19th AUG 2021

SMART CONTRACT AUDIT REPORT

version v1.0

BEP20 Security Audit and General Analysis

HAECHI AUDIT

Table of Contents

0 Issues (0 Critical, 0 Major, 0 Minor) Found

About HAECHI AUDIT

HAECHI AUDIT is a flagship service of HAECHI LABS, the leader of the global blockchain industry. HAECHI AUDIT provides specialized and professional smart contract security auditing and development services.

We are experts with years of experience in the research and development of blockchain technology. We are the only blockchain technology company selected for the Samsung Electronics Startup Incubation Program in recognition of our expertise. We have also received technology grants from the Ethereum Foundation and Ethereum Community Fund.

Cryptocurrency exchanges worldwide trust HAECHI AUDIT's security audit reports. Indeed, numerous clients have successfully listed on Huobi, OKEX, Upbit, and Bithumb, etc. after passing HAECHI AUDIT smart contract security audit.

Representative clients and partners include the Global Blockchain Project and Fortune Global 500 corporations, in addition to Ground X (Kakao Corp. subsidiary), Carry Protocol, Metadium, LG, Hanwha, and Shinhan Bank. Over 60 clients have benefited from our most reliable smart contract security audit and development services.

Inquiries: audit@haechi.io

Website: audit.haechi.io

01. Introduction

This report was prepared to audit the security of Hodls1 smart contract created by Hodls Money team. HAECHI AUDIT conducted the audit focusing on whether the smart contract created by Hodls Money team is soundly implemented and designed as specified in the published materials, in addition to the safety and security of the smart contract.

The issues discovered are categorized into CRITICAL, MAJOR, MINOR, TIPS according to their importance.

CRITICAL	Critical issues must be resolved as critical flaws that can harm a wide range of users.
MAJOR	Major issues require correction because they either have security problems or are implemented not as intended.
MINOR	Minor issues can potentially cause problems and therefore require correction.
TIPS	Tips issues can improve the code usability or efficiency when corrected.

HAECHI AUDIT recommends Hodls Money team improve all issues discovered.

The following issue explanation uses the format of {file name}#{line number}, {contract name}#{function/variable name} to specify the code. For instance, *Sample.sol:20* points to the 20th line of Sample.sol file, and *Sample#fallback()* means the fallback() function of the Sample contract.

Please refer to the Appendix to check all results of the tests conducted for this report.

02. Summary

The codes used in this Audit can be found at Bscscan (https://testnet.bscscan.com/address/0x3C537d1f7C9ab3fb3C9eBCb76e9105878494c07D#code).

Issues

HAECHI AUDIT found 0 critical issues, 0 major issues, 0 minor issues. There is 1 Tips issue explained that would improve the code's usability and efficiency upon modification.

Severity	Issue	Status
TIPS	There is an unused function.	(Found - v.1.0)

03. Overview

Contracts Subject to Audit

- Context
- Address
- Ownable
- Hodls1

Key Features

Hodls Money team implemented BEP 20 Smart contracts that performs the following functions.

- Token transfer (Transfer)
- Reflection Token

Roles

Hodls1 Smart contract has the following privileges.

- **Owner**

The specification for the control of each privilege is as follows.

Role	MAX	Addable	Deletable	Transferable	Renouncable
Owner	1	X	X	O	O

Each privilege can access the following functions.

Role	Functions
Owner	*Ownable#transferOwnership()* *Ownable#renounceOwnership()* *Hodls1#excludeAccount()* *Hodls1#includeAccount()* *Hodls1#includeLockupAccount()* *Hodls1#setFee()* *Hodls1#setBurn()* *Hodls1#setTeamFee()* *Hodls1#setTeamAddress()*

04. Issues Found

TIPS : There is an unused function (Found - v.1.0)

TIPS

```
23   function _msgData() internal view virtual returns (bytes memory) {
24     this; // silence state mutability warning without generating bytecode
25     return msg.data;
26   }
```

[https://testnet.bscscan.com/address/0x3C537d1f7C9ab3fb3C9eBCb76e9105878494c07D#code#L19]

Context#_msgData() function is not in use.

```
1077   function _getTaxFee() private pure returns (uint256) {
1078     return _TAX_FEE;
1079   }
```

[https://testnet.bscscan.com/address/0x3C537d1f7C9ab3fb3C9eBCb76e9105878494c07D#code#L1084]

Hodls1#_getTaxFee() function is not in use.

```
1084   function _getMaxTxAmount() private pure returns (uint256) {
1085     return _MAX_TX_SIZE;
1086   }
```

[https://testnet.bscscan.com/address/0x3C537d1f7C9ab3fb3C9eBCb76e9105878494c07D#code#L1084]

Hodls1#_getMaxTxAmount() function is not in use.

05. Disclaimer

This report does not guarantee investment advice, the suitability of the business models, and codes that are secure without bugs. This report shall only be used to discuss known technical issues. Other than the issues described in this report, undiscovered issues may exist such as defects in Binance Smart Chain and Solidity. To write secure smart contracts, correction of discovered problems and sufficient testing thereof are required.

Appendix A. Test Results

The following results are unit test results that cover the key logic of the smart contract subject to the security audit. Parts marked in red are test cases that failed to pass the test due to having issues.

```
Hodls1
  #constructor()
    should set name properly
    should set symbol properly
    should set decimals properly
    should set initial supply properly
  BEP20 Spec
   #transfer()
     should fail if recipient is ZERO_ADDRESS
     should fail if sender is locked
     should fail if amount is zero
     should fail amount is greater than _MAX_TX_SIZE
     should fail if sender's amount is lower than balance (138ms)
    when succeeded
      sender's balance should decrease
      recipient's balance should increase
      should emit Transfer event
   #transferFrom()
     should fail if sender is ZERO_ADDRESS
     should fail if recipient is ZERO_ADDRESS
     should fail if sender's amount is lower than transfer amount (51ms)
     should fail if allowance is lower than transfer amount (77ms)
     should fail even if try to transfer sender's token without approve process (41ms)
    when succeeded
      sender's balance should decrease
      recipient's balance should increase
```

```
        should emit Transfer event
        allowance should decrease
        should emit Approval event
    #approve()
      should fail if spender is ZERO_ADDRESS
     valid case
        allowance should set appropriately
        should emit Approval event
    #increaseAllowance()
      should fail if spender is ZERO_ADDRESS
      should fail if overflows
     valid case
        allowance should set appropriately
        should emit Approval event
    #decreaseAllowance()
      should fail if spender is ZERO_ADDRESS
      should fail if overflows
     valid case
        allowance should set appropriately
        should emit Approval event
  #excludeAccount()
     should fail if msg.sender is not owner
     should fail if try to exclude router address
     should fail if account is already excluded
    valid case
      _tOwned should be set properly (102ms)
      _isExcluded should be true
      excludeList should be set properly
  #includeAccount()
     should fail if msg.sender is not owner
     should fail if account is already included
    valid case
      _isExcluded should be set false
      excludeList should be set properly
```

```
      _tOwned should be zero (86ms)
#includeLockupAccount()
    should fail if msg.sender is not owner
    should fail if account is already locked up
  valid case
      account should added to _lockupList
#setFee()
    should fail if msg.sender is not owner
  valid case
      _isAppliedFee should be set properly
#setBurn()
    should fail if msg.sender is not owner
  valid case
      _isAppliedBurn should be set properly
#setTeamFee()
    should fail if msg.sender is not owner
  valid case
      _isAppliedTeamFee should be set properly
#setTeamAddress()
    should fail if msg.sender is not owner
  valid case
      _teamAddress should be set properly
transfer standard
    sender's token reflection updates
    recipient's token reflection updates
    contract token reflection updates
    token supply updates
    teamAddress takes fee
transfer from excluded
    sender's token reflection updates
    sender's token updates
    recipient's token reflection updates
    contract token reflection updates
```

```
    token supply updates
    teamAddress takes fee
  transfer to excluded
    sender's token reflection updates
    recipient's token reflection updates
    recipient's token updates
    contract token reflection updates
    token supply updates
    teamAddress takes fee
  transfer both excluded
    sender's token reflection updates
    sender's token updates
    recipient's token reflection updates
    recipient's token updates
    contract token reflection updates
    token supply updates
    teamAddress takes fee
  Unused
    #_msgData() (85ms)
    #_getTaxFee()
    #_getMaxTxAmount()

Address
  #isContract()
    returns false for account address
    returns true for contract address
  #sendValue()
    should fail if try to transfer more than sender contract amounts
   when sender contract has ethers
      sends 0 wei
      sends non-zero amounts
      sends the whole balance
      should fail if try to send more than the amounts
   with contract recipient
```

```
      sends ether
      shoudl fail if recipient reverts
  #functionCall()
    with valid contract receiver
      calls the requested function
      reverts when the called function reverts with no reason
      reverts when the called function reverts, bubbling up the revert reason
      reverts when the called function runs out of gas
      reverts when the called function throws
      reverts when function does not exist
    with non-contract receiver
      reverts when address is not a contract
  #functionCallWithValue()
    with zero value
      calls the requested function
    with non-zero value
      reverts if insufficient sender balance
      calls the requested function with existing value
      calls the requested function with transaction funds
      reverts when calling non-payable functions

Ownable
  #constructor()
    should set msg.sender to owner
  #transferOwnership()
    should fail when msg.sender is not owner
    should fail if try to transfer ownership to ZERO_ADDRESS
    should change owner to newOwner
    should emit transferOwnership event
  #renounceOwnership()
    should fail when msg.sender is not owner
    owner release ownership
    should emit transferOwnership event
```

SafeMath
 add
  adds correctly
  reverts on addition overflow
 sub
  subtracts correctly
  reverts if subtraction result would be negative
 mul
  multiplies correctly
  multiplies by zero correctly
  reverts on multiplication overflow
 div
  divides correctly
  divides zero correctly
  returns complete number result on non-even division
  reverts on division by zero
 mod
  reverts with a 0 divisor
  modulos correctly
   when the dividend is smaller than the divisor
   when the dividend is equal to the divisor
   when the dividend is larger than the divisor
   when the dividend is a multiple of the divisor

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/					
Hodls1.sol	**100**	**100**	**100**	**100**	

[Table 1] Test Case Coverage