HAECHI AUDIT

Eco Defi

Smart Contract Security Analysis

Published on : Oct 13, 2021

Version v1.0

HAECHI AUDIT

Smart Contract Audit Certificate

Eco Defi

Security Report Published by HAECHI AUDIT
v1.0 Oct 13, 2021

Auditor : Felix Kim

[illegible]

Executive Summary

Severity of Issues	Findings	Resolved	Unresolved	Acknowledged	Comment
Critical	-	-	-	-	-
Major	-	-	-	-	-
Minor	-	-	-	-	-
Tips	2	-	-	-	-

TABLE OF CONTENTS

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

ABOUT US

We have the vision to empower the next generation of finance. By providing security and trust in the blockchain industry, we dream of a world where everyone can easily access blockchain technology.

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 a team of experts with years of experience in the blockchain field and have been trusted by 300+ project groups. Our notable partners include Sushiswap,1inch, Klaytn, Badger, etc.

HAECHI AUDIT is 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.

Inquiries : audit@haechi.io
Website : audit.haechi.io

INTRODUCTION

This report was prepared to audit the security of ECOPToken smart contract created by Eco Defi team. HAECHI AUDIT focused on whether the smart contract created by Eco Defi team is soundly implemented and designed as specified in the published materials, in addition to the safety and security of the smart contract.

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 Eco Defi 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, while *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.

SUMMARY

The codes used in this Audit can be found at Etherscan (https://etherscan.io/address/0x0106a1122fe94a9cf151097c1fe17229ec78ffad#code).

Issues HAECHI AUDIT found 0 critical issues, 0 major issues, and 0 minor issues. There are 2 issues categorized as Tips that can improve the code's usability or efficiency upon modification.

Severity	Issue	Status
TIPS	There are missing Events.	(Found - v1.0)
TIPS	The SafeMath#pwr function returns a specific value for an undefined value.	(Found - v1.0)

OVERVIEW

Contracts subject to audit

- ❖ SafeMath
- ❖ owned
- ❖ ECOPToken

FINDINGS

💡 **TIPS**

There are missing events.

(Found - v.1.0)

The following list shows functions with missing Events.

Function	Expected Event	Emitted Event	Omitted Event
burn	Transfer, Burn	Transfer	Burn
burnFrom	Transfer, Burn, Approval	Transfer	Burn, Approval
transferFrom	Transfer, Approval	Transfer	Approval
transferOwnership	OwnershipTransferred	-	OwnershipTransferred

Without Event, it is difficult to identify in real-time whether correct values are recorded on the blockchain. In this case, it becomes problematic to determine whether the corresponding value has been changed in the application and whether the corresponding function has been called.

Thus, we recommended adding Events corresponding to the change occurring in the function.

💡**TIPS**

The SafeMath#pwr function returns a specific value for an undefined value.

(Found - v.1.0)

```
function pwr(uint256 x, uint256 y)
internal
pure
returns(uint256) {
  if (x == 0)
    return (0);
  else if (y == 0)
    return (1);
  else {
    uint256 z = x;
    for (uint256 i = 1; i < y; i++)
      z = mul(z, x);
    return (z);
  }
}
```

[https://etherscan.io/address/0x0106a1122fe94a9cf151097c1fe17229ec78ffad#code#L195-L209]

The *SafeMath#pwr(x, y)* function returns the value of x^y.

In math, 0^0 is an undefined value. However, currently, when *SafeMath#pwr(0, 0)* is called, the value of 0 is returned.

It is recommended to revert with an appropriate error message when *SafeMath#pwr(0, 0)* is called.

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 on the main-net. In order to write secure smart contracts, correction of discovered problems and sufficient testing thereof are required.

Appendix A. Test Results

The following results show the unit test results covering 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 existing issues.

```
ECOPToken Coin
  #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 sender's amount is more than balance
      valid case
        ✓ sender's balance should decrease
        ✓ recipient's balance should increase
        ✓ should emit Transfer event
    #transferFrom()
      ✓ should fail if sender's amount is more than balance
      ✓ should fail if sender's amount is more than allowance
      valid case
        ✓ sender's balance should decrease
        ✓ recipient's balance should increase
        ✓ allowance should decrease
        ✓ should emit Transfer event
        1) should emit Approval event
    ERC20Burnable spec
      #burn()
        ✓ 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
      #burnFrom()
        ✓ should fail if value is greater than allowance
        ✓ should fail if value is greater than balance
        valid case
```

```
          ✓ allowance should decrease
          ✓ burner's balance should decrease
          ✓ totalSupply should decrease
          3) should emit Approval event
          4) should emit Transfer event
          ✓ should emit Burn event

  owned
    #constructor()
      ✓ should set msg.sender to owner
    #transferOwnership()
      ✓ should fail when msg.sender is not owner
      valid case
        ✓ should set newOwner properly

  SafeMath
    sqrt
      ✓ calculate correctly
    sq
      ✓ calculate correctly
      ✓ reverts on multiplication overflow
    pwr
      6) should reverts if base and exponent is zero
      ✓ returns 0 if base is 0
      ✓ returns 1 if exponent is 0
      ✓ calculate correctly
      ✓ reverts on multiplication overflow
    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/					
ECPOToken.sol	100	100	100	100	

[Table 1] Test Case Coverage

End of Document