website Review

9. Certificate of Proof

Audit Summary

Audit Scope

Project Name	Phuket Holiday Coin
Website	phuketholidaycoin.me
Blockchain	Binance Smart Chain
Smart Contract Language	Solidity
Contract Address	0xd372380493213cEB1761673530b2b17Dfa3Eb5F0
Audit Method	Static Analysis, Manual Review
Date of Audit	26 September 2022

This audit report has been prepared by Coinsult's experts at the request of the client. In this audit, the results of the static analysis and the manual code review will be presented. The purpose of the audit is to see if the functions work as intended, and to identify potential security issues within the smart contract.

The information in this report should be used to understand the risks associated with the smart contract. This report can be used as a guide for the development team on how the contract could possibly be improved by remediating the issues that were identified.

Tokenomics

Rank	Address	Quantity (Token)	Percentage
1	0x529fdbc45005464a6bac1d135e71674f0c881605	5,000,000,000	100.0000%

Source Code

Coinsult was commissioned by Phuket Holiday Coin to perform an audit based on the following code:

<https://bscscan.com/address/0xd372380493213cEB1761673530b2b17Dfa3Eb5F0#code>

Disclaimer

This audit report has been prepared by Coinsult's experts at the request of the client. In this audit, the results of the static analysis and the manual code review will be presented. The purpose of the audit is to see if the functions work as intended, and to identify potential security issues within the smart contract.

The information in this report should be used to understand the risks associated with the smart contract. This report can be used as a guide for the development team on how the contract could possibly be improved by remediating the issues that were identified.

Coinsult is not responsible if a project turns out to be a scam, rug-pull or honeypot. We only provide a detailed analysis for your own research.

Coinsult is not responsible for any financial losses. Nothing in this contract audit is financial advice, please do your own research.





The information provided in this audit is for informational purposes only and should not be considered investment advice. Coinsult does not endorse, recommend, support or suggest to invest in any project.

Coinsult can not be held responsible for when a project turns out to be a rug-pull, honeypot or scam.

Global Overview







Manual Code Review

In this audit report we will highlight the following issues:

Vulnerability Level	Total	Pending	Acknowledged	Resolved
 Informational	0	0	0	0
 Low-Risk	2	0	2	0
 Medium-Risk	2	0	0	2
 High-Risk	1	0	0	1

Privilege Overview

Coinsult checked the following privileges:

Contract Privilege	Description
Owner can mint?	 Owner can mint new tokens
Owner can blacklist?	 Owner cannot blacklist addresses
Owner can set fees > 25%?	 Owner cannot set the sell fee to 25% or higher
Owner can exclude from fees?	 Owner cannot exclude from fees
Owner can pause trading?	 Owner cannot pause the contract
Owner can set Max TX amount?	 Owner cannot set max transaction amount

More owner privileges are listed later in the report.

● **Low-Risk:** Could be fixed, will not bring problems.

Different Pragma Directives Are Used

- Version used: ['0.6.12', '>=0.4.0', '>=0.6.0=0.6.4']
- >=0.6.0=0.6.0=0.6.4 (#101)
- >=0.6.0=0.4.0 (#360)
- 0.6.12 (#658)

Recommendation

Use one Solidity version.

● **Low-Risk:** Could be fixed, will not bring problems.

Avoid relying on `block.timestamp`

`block.timestamp` can be manipulated by miners.

```
require(now <= expiry, "PHCF::delegateBySig: signature expired");
```

Recommendation

Do not use `block.timestamp`, `now` or `blockhash` as a source of randomness

Exploit scenario

```
contract Game {  
  
    uint reward_determining_number;  
  
    function guessing() external{  
        reward_determining_number = uint256(block.blockhash(10000)) % 10;  
    }  
}
```

Eve is a miner. Eve calls `guessing` and re-orders the block containing the transaction. As a result, Eve wins the game.

● **Low-Risk:** Could be fixed, will not bring problems.

Too many digits

Literals with many digits are difficult to read and review.

```
uint256 public maxSupply = 1000000000 ether;
```

Recommendation

Use: Ether suffix, Time suffix, or The scientific notation

Exploit scenario

```
contract MyContract{
    uint 1_ether = 1000000000000000000;
}
```

While `1_ether` looks like 1 ether, it is 10 ether. As a result, it's likely to be used incorrectly.

● **Medium-Risk:** Should be fixed, could bring problems.

Duplicate mint functions (✅ Resolved)

```
function mint(uint256 amount) public onlyOwner returns (bool) {
    _mint(_msgSender(), amount);
    return true;
}

/// @notice Creates `_amount` token to `_to`. Must only be called by the owner (MasterChef).
function mint(address _to, uint256 _amount) public onlyMinter returns(bool) {
    uint256 totalSupplyAfterMint = totalSupply().add(_amount);
    require(totalSupplyAfterMint <= maxSupply, "PHCF::can't mint over max supply");
    _mint(_to, _amount);
    _moveDelegates(address(0), _delegates[_to], _amount);
    return true;
}
```

Recommendation

Remove the first mint function, it takes up space in the contract.

✅ Function removed

● **Medium-Risk:** Should be fixed, could bring problems.

Way to bypass `require(totalSupplyAfterMint < maxSupply)` (✅ Resolved)

```
/// @notice Creates `_amount` token to `_to`. Must only be called by the owner (MasterChef).
function mint(address _to, uint256 _amount) public onlyMinter returns(bool) {
    uint256 totalSupplyAfterMint = totalSupply().add(_amount);
    require(totalSupplyAfterMint < maxSupply, "PHCF::can't mint over max supply");
    _mint(_to, _amount);
    _moveDelegates(address(0), _delegates[_to], _amount);
    return true;
}
```

Recommendation

By using the other mint function, the owner can bypass the maxSupply require statement.

✅ Second mint function removed

● **High-Risk:** Must be fixed, will bring problems.

Mint functions requires totalSupply to be below maxSupply, but maxSupply can be updated ✅ Resolved

```
/// @notice Creates `_amount` token to `_to`. Must only be called by the owner (MasterChef).
function mint(address _to, uint256 _amount) public onlyOwner returns(bool) {
    uint256 totalSupplyAfterMint = totalSupply().add(_amount);
    require(totalSupplyAfterMint <= maxSupply, "PHCF::can't mint over max supply");
    _mint(_to, _amount);
    _moveDelegates(address(0), _delegates[_to], _amount);
    return true;
}

/**
 * @dev Update max supply.
 * Can only be called by the current operator.
 */
function updateMaxSupply(uint256 _maxSupply) public onlyOwner {
    emit MaxSupplyUpdated(_maxSupply);
    maxSupply = _maxSupply;
}
```

Recommendation

This way, the owner can mint unlimited amount of tokens. Remove the function to update maxSupply.

✅ Function removed to update maxSupply. Hardcoded at 10.000.000.000

Contract Privileges

Maximum Fee Limit Check


Coinsult tests if the owner of the smart contract can set the transfer, buy or sell fee to 25% or more. It is bad practice to set the fees to 25% or more, because owners can prevent healthy trading or even stop trading when the fees are set too high.

Type of fee	Description
Transfer fee	● Owner cannot set the transfer fee to 25% or higher
Buy fee	● Owner cannot set the buy fee to 25% or higher
Sell fee	● Owner cannot set the sell fee to 25% or higher

Type of fee	Description
Max transfer fee	0%
Max buy fee	0%
Max sell fee	0%


Contract Pausability Check

Coinsult tests if the owner of the smart contract has the ability to pause the contract. If this is the case, users can no longer interact with the smart contract; users can no longer trade the token.

Privilege Check	Description
Can owner pause the contract?	 Owner cannot pause the contract


Max Transaction Amount Check

Coinsult tests if the owner of the smart contract can set the maximum amount of a transaction. If the transaction exceeds this limit, the transaction will revert. Owners could prevent normal transactions to take place if they abuse this function.

Privilege Check	Description
Can owner set max tx amount?	 Owner cannot set max transaction amount

Exclude From Fees Check

Coinsult tests if the owner of the smart contract can exclude addresses from paying tax fees. If the owner of the smart contract can exclude from fees, they could set high tax fees and exclude themselves from fees and benefit from 0% trading fees. However, some smart contracts require this function to exclude routers, dex, cex or other contracts / wallets from fees.


Privilege Check	Description
Can owner exclude from fees?	 Owner cannot exclude from fees

Ability To Mint Check

Coinsult tests if the owner of the smart contract can mint new tokens. If the contract contains a mint function, we refer to the token's total supply as non-fixed, allowing the token owner to "mint" more tokens whenever they want.

A mint function in the smart contract allows minting tokens at a later stage. A method to disable minting can also be added to stop the minting process irreversibly.


Minting tokens is done by sending a transaction that creates new tokens inside of the token smart contract. With the help of the smart contract function, an unlimited number of tokens can be created without spending additional energy or money.

Privilege Check	Description
Can owner mint?	 Owner can mint new tokens

Ability To Blacklist Check

Coinsult tests if the owner of the smart contract can blacklist accounts from interacting with the smart contract. Blacklisting methods allow the contract owner to enter wallet addresses which are not allowed to interact with the smart contract.

This method can be abused by token owners to prevent certain / all holders from trading the token. However, blacklists might be good for tokens that want to rule out certain addresses from interacting with a smart contract.

Privilege Check	Description
Can owner blacklist?	 Owner cannot blacklist addresses

Other Owner Privileges Check

Coinsult lists all important contract methods which the owner can interact with.

⚠ Contract contains `delegateBySig`

Notes

Notes by Phuket Holiday Coin

No notes provided by the team.

Notes by Coinsult

Owner can mint until max 10.000.000.000

Contract Snapshot

This is how the constructor of the contract looked at the time of auditing the smart contract.

```
contract PhcfToken is BEP20 {
    // Transfer tax rate in basis points. (default 1%)
    uint16 public transferTaxRate = 100;

    // Burn rate % of transfer tax. (default 20% x 1% = 0.2% of total amount).
    uint16 public burnRate = 20;
    // Max transfer tax rate: 10%.
    uint16 public constant MAXIMUM_TRANSFER_TAX_RATE = 1000;
    // Burn address
    address public constant BURN_ADDRESS = 0x00000000000000000000000000000000dEaD;

    // Max transfer amount rate in basis points. (default is 0.1% of total supply)
    uint16 public maxTransferAmountRate = 10000;
    // Addresses that excluded from antiWhale
    mapping(address => bool) private _excludedFromAntiWhale;
    // Automatic swap and liquify enabled
    bool public swapAndLiquifyEnabled = false;
    // Min amount to liquify. (default 500 PHCFs)
    uint256 public minAmountToLiquify = 500 ether;
    // The swap router, modifiable. Will be changed to PHC Finance's router when our own AMM release
    IUniswapV2Router02 public phcfRouter;
    // The trading pair
    address public phcfPair;
    // In swap and liquify
    bool private _inSwapAndLiquify;
    // max supply
    uint256 public maxSupply = 10000000000 ether;

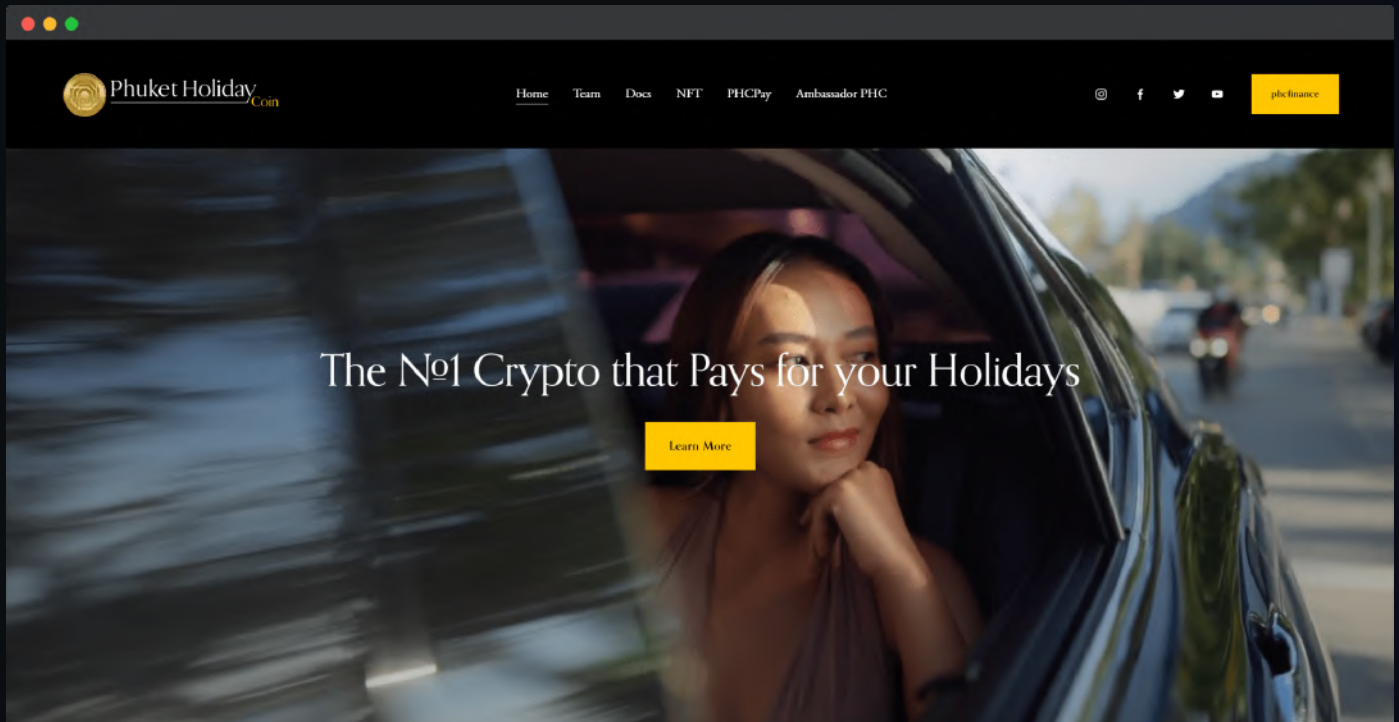
    // The operator can only update the transfer tax rate
    address private _operator;

    EnumerableSet.AddressSet private _minters;
    EnumerableSet.AddressSet private _blockAddrs;

    // Events
    event OperatorTransferred(address indexed previousOperator, address indexed newOperator);
```

Website Review

Coinsult checks the website completely manually and looks for visual, technical and textual errors. We also look at the security, speed and accessibility of the website. In short, a complete check to see if the website meets the current standard of the web development industry.



Type of check	Description
Mobile friendly?	● The website is mobile friendly
Contains jQuery errors?	● The website does not contain jQuery errors
Is SSL secured?	● The website is SSL secured
Contains spelling errors?	● The website does not contain spelling errors

Certificate of Proof

● Not KYC verified by Coinsult

Phuket Holiday Coin

Audited by Coinsult.net



Date: 26 September 2022

✓ Advanced Manual Smart Contract Audit

End of report
Smart Contract Audit

Request your smart contract audit / KYC

t.me/coinsult_tg