11th DECEMBER 2020

SMART CONTRACT AUDIT REPORT

-

version v2.0

Smart Contract Security Audit and General Analysis

HAECHI AUDIT

Table of Contents

2 Issues (1 Critical, 0 Major, 1 Minor) Found

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 Neverlose.money 팀이 제작한 Neverlose.money 스마트 컨트랙트의 보안을 감사하기 위해 작성되었습니다. HAECHI AUDIT 는 Neverlose.money 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 Neverlose.money 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/Steemhunt/neverlose.money-contract)에서 찾아볼 수 있습니다.
Audit에 사용된 코드의 마지막 커밋은 "ea243ff3fe785d7fff801fe5eac510204bf55d02"입니다.

Issues

HAECHI AUDIT에서는 Critical 이슈 1개, Major 이슈 0개, Minor 이슈 1개를 발견하였으며 수정했을 때 코드의 사용성이나 효율성이 더 좋아질 수 있는 사항들을 1개의 Tips 카테고리로 나누어 서술하였습니다.

Severity	Issue	Status
CRITICAL	LockUpPool#addLockUpPool(), WRNRewardPool#addLockUpRewardPool() 에서 maxLockUpLimit이 0 이어도 등록됩니다	(Found - v1.0) (Resolved - v2.0)
MINOR	LockUpPool#addLockUpPool()를 이용하여 WRNRewardPool#addLockUpRewardPool() 를 사용하지 않고 새로운 pool을 등록 할 수 있습니다	(Found - v1.0) (Acknowledged - v2.0)
TIPS	코드의 복잡도를 낮추기 위해 컨트랙트 구조의 변경을 권장합니다	(Found - v1.0)

Update

[v2.0] - 새로운 커밋 a7a9ed1a8dd83ed2b3ece61ada858f7ae171a203 에서 1개의 이슈가 해결되었으며 1개의 이슈가 의도한 것이라는 답변을 받았습니다.

03. Overview

Contracts Subject to Audit

- ERC20Token
- LockUpPool
- WRNRewardPool

Key Features

Neverlose.money 팀은 아래의 기능을 수행하는 Smart contract를 구현하였습니다.

- 특정 토큰에 대한 pool 등록
- 각각의 pool에 토큰을 예치하고 예치 기간을 지키지 못한 경우 패널티 부과
- 예치 기간을 준수한 경우, 해당 pool에 모인 패널티를 분배
- 비율에 따라 WRN 토큰 배분

Roles

Neverlose.money 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

권한으로 접근 할 수 있는 기능은 다음과 같습니다.

Role	Functions
Owner	*LockUpPool#addLockUpPool()* *LockUpPool#updateMaxLimit()* *LockUpPool#setEmergencyMode()* *LockUpPool#setFundAddress()* *WRNRewardPool#addLockUpRewardPool()* *WRNRewardPool#updatePoolMultiplier()* *Ownable#transferOwnership()* *Ownable#renounceOwnership()*

04. Issues Found

CRITICAL: LockUpPool#addLockUpPool(), WRNRewardPool#addLockUpRewardPool()에서 maxLockUpLimit을 0으로 등록할 수 있습니다. [Unexpected Behavior] (Found - v1.0)

CRITICAL

Problem Statement

LockUpPool#addLockUpPool(), WRNRewardPool#addLockUpRewardPool() 함수에서는 maxLockUpLimit을 인자로 받습니다. 이 때 등록된 pool임을 확인하기위해 해당 pool의 maxLockUpLimit이 0인지 확인합니다. 하지만 maxLockUpLimit이 0으로 입력되어도 함수가 실패하지 않기 때문에, maxLockUpLimit값을 0으로 입력하여 같은 pool을 여러번 등록 할 수 있습니다.

이로 인해 생기는 부작용은 다음과 같습니다

- LockUpPool#pools 배열의 길이가 길어집니다.
- LockUpPool#pools 배열의 길이가 길어지므로 WRNReward#updateAllPools() 가 실패할 가능성이 높아집니다.
- WRNRewardPool#addLockUpRewardPool() 에서 multiplier가 0이 아니고, maxLockUpLimit이 0으로 여러번 호출하면 totalMultiplier와 실제 pool별 multiplier의 합이 달라집니다

- WRNRewardPool#addLockUpRewardPool() 에서 multiplier가 255이고, maxLockUpLimit이 0이 되게 258 번 호출하면 totalMultiplier가 overflow 가 발생하여 줄어들게 되고 pool의 multiplier보다 totalMultiplier가 더 작아질 수 있습니다.

Recommendation

pool을 등록하는 함수에 maxLockUpLimit이 0이면 등록에 실패하도록 변경하십시오.

Update

[v2.0] - LockUpPool#addLockUpPool() 에서 maxLockUpLimit이 0이면 등록에 실패하도록 변경되어 이슈가 해결되었습니다

MINOR : LockUpPool#addLockUpPool()를 이용하여 WRNRewardPool#addLockUpRewardPool()를 사용하지 않고 새로운 pool을 등록 할 수 있습니다. (Found - v1.0) (Acknowledged - v2.0)

MINOR

Problem Statement

LockUpPool#addLockUpPool(), WRNRewardPool#addLockUpRewardPool() 함수는 둘 다 pool을 등록하는 함수입니다. WRNRewardPool#addLockUpRewardPool() 함수가 WRNRewardPool을 운영하는 과정에서 주로 사용될 함수이지만, LockUpPool#addLockUpPool() 함수 또한 사용이 가능합니다.

Recommendation

일관된 사용을 보장하기 위해 컨트랙트 구조 변경 및 함수의 visibility를 변경 하시는 것을 추천드립니다

Update

[v2.0] - NeverLose.money 팀에서 해당 이슈를 인지하고 있으며, 위 함수의 visibility가 public 인 것은 unittest를 진행하기 위해 의도된 것 이라고 답변하였습니다

TIPS : 코드의 복잡도를 낮추기 위해 컨트랙트 구조의 변경을 권장합니다(Found - v1.0)

TIPS

현재 구현에서는, 하나의 컨트랙트로 여러가지 토큰의 pool을 관리하는 구조 입니다. 이로 관리하기 위해 pool에 해당하는 정보를 담는 구조체들을 사용하고 있고 pool별로 유저의 값을 관리하는 구조체가 각각 pool에 별도로 저장되는 형태를 띄고 있습니다. 여러 구조체를 중첩하여 사용하게 되는 경우, 컨트랙트의 난이도가 올라가고 가독성을 낮추게 됩니다. 이를 해결하기 위해, 각각의 토큰의 예치/벌금 부과/벌금 분배 만을 담당하는 Pool 컨트랙트와, pool에게 WRN을 분배하고 새로운 pool등록을 담당하는 WRNRewrader로 분리하여 WRNRewarder를 통해 Pool을 배포하고 관리할 수 있도록 변경 하시는 것을 추천드립니다

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 블록체인 프로토콜 상의 결함 등 발견되지 않은 문제들이 있을 수 있습니다. 안전한 스마트 컨트랙트를 작성하기 위해서는 발견된 문제들에 대한 수정과 충분한 테스트가 필요합니다.