ETH3.0

Smart Contract Security Audit

No. 202308111616

Aug 11th, 2023

SECURING BLOCKCHAIN ECOSYSTEM

Contents

Summary of Audit Results

a

Project Description:

1. Basic Token Information

Token name	Ethereum 2 . 0
Token symbol	ETH3.0
Decimals	18
Total supply	120,269,517
Token type	ERC-20

Table 1 ETH3.
0token info

2. Business overview

Amount	Unlock time(UTC)
5,000,000	2023.09.01 00:00
5,000,000	2023.10.01 02:52
5,000,000	2023.11.01 02:54
5,000,000	2023.12.01 02:55
10,000,000	2024.02.01 02:56
10,000,000	2024.04.01 02:57
10,000,000	2024.06.01 02:58
10,000,000	2024.08.01 02:59
10,000,000	2024.10.01 03:03
10,000,000	2024.12.01 05:25

Table 2 ETH3.
0token lock information

1 Overview

1.1 Project Overview

Project Name	ETH3.0
Project language	Solidity
Platform	Ethereum
Contract Address	0x49a9363DB20615e455eb67B2b724739Ea0A6c81D

1.2 Audit Overview

Audit work duration: Aug 11, 2023 – Aug 11, 2023

Audit team: Beosin Security Team

1.3 Audit Method

The audit methods are as follows:

1. Formal Verification

Formal verification is a technique that uses property-based approaches for testing and verification. Property specifications define a set of rules using Beosin's library of security expert rules. These rules call into the contracts under analysis and make various assertions about their behavior. The rules of the specification play a crucial role in the analysis. If the rule is violated, a concrETH3.0 testcase is provided to demonstrate the violation.

2. Manual Review

Using manual auditing methods, the code is read line by line to identify potential security issues. This ensures that the contract's execution logic aligns with the client's specifications and intentions, thereby safeguarding the accuracy of the contract's Business logic.

The manual audit is divided into three groups to cover the entire auditing process:

The Basic Testing Group is primarily responsible for interpreting the project's code and conducting comprehensive functional testing.

The Simulated Attack Group is responsible for analyzing the audited project based on the collected historical audit vulnerability database and security incident attack models. They identify potential attack vectors and collaborate with the Basic Testing Group to conduct simulated attack tests.

The Expert Analysis Group is responsible for analyzing the overall project design, interactions with third parties, and security risks in the on-chain operational environment. They also conduct a review of the entire audit findings.

3. Static Analysis

Static analysis is a method of examining code during compilation or static analysis to ETH3.0 ctissues. Beosin-VaaS can ETH3.0more than 100 common smart contract vulnerabilities through static analysis, such as reentrancy and block paramETHEr dependency. It allows early and efficient discovery of problems to improve code quality and security.

2 Audit Content

2.1 Datailed Audit of Contract ETH3.0

(1) Approve functions

- Description: The ETH3.0 token contract approve functions include increaseAllowance, and decreaseAllowance. When a user authorizes using the approve function, the authorization value for the user address should be set to zero before a new authorization value is authorized. The increaseAllowance and decreaseAllowance functions correspond to increasing the authorization value of the corresponding user and decreasing the authorization value of the corresponding user respectively. It is recommended that the user use the increaseAllowance and decreaseAllowance functions for authorization when authorizing.

```
    function increaseAllowance(address spender, uint256 addedValue)
        public
        virtual
        returns (bool)
    {
        _approve(
            _msgSender(),
            spender,
            _allowances[_msgSender()][spender].add(addedValue)
        );
        return true;
    }
```

Figure 1 Source code of increaseAllowance function

```
    function decreaseAllowance(address spender, uint256 subtractedValue)
        public
        virtual
        returns (bool)
    {
        _approve(
            _msgSender(),
            spender,
            _allowances[_msgSender()][spender].sub(
                subtractedValue,
                "ERC20: decreased allowance below zero"
            )
        );
        return true;
    }
```

Figure 2 Source code of decreaseAllowance function

```solidity
    function approve(address spender, uint256 amount)
        public
        virtual
        override
        returns (bool)
    {
        _approve(_msgSender(), spender, amount);
        return true;
    }
```

Figure 3 Source code of approve function

- Related functions: increaseAllowance, decreaseAllowance, approve

(2) Transfer functions

- Description: The ETH3.0 contract token Transfer function includes Transfer a transferFrom . transfer The function is transferred directly by the user, while the transferFrom function is transferred by the agent user. The transfer user needs to provide sufficient authorization to the

```solidity
    function transferFrom(
        address sender,
        address recipient,
        uint256 amount
    ) public virtual override returns (bool) {
        _transfer(sender, recipient, amount);
        _approve(
            sender,
            _msgSender(),
            _allowances[sender][_msgSender()].sub(
                amount,
                "ERC20: transfer amount exceeds allowance"
            )
        );
        return true;
    }
```

Figure 4 Source code of transferFrom function

```
    function transfer(address recipient, uint256 amount)
        public
        virtual
        override
        returns (bool)
    {
        _transfer(_msgSender(), recipient, amount);
        return true;
    }
```

Figure 5 Source code of transfer function

- Related functions: transferFrom, transfer

(3) Privilege management functions

- Description: The renounceOwnership and transferOwnership functions are called by owner to remove owner permissions and modify owner permissions.

```
    function renounceOwnership() public virtual onlyOwner {
        _setOwner(address(0));
    }
```

Figure 6 Source code of renounceOwnership function

```
    function transferOwnership(address newOwner) public virtual onlyOwner {
        require(newOwner != address(0), "Ownable: new owner is the zero address");
        _setOwner(newOwner);
    }
```

Figure 7 Source code of transferOwnership function

- Related functions: transferOwnership, renounceOwnership

3 Appendix

3.1 Vulnerability Assessment Metrics and Status in Smart Contracts

3.1.1 Metrics

In order to objectively assess the severity level of vulnerabilities in blockchain systems, this report provides detailed assessment metrics for security vulnerabilities in smart contracts with reference to CVSS 3.1 (Common Vulnerability Scoring System Ver 3.1).

According to the severity level of vulnerability, the vulnerabilities are classified into four levels: "critical", "high", "medium" and "low". It mainly relies on the degree of impact and likelihood of exploitation of the vulnerability, supplemented by other comprehensive factors to dETHErmine of the severity level.

Impact / Likelihood	Severe	High	Medium	Low
Probable	Critical	High	Medium	Low
Possible	High	Medium	Medium	Low
Unlikely	Medium	Medium	Low	Info
Rare	Low	Low	Info	Info

3.1.2 Degree of impact

Severe

Severe impact generally refers to the vulnerability can have a serious impact on the confidentiality, integrity, availability of smart contracts or their economic model, which can cause substantial economic losses to the contract Business system, large-scale data disruption, loss of authority management, failure of key functions, loss of credibility, or indirectly affect the operation of other smart contracts associated with it and cause substantial losses, as well as other severe and mostly irreversible harm.

High

High impact generally refers to the vulnerability can have a relatively serious impact on the confidentiality, integrity, availability of the smart contract or its economic model, which can cause a greater economic loss, local functional unavailability, loss of credibility and other impact to the contract Business system.

Medium

Medium impact generally refers to the vulnerability can have a relatively minor impact on the confidentiality, integrity, availability of the smart contract or its economic model, which can cause a small amount of economic loss to the contract Business system, individual Business unavailability and other impact.

Low

Low impact generally refers to the vulnerability can have a minor impact on the smart contract, which can pose certain security threat to the contract Business system and needs to be improved.

3.1.4 Likelihood of Exploitation

Probable

Probable likelihood generally means that the cost required to exploit the vulnerability is low, with no special exploitation threshold, and the vulnerability can be triggered consistently.

Possible

Possible likelihood generally means that exploiting such vulnerability requires a certain cost, or there are certain conditions for exploitation, and the vulnerability is not easily and consistently triggered.

Unlikely

Unlikely likelihood generally means that the vulnerability requires a high cost, or the exploitation conditions are very demanding and the vulnerability is highly difficult to trigger.

Rare

Rare likelihood generally means that the vulnerability requires an extremely high cost or the conditions for exploitation are extremely difficult to achieve.

3.1.5 Fix Results Status

Status	Description
Fixed	The project party fully fixes a vulnerability.
Partially Fixed	The project party did not fully fix the issue, but only mitigated the issue.
Acknowledged	The project party confirms and chooses to ignore the issue.

3.2 Audit Categories

No.	Categories	Subitems
1	Coding Conventions	Compiler Version Security
		Deprecated Items
		Redundant Code
		require/assert Usage
		Gas Consumption
2	General Vulnerability	Integer Overflow/Underflow
		Reentrancy
		Pseudo-random Number Generator (PRNG)
		Transaction-Ordering Dependence
		DoS (Denial of Service)
		Function Call Permissions
		call/delegatecall Security
		Returned Value Security
		tx.origin Usage
		Replay Attack
		Overriding Variables
		Third-party Protocol Interface Consistency
3	Business Security	Business Logics
		Business Implementations
		Manipulable Token Price
		Centralized Asset Control
		Asset Tradability
		Arbitrage Attack

Beosin classified the security issues of smart contracts into three categories: Coding Conventions, General Vulnerability, Business Security. Their specific definitions are as follows:

Coding Conventions

Audit whether smart contracts follow recommended language security coding practices. For example, smart contracts developed in Solidity language should fix the compiler version and do not use deprecated keywords.

General Vulnerability

General Vulnerability include some common vulnerabilities that may appear in smart contract projects. These vulnerabilities are mainly related to the characteristics of the smart contract itself, such as integer overflow/underflow and denial of service attacks.

Business Security

Business security is mainly related to some issues related to the Business realized by each project, and has a relatively strong pertinence. For example, whether the lock-up plan in the code match the white paper, or the flash loan attack caused by the incorrect setting of the price acquisition oracle.

*Note that the project may suffer stake losses due to the integrated third-party protocol. This is not something Beosin can control. Business security requires the participation of the project party. The project party and users need to stay vigilant at all times.

3.3 Disclaimer

The Audit Report issued by Beosin is related to the services agreed in the relevant service agreement. The Project Party or the Served Party (hereinafter referred to as the "Served Party") can only be used within the conditions and scope agreed in the service agreement. Other third parties shall not transmit, disclose, quote, rely on or tamper with the Audit Report issued for any purpose.

The Audit Report issued by Beosin is made solely for the code, and any description, expression or wording contained therein shall not be interpreted as affirmation or confirmation of the project, nor shall any warranty or guarantee be given as to the absolute flawlessness of the code analyzed, the code team, the Business model or legal compliance.

The Audit Report issued by Beosin is only based on the code provided by the Served Party and the technology currently available to Beosin. However, due to the technical limitations of any organization, and in the event that the code provided by the Served Party is missing information, tampered with, deleted, hidden or subsequently altered, the audit report may still fail to fully enumerate all the risks.

The Audit Report issued by Beosin in no way provides investment advice on any project, nor should it be utilized as investment suggestions of any type. This report represents an extensive evaluation process designed to help our customers improve code quality while mitigating the high risks in blockchain.

3.4 About Beosin

Beosin is the first institution in the world specializing in the construction of blockchain security ecosystem. The core team members are all professors, postdocs, PhDs, and Internet elites from world-renowned academic institutions. Beosin has more than 20 years of research in formal verification technology, trusted computing, mobile security and kernel security, with overseas experience in studying and collaborating in project research at well-known universities. Through the security audit and defense deployment of more than 2,000 smart contracts, over 50 public blockchains and wallets, and nearly 100 exchanges worldwide, Beosin has accumulated rich experience in security attack and defense of the blockchain field, and has developed several security products specifically for blockchain.

BEOSIN
Blockchain Security

Official Website
https://www.beosin.com

Telegram
https://t.me/beosin

Twitter
https://twitter.com/Beosin_com

Email
service@beosin.com