

Zulu Network

Smart Contract Security Audit

No. 202409251639

Sep 25th, 2024



SECURING BLOCKCHAIN ECOSYSTEM

WWW.BEOSIN.COM

Contents

| | |
|--|-----------|
| 1 Overview | 5 |
| 1.1 Project Overview | 5 |
| 1.2 Audit Overview | 5 |
| 1.3 Audit Method | 5 |
| 2 Audit Content | 7 |
| 2.1 Daitaled Audit of Contract ERC20Permit | 7 |
| 2.2 Daitaled Audit of Contract Zulu | 8 |
| 2.3 Daitaled Audit of Contract Ownable | 9 |
| 3 Appendix | 10 |
| 3.1 Vulnerability Assessment Metrics and Status in Smart Contracts | 10 |
| 3.2 Audit Categories | 13 |
| 3.3 Disclaimer | 15 |
| 3.4 About Beosin | 16 |

Summary of Audit Results

After the audit, no risk items were identified in the Zulu Network project. Below is a detailed introduction to the project.

● Basic Token Information

| | |
|--------------|--------------|
| Token name | Zulu Network |
| Token symbol | ZULU |
| Decimals | 18 |
| Total Supply | 1 Billion |
| Token type | ERC20 |

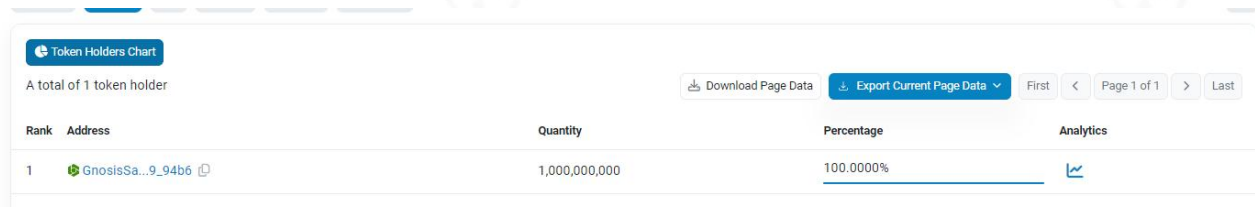
Table 1 ZULU token info

● Project Description:

Zulu Network is an ERC20 token project deployed on Ethereum with an initial supply of 1 billion tokens (burnable but not mintable). Upon contract deployment, all tokens are allocated to the owner address, and ownership rights can be transferred.

Additionally, the owner has the authority to pause transfers. When the pause state is activated, users will be unable to transfer tokens.

The owner also holds exclusive rights to burn tokens from their own account (users cannot burn tokens), making the owner the sole entity capable of burning tokens.



| Rank | Address | Quantity | Percentage | Analytics |
|------|-------------------|---------------|------------|-----------|
| 1 | GnosisSa...9_94b6 | 1,000,000,000 | 100.0000% | |

Picture 1 Current Token Allocation of ZULU

Currently, the owner has transferred all tokens to a multisig wallet for management (address: 0xd6a995CA25B2d0775E2F9418dAcF6941DC5E94b6). This multisig wallet is controlled by three managing addresses:

- Owner Address 1: 0x24a2B818dF9dD18315e7808cdccEc48F92CB1092
- Owner Address 2: 0x20AF8cFc3F8bD8AB42da8E4F6Cf2f33e4F732977
- Owner Address 3: 0x7CcA42664F4A38f8880623fbacc30b198A666644.

The signature threshold is set at 2, meaning that as long as two or more managing addresses approve, tokens in the multisig wallet can be allocated.

1 Overview

1.1 Project Overview

| | |
|------------------|--|
| Project Name | Zulu Network |
| Project Language | Solidity |
| Platform | Ethereum |
| Contract Address | 0x7b67D8B4da0B17A9B98EddC21230b60c8EDe69A4 |

1.2 Audit Overview

Audit work duration: Sep 25, 2024 – Sep 25, 2024

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 concrete test case 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 detect issues. Beosin-VaaS can detect more than 100 common smart contract vulnerabilities through static analysis, such as reentrancy and block parameter dependency. It allows early and efficient discovery of problems to improve code quality and security.

2 Audit Content

2.1 Detailed Audit of Contract ERC20Permit

(1) permit function

- Description: The `permit` function allows users to authorize a `spender` address to spend their tokens via an offline signature. In the signed data, users need to specify themselves as the `owner`, set a `deadline` (the signature must be executed before this time), and define the amount to be authorized (`value`). Additionally, the data must include a specified `nonce` (to prevent signature reuse) and `_domainSeparatorV4` (to bind chain-specific data, preventing cross-chain replay attacks). After the offline signature is generated, the parameters, along with the generated `r`, `s`, and `v` values, can be submitted by a third party for verification within the contract. Once verified, the `spender` will receive token authorization from the user.

```
function permit(
    address owner,
    address spender,
    uint256 value,
    uint256 deadline,
    uint8 v,
    bytes32 r,
    bytes32 s
) public virtual {
    if (block.timestamp > deadline) {
        revert ERC2612ExpiredSignature(deadline);
    }
}
```

2.2 Detailed Audit of Contract Zulu

(1) `pause`, `unpause`, `burn` functions

- The `pause`, `unpause`, and `burn` functions are all controlled by the `owner`. The `pause` and `unpause` functions allow the `owner` to manage token transfers between users, while the `burn` function enables the `owner` to burn tokens from their own account.

```
function pause() public onlyOwner {
    _pause();
}

function unpause() public onlyOwner {
    _unpause();
}

function burn(uint256 amount) public onlyOwner {
    _burn(msg.sender, amount);
}
```


2.3 Detailed Audit of Contract Ownable

(1) `_checkOwner` function

- Description: The `onlyOwner` modifier invokes the `_checkOwner` function for permission verification. This function checks whether the caller's address matches the `owner` address recorded in the contract. If they match, the verification is successful.

```
modifier onlyOwner() {
    _checkOwner();
    _;
}

function owner() public view virtual returns (address) {
    return _owner;
}

function _checkOwner() internal view virtual {
    if (owner() != _msgSender()) {
        revert OwnableUnauthorizedAccount(_msgSender());
    }
}
```

(2) `renounceOwnership` `transferOwnership` functions

- Description: The `renounceOwnership` function allows the `owner` to relinquish ownership rights, while the `transferOwnership` function enables the transfer of ownership rights to another user.

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

function transferOwnership(address newOwner) public virtual onlyOwner {
    {
        if (newOwner == address(0)) {
            revert OwnableInvalidOwner(address(0));
        }
        _transferOwnership(newOwner);
    }
}
```

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 determine 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.

3.1.3 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.

3.1.4 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
Web3 Security & Compliance



Official Website

<https://www.beosin.com>



Telegram

<https://t.me/beosin>



Twitter

https://twitter.com/Beosin_com



Email

service@beosin.com