4th JUNE 2021

XIDO FINANCE

SMART CONTRACT AUDIT REPORT

-

version v1.0

ERC20 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 aims to audit the security of XIDO smart contract created by Xido.finance team. HAECHI AUDIT conducted the audit focusing on whether the smart contract created by Xido.finance 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 are those that can improve the usability or efficiency of the code when corrected.

HAECHI AUDIT recommends Xido.finance team improve all issues discovered.

The following issue description 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 GitHub (https://github.com/xidofinance/TokenContract/tree/34541de655df3faead0803e698f3d20ce549f9b1). The last commit of the code used for this Audit is "34541de655df3faead0803e698f3d20ce549f9b1".

Issues

HAECHI AUDIT found 0 critical issues, 0 major issues, and 0 minor issues. There are 2 Tips issues explained for improving the code's usability and efficiency.

Severity	Issue	Status
TIPS	There is a missing Event	(Found - v1.0)
TIPS	There is an unused function	(Found - v1.0)

03. Overview

Contracts Subject to Audit

- SafeMath
- Ownable
- ERC20
- XIDO

Key Features

Xido.finance team implemented ERC20 Smart contract that performs the following functions.

- Token burn (Burn)
- Token transfer restriction (freeze)

Roles

XIDO Smart contract has the following authorities.

- Owner

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

Role	MAX	Addable	Deletable	Transferabl e	Renouncabl e
Owner	1	X	X	O	X

Each authority can access the following functions.

Role	Functions
Owner	*Ownable#transferOwnership()* *XIDO#freezeAccount()* *XIDO#unfreezeAccount()* *XIDO#burn()*

04. Issues Found

TIPS : There is a missing Event (Found - v.1.0)

TIPS

The following is a list of functions with a missing Event.

Function	Expected Event	Emitted Event	Omitted Event
transferFrom	Transfer, Approval	Transfer	Approval
burn	Transfer, Burn	Burn	Transfer

When an Event is missing, it is difficult to identify in real-time whether an accurate value is recorded on the blockchain. In this case, it is difficult to determine whether the corresponding value has been changed and whether the corresponding function has been called in the application.

Thus, we recommend adding an Event that corresponds to the change occurring in the pertinent function.

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

TIPS

```
function isContract(address addr) internal view returns (bool) {
    uint size;
    assembly{size := extcodesize(addr)}
    return size > 0;
}
```

[https://github.com/xidofinance/TokenContract/blob/34541de655df3faead0803e698f3d20ce549f9b1/Token.sol#L197-L201]

Although *XIDO#isContract()* function is declared as internal, there is no place to use it inside the XIDO contract.

05. Disclaimer

This report does not guarantee investment advice, 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 Ethereum and Solidity. In order 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.

```
  Ownable
  #constructor()
    ✓ should set msg.sender to owner
    ✓ should set ZERO address to newOwner
  #transferOwnership()
    ✓ should fail when msg.sender is not owner
    ✓ should fail if try to transfer ownership to ZERO_ADDRESS
    valid case
      ✓ should set newOwner properly
  #acceptOwnership()
    ✓ should fail when msg.sender is not newOwner
    vaild case
      ✓ should change owner to newOwner
      ✓ should emit OwnershipTransferred 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


Xido Finance
  #constructor()
    ✓ name set properly
    ✓ symbol set properly
    ✓ decimals set properly
    ✓ initialSupply set properly
    ✓ totalSupply set properly
  ERC20 Spec
    #approve()
      valid case
        ✓ allowance should set appropriately
        ✓ should emit Approval event
    #transfer()
    ✓ should fail if recipient is AddressZero
    ✓ should fail if sender's amount is more than balance
    ✓ should fail if sender is freezed
    valid case
      ✓ sender's balance should decrease
      ✓ receipient's balance should increase
      ✓ should emit Transfer event
  #transferFrom()
    ✓ should fail if recipient is AddressZero
    ✓ should fail if sender's amount is more than balance
    ✓ should fail if sender's amount is more than allowance
    ✓ should fail if try to transfer sender's token without approve process
    ✓ should fail if sender is freezed
    valid case
      ✓ sender's balance should decrease
      ✓ receipient's balance should increase
      ✓ allowance should decrease
      ✓ should emit Transfer event
      1) should emit Approval event
  ERC20Burnable spec
    #burn()
      ✓ should fail if msg.sender is not owner
      ✓ should fail if try to burn more than burner's balance
      valid case
        ✓ burner's balance should decrease
        ✓ totalSupply should decrease
        2) should emit Transfer event
        ✓ should emit Burn event
Freezable spec
```

```
#freezeAccount()
  ✓ should fail if msg.sender is not owner
  ✓ should fail if account is already freezed
  valid case
    ✓ account should be freezeAccount
    ✓ should emit Freeze event
#unfreezeAccount()
  ✓ should fail if msg.sender is not owner
  ✓ should fail if account is already unfreezed
  valid case
    ✓ account should be unfreezed
    ✓should emit Unfreeze event
```

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/					
XIDO.sol	100	97.73	100	100	

[Table 1] Test Case Coverage

(Because a case where msg.sender is ZERO_ADDRESS in Ownable#onlyNewOwner modifier cannot happen, Branch Coverage will not be 100%.)