10th APRIL 2020

Fromm Car

SMART CONTRACT AUDIT REPORT

-

version 1.0

ERC20 Security Audit and General Analysis

HAECHI AUDIT

Table of Contents

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

Table of Contents

About HAECHI AUDIT

01. Introduction

02. Summary

- Issues
- Notices

03. Overview

- Contracts Subject to Audit
- Key Features
- Roles
- Notice
  - Owner가 Renounce 될 수 있습니다.
  - Owner가 무한히 토큰을 추가발행 할 수 있습니다.

04. Issues Found

05. Disclaimer

Appendix A. Test Results

Appendix B. Known Issues

About HAECHI AUDIT

HAECHI AUDIT은 글로벌 블록체인 업계를 선도하는 HAECHI LABS의 대표 서비스 중 하나로, 스마트 컨트랙트 보안 감사 및 개발을 전문적으로 제공합니다.

다년간 블록체인 기술 연구 개발 경험을 보유하고 있는 전문가들로 구성되어 있으며, 그 전문성을 인정받아 블록체인 기술 기업으로는 유일하게 삼성전자 스타트업 육성 프로그램에 선정된 바 있습니다. 또한, 이더리움 재단과 이더리움 커뮤니티 펀드로부터 기술 장려금을 수여받기도 하였습니다.

HAECHI AUDIT의 보안감사 보고서는 전세계 암호화폐 거래소들의 신뢰를 받고 있습니다. 실제로 많은 클라이언트들이 HAECHI AUDIT 스마트 컨트랙트 보안감사를 거친 후에, Huobi, OKEX, Upbit, Bithumb 등에 성공적으로 상장하였습니다.

대표적인 클라이언트 및 파트너사로는 글로벌 블록체인 프로젝트와 포춘 글로벌 500대 기업들이 있으며, 카카오의 자회사인 Ground X, Carry 프로토콜, Metadium, LG, 한화, 신한은행 등이 있습니다. 지금까지 약 60여곳 이상의 클라이언트를 대상으로 가장 신뢰할 수 있는 스마트 컨트랙트 보안감사 및 개발 서비스를 제공하였습니다.

문의 : audit@haechi.io
웹사이트 : audit.haechi.io

01. Introduction

본 보고서는 FrommCar 팀이 제작한 FrommCar 스마트 컨트랙트의 보안을 감사하기 위해 작성되었습니다. HAECHI AUDIT 는 FrommCar 팀이 제작한 스마트 컨트랙트의 구현 및 설계가 공개된 자료에 명시한 것처럼 잘 구현이 되어있고, 보안상 안전한지에 중점을 맞춰 감사를 진행했습니다.

발견된 이슈는 중요도 차이에 따라 CRITICAL, MAJOR, MINOR, TIPS 로 나누어집니다.

CRITICAL	Critical 이슈는 광범위한 사용자가 피해를 볼 수 있는 치명적인 보안 결점으로 반드시 해결해야 하는 사항입니다.
MAJOR	Major 이슈는 보안상에 문제가 있거나 의도와 다른 구현으로 수정이 필요한 사항입니다.
MINOR	Minor 이슈는 잠재적으로 문제를 발생시킬 수 있으므로 수정이 요구되는 사항입니다.
TIPS	Tips 이슈는 수정했을 때 코드의 사용성이나 효율성이 더 좋아질 수 있는 사항입니다.

HAECHI AUDIT는 FrommCar 팀이 발견된 모든 이슈에 대하여 개선하는 것을 권장합니다.

이어지는 이슈 설명에서는 코드를 세부적으로 지칭하기 위해서 {파일 이름}#{줄 번호}, {컨트랙트 이름}#{함수/변수 이름} 포맷을 사용합니다. 예를 들면, *Sample.sol:20*은 Sample.sol 파일의 20번째 줄을 지칭하며, *Sample#fallback()* 는 Sample 컨트랙트의 fallback() 함수를 가리킵니다

보고서 작성을 위해 진행된 모든 테스트 결과는 Appendix에서 확인 하실 수 있습니다.

02. Summary

Audit에 사용된 코드는 Github (https://github.com/HAECHI-LABS/FrommCarToken)에서 찾아볼 수 있습니다. Audit에 사용된 코드의 마지막 커밋은 "72b1481e92a08e534f31e097cac257597a380f17"입니다.

Issues

HAECHI AUDIT에서는 Critical 이슈 0개, Major 이슈 0개, Minor 이슈 0개를 발견하였으며 수정했을 때 코드의 사용성이나 효율성이 더 좋아질 수 있는 사항들을 0개의 Tips 카테고리로 나누어 서술하였습니다.

Notices

Notice는 스마트 컨트랙트 사용자와 운영자가 모두 사용상에 주의해야 할 점들 입니다. 이 파트에서는 FrommCar 팀에서 의도한 사항으로 확인된 이슈 또는 스마트 컨트랙트 코드가 아닌 외부 요인에 의해 발생할 수 있는 이슈를 다룹니다.

Category	Issue	Status
Notice	**Owner**가 Renounce 될 수 있습니다.	(Noticed - v1.0)

03. Overview

Contracts Subject to Audit

- ERC20
- ERC20Burnable
- ERC20Lockable
- ERC20Mintable
- FrommCar
- Library
    - SafeMath
    - Freezable
    - Ownable
    - Pausable

Key Features

FrommCar 팀은 아래의 기능을 수행하는 ERC20 Smart contract를 구현하였습니다.

- 토큰 소각
- 토큰 전송 제한
- 계좌별 토큰 전송 제한
- 토큰 발행

Roles

FrommCar Smart contract에는 다음과 같은 권한이 있습니다.

- **Owner**

각 권한의 제어에 대한 명세는 다음과 같습니다.

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

각 권한으로 접근 할 수 있는 기능은 다음과 같습니다.

Role	Functions
Owner	*ERC20Lockable#releaseLock()* *ERC20Mintable#mint()* *ERC20Mintable#finishMint()* *Freezable#freeze()* *Freezable#unFreeze()* *Ownable#transferOwnership()* *Ownable#renounceOwnership()* *Pausable#pause()* *Pausable#unPause()*

Notice

- **Owner가 Renounce 될 수 있습니다.**

  *Ownable#transferOwnership()*에서 ZERO_ADDRESS를 입력하거나, *Ownable#renounceOwnership()*를 호출 하는 경우 **Owner**가 사라질 수 있으며 위 표에 명시된 **Owner**만 호출 할 수 있는 함수를 호출 할 수 없게 됩니다.

04. Issues Found

발견된 이슈가 없습니다.

05. Disclaimer

해당 리포트는 투자에 대한 조언, 비즈니스 모델의 적합성, 버그 없이 안전한 코드를 보증하지 않습니다. 해당 리포트는 알려진 기술 문제들에 대한 논의의 목적으로만 사용됩니다. 리포트에 기술된 문제 외에도 이더리움, 솔리디티 상의 결함 등 발견되지 않은 문제들이 있을 수 있습니다. 안전한 스마트 컨트랙트를 작성하기 위해서는 발견된 문제들에 대한 수정과 충분한 테스트가 필요합니다.

Appendix A. Test Results

아래 결과는, 보안 감사 대상인 스마트 컨트랙트의 주요 로직을 커버하는 unit test 결과입니다. 붉은색으로 표시된 부분은 이슈가 존재하여 테스트에 통과하지 못한 테스트 케이스입니다.

```
Contract: Pausable
  #constructor()
    ✓ should success construct contract (1304ms)
  after initialization
    #pause()
      ✓ should fail when already paused (77ms)
      ✓ should fail if msg.sender is not pauser (41ms)
      ✓ should emit Paused event for valid case (55ms)
    #unPause()
      ✓ should fail when already unpaused
      ✓ should fail if msg.sender is not pauser (67ms)
      ✓ should emit Unpaused event for valid case (68ms)

Contract: Pausable
  #freeze()
    ✓ should fail if msg.sender is not owner (56ms)
    valid case
      ✓ target address freezed
      ✓ should emit Freeze event
  #unFreeze()
    ✓ should fail if msg.sender is not owner
    valid case
      ✓ target address unfreezed
      ✓ should emit Unfreeze event

Contract: FrommCar
  #constructor()
    ✓ contract caller set to owner
    ✓ contract initializer's balance set to initial supply
    ✓ name, symbol, decimals set properly (43ms)
  ERC20 Spec
    #transfer()
      ✓ should fail if recipient is ZERO_ADDRESS
      ✓ should fail if sender's amount is lower than balance
      modifiers
```

```
    ✓ should not work when frozen
    ✓ should not work when paused
    ✓ should not be able to send more than user's unlocked balance (133ms)
   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 (51ms)
   ✓ should fail if sender's amount is lower than transfer amount (112ms)
   ✓ should fail if allowance is lower than transfer amount (49ms)
   ✓ should fail even if try to transfer sender's token without approve process
   modifiers
    ✓ should not work when frozen
    ✓ should not work when paused
    ✓ should not be able to send more than user's unlocked balance (103ms)
   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
ERC20Lockable Spec
  #transferWithLockUp()
   ✓ should fail if locked is ZERO_ADDRESS
   ✓ should fail if sender's amount is lower than burnAmount (44ms)
   ✓ should fail if try to lock with set due to past time
   valid case
    ✓ sender's balance should decrease
    ✓ locked's balance should increase
    ✓ locked's total locked amount should increase
    ✓ locked's lock info update properly
    ✓ should emit Transfer event
    ✓ should emit Lock event
  #unlock()
   valid case
```

```
    ✓ locked user's amount should increase amount of locked
    ✓ should delete lock information (76ms)
    ✓ should emit Unlock event
  #releaseLock()
   ✓ should fail if msg.sender is not owner
   valid case
    ✓ locked user's amount should not change
    ✓ should delete lock information
    ✓ should emit Unlock event
ERC20Mintable Spec
  #mint()
   ✓ should fail if msg.sender is not owner
   ✓ should fail when paused (40ms)
   ✓ should not be able to mint over cap (51ms)
   ✓ should fail if overflows
   ✓ should fail if try to mint to ZERO_ADDRESS
   ✓ should fail if mint is finished (55ms)
   valid case
    ✓ receiver's amount should increase
    ✓ totalSupply should increase
    ✓ should emit Transfer event
    ✓ should emit Mint event
  #finishMint()
   ✓ should fail if msg.sender is not owner
   ✓ should fail if already finished (40ms)
   ✓ should set _mintingFinished to true (39ms)
ERC20Burnable Spec
  #burn()
   ✓ should fail if overflows
   modifiers
    ✓ should not work when paused
   valid case
    ✓ totalSupply should decrease
    ✓ account's balance should decrease
    ✓ should emit Transfer event
    ✓ should emit Burn event
  #burnFrom()
   ✓ should fail if account is ZERO_ADDRESS
   ✓ should fail if account's amount is lower than burn amount (51ms)
   ✓ should fail if allowance is lower than burn amount (46ms)
   ✓ should fail even if try to burn account's this.token without approve process (41ms)
   ✓ should fail when paused (63ms)
```

```
    modifiers
      ✓ should not work when paused
    valid case
      ✓ totalSupply should decrease
      ✓ account's balance should decrease
      ✓ should emit Transfer event
      ✓ allowance should decrease
      ✓ should emit Approval event
      ✓ should emit Burn event

Contract: Ownable
  #constructor()
    ✓ should set owner to contract initializer
  #owner()
    ✓ should return appropriate owner
  #renounceOwnership()
    ✓ should fail if msg.sender is not owner
    valid case
      ✓ should emit OwnershipTransferred event
  #transferOwnership()
    ✓ should fail if msg.sender is not owner
    ✓ should fail if newOwner is ZERO_ADDRESS
    valid case
      ✓ should emit OwnershipTransferred event
      ✓ should set owner to newOwner

Contract: 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 divison 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					
FrommCar.sol	**100**	**100**	**100**	**100**	
contracts/erc20					
ERC20.sol	**100**	**100**	**100**	**100**	
ERC20Burnable.sol	**100**	**100**	**100**	**100**	
ERC20Lockable.sol	**100**	**100**	**100**	**100**	
ERC20Mintable.sol	**100**	**100**	**100**	**100**	
contracts/library/					
Freezable.sol	**100**	**100**	**100**	**100**	
Ownable.sol	**100**	**100**	**100**	**100**	
Pausable.sol	**100**	**100**	**100**	**100**	
SafeMath.sol	**100**	**100**	**100**	**100**	

[표 1] Test Case Coverage

Appendix B. Known Issues

Description	SWC	Test Result
Function Default Visibility	**100**	**Pass**
Integer Overflow and Underflow	**101**	**Pass**
Outdated Compiler Version	**102**	**Pass**
Unprotected Ether Withdrawal	**105**	**Pass**
Unprotected SELFDESTRUCT Instruction	**106**	**Pass**
Reentrancy	**107**	**Pass**
State Variable Default Visibility	**108**	**Pass**
Uninitialized Storage Pointer	**109**	**Pass**
Use of Deprecated Solidity Functions	**111**	**Pass**
Delegatecall to Untrusted Callee	**112**	**Pass**
Authorization through tx.origin	**115**	**Pass**
Timestamp Dependence	**116**	**Pass**
Signature Malleability	**117**	**Pass**
Shadowing State Variables	**119**	**Pass**
Randomness from Chain Attributes	**120**	**Pass**
Missing Protection against Signature Replay Attacks	**121**	**Pass**
Lack of Proper Signature Verification	**122**	**Pass**
Write to Arbitrary Storage Location	**124**	**Pass**
Incorrect Inheritance Order	**125**	**Pass**
Right to Left Override control character	**130**	**Pass**
Presence of unused variables	**131**	**Pass**