SECURITY AUDIT OF

SPACEPI TOKEN

Public Report

Nov 21, 2023

Verichains Lab

info@verichains.io

https://www.verichains.io

Driving Technology > Forward

ABBREVIATIONS

Name	Description
Ethereum	An open source platform based on blockchain technology to create and distribute smart contracts and decentralized applications.
Smart contract	A computer protocol intended to digitally facilitate, verify or enforce the negotiation or performance of a contract.
Solidity	A contract-oriented, high-level language for implementing smart contracts for the Ethereum platform.
Solc	A compiler for Solidity.
ERC20	ERC20 (BEP20 in Binance Smart Chain or *x*RP20 in other chains) tokens are blockchain-based assets that have value and can be sent and received. The primary difference with the primary coin is that instead of running on their own blockchain, ERC20 tokens are issued on a network that supports smart contracts such as Ethereum or Binance Smart Chain.

EXECUTIVE SUMMARY

This Security Audit Report was prepared by Verichains Lab on Nov 21, 2023. We would like to thank the SpacePi for trusting Verichains Lab in auditing smart contracts. Delivering high-quality audits is always our top priority.

This audit focused on identifying security flaws in code and the design of the SpacePi Token. The scope of the audit is limited to the source code files provided to Verichains. Verichains Lab completed the assessment using manual, static, and dynamic analysis techniques.

During the audit process, the audit team had identified some vulnerable issues in the smart contracts code.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY **5**

1.1. About SpacePi Token **5**

1.2. Audit scope **5**

1.3. Audit methodology **5**

1.4. Disclaimer **6**

1.5. Acceptance Minute **6**

2. AUDIT RESULT **7**

2.1. Overview **7**

2.1.1. ERC20 contract 7

2.1.2. StakeLPEarnMultiToken contract 7

2.1.3. Distributor contract 7

2.2. Findings **8**

2.2.1. Users cannot withdraw their tokens if they have no rewards - MEDIUM 8

2.2.2. A SLP token implements some ERC20 functions incorrectly and unnecessarily - INFORMATIVE 9

2.2.3. An admin can set the rate to zero to disable users exchanging their vouchers - INFORMATIVE 10

3. VERSION HISTORY **11**

1. MANAGEMENT SUMMARY

1.1. About SpacePi Token

As a blockchain game supported by the concept of NFT + Metaverse, SpacePi includes several core sections in terms of platform ecological construction. NFT game system Metaverse game ecology SpacePi Exchange

1.2. Audit scope

This audit focused on identifying security flaws in code and the design of SpacePi Token. It was conducted on commit `f29aa53297f4bfeb8f33b87a03ed24189486ecb7` from git repository link: *https://github.com/SpacePiCom/Contracts*

The latest version of the following files was made available in the course of the review:

SHA1 Sum	File
`95d0cae054ab6322f5d56835ed60b1699a39c2f4`	`StakeLpEarnMultiToken.sol`

1.3. Audit methodology

Our security audit process for smart contract includes two steps:

- Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using public and RK87, our in-house smart contract security analysis tool.
- Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.

Following is the list of commonly known vulnerabilities that were considered during the audit of the smart contract:

- Integer Overflow and Underflow
- Timestamp Dependence
- Race Conditions
- Transaction-Ordering Dependence
- DoS with (Unexpected) revert
- DoS with Block Gas Limit
- Gas Usage, Gas Limit and Loops
- Redundant fallback function
- Unsafe type Inference
- Reentrancy
- Explicit visibility of functions state variables (external, internal, private and public)

- Logic Flaws

For vulnerabilities, we categorize the findings into categories as listed in table below, depending on their severity level:

SEVERITY LEVEL	DESCRIPTION
CRITICAL	A vulnerability that can disrupt the contract functioning; creates a critical risk to the contract; required to be fixed immediately.
HIGH	A vulnerability that could affect the desired outcome of executing the contract with high impact; needs to be fixed with high priority.
MEDIUM	A vulnerability that could affect the desired outcome of executing the contract with medium impact in a specific scenario; needs to be fixed.
LOW	An issue that does not have a significant impact, can be considered as less important.

Table 1. Severity levels

1.4. Disclaimer

SpacePi acknowledges that the security services provided by Verichains, are conducted to the best of their professional abilities but cannot guarantee 100% coverage of all security vulnerabilities. SpacePi understands and accepts that despite rigorous auditing, certain vulnerabilities may remain undetected. Therefore, SpacePi agrees that Verichains shall not be held responsible or liable, and shall not be charged for any hacking incidents that occur due to security vulnerabilities not identified during the audit process.

1.5. Acceptance Minute

This final report served by Verichains to the SpacePi will be considered an Acceptance Minute. Within 7 days, if no any further responses or reports is received from the SpacePi, the final report will be considered fully accepted by the SpacePi without the signature.

2. AUDIT RESULT

2.1. Overview

The SpacePi Token was written in the `Solidity` programming language. The code was compiled with `Solc` compiler version `0.8.4` and the resulting bytecode was deployed on the `Ethereum` blockchain. The source code was written based on OpenZeppelin's library.

2.1.1. ERC20 contract

A `stakeLP` (also known as SLP) is an ERC20 token that is minted under the `StakeLPEarnMultiToken` contract when a user stakes an LP token. Although the contract implements the `IERC20`, it is solely used to mint the `SLP` token and does not utilize any other features of the ERC20 token.

2.1.2. StakeLPEarnMultiToken contract

The `StakeLPEarnMultiToken` contract extends the `ERC20` contract, the `Ownable` contract and the `ReentrancyGuard` contract.

The `Ownable` contract is used to manage the owner of the contract. The `ReentrancyGuard` contract is used to prevent reentrancy attacks.

This contract serves the purpose of depositing, withdrawing and earning an LP token. The LP token is defined during the contract deployment.

Users will receive rewards after a certain period of time following their LP token deposit. Which each block number increase, the reward also increase. Once the specified block number is reached, the user will receive `80%` of the `138,888,888,888,888,900` LP tokens divided by the total pool amount, calculated as follows:

```
perSecond = 138888888888888900;
userReward = userAmount * perSecond / totalAmountOfPool;
```

The contract also allows users to refer others. A referred user will receive 20% of the reward from a depositing user.

2.1.3. Distributor contract

The `Distributor` contract extends the `Ownable` contract and defines an exchange feature that enables users to exchange their vouchers for ERC20 tokens. The vouchers are distributed by the `StakeLPEarnMultiToken` contract.

The `Distributor` contract allows an admin to set the rate of the voucher to the ERC20 token. This rate is used to calculate the amount of ERC20 tokens that a user can exchange from their voucher.

Additionally, the owner of this contract who is the `StakeLPEarnMultiToken` contract, can withdraw all the balance tokens of this contract by calling the `withdraw` function.

2.2. Findings

During the audit process, the audit team found some vulnerabilities in the given version of the SpacePi Token.

#	Issue	Severity	Status
1	Users cannot withdraw their tokens if they have no rewards	MEDIUM	Acknowledged
2	A SLP token implements some ERC20 functions incorrectly and unnecessarily	INFORMATIVE	Acknowledged
3	An admin can set the rate to zero to disable users exchanging their vouchers	INFORMATIVE	Acknowledged

2.2.1. Users cannot withdraw their tokens if they have no rewards - MEDIUM

Affected contract: StakeLPEarnMultiToken

The `withdraw` function allows a user to withdraw their LP tokens and rewards if the user has rewards. If the user has not reward, the `withdraw` function will revert the transaction.

The scenario occurs when a user deposits and withdraws their LP tokens in a short period, possibly within the same block number. The user will not receive rewards if they withdraw their LP tokens before the next block number because rewards are calculated for each block number. In this case, the user cannot withdraw their LP tokens.

Below is the code snippet for the `withdraw` function:

```
function withdraw(uint256 _amount) external nonReentrant {
    address ua = msg.sender;
    UserInfo storage user = users[ua];
    require(user.amount >= _amount, "not enough amount");
    updatePool();
    if (user.amount > 0) {
        referral(ua);
    }
    // ...
}
function referral(address user) private {
    UserInfo storage u = users[user];
    uint256 accPerShare = pool.accPerShare;
    uint256 pendingAmount = (u.amount * accPerShare / ACCURACY) - u.rewardDebt;
    require(pendingAmount > 0, "not enough reward");
    // ...
}
```

RECOMMENDATION

The withdraw function should allow a user to withdraw their LP tokens if the user has no rewards.

```
function referral(address user) private {
    UserInfo storage u = users[user];
    uint256 accPerShare = pool.accPerShare;
    uint256 pendingAmount = (u.amount * accPerShare / ACCURACY) - u.rewardDebt;
    if (pendingAmount == 0) return;
    // ...
}
```

UPDATES

Nov 21, 2023: The SpacePi team has been acknowledged the issue.

2.2.2. A SLP token implements some ERC20 functions incorrectly and unnecessarily - INFORMATIVE

Affected contract: ERC20

The SLP token allows the StakeLPEarnMultiToken contract to mint the SLP token and the SLP token does not use any other features.

However, the allowance and approve functions always return maximum and true, which is incorrect and does not follow the ERC20 standard. Since the token does not implement a transfer feature, the allowance and approve functions are unnecessary.

```
contract ERC20 is IERC20 {
    //...
    function allowance(address, address) external pure returns (uint256) {
        return type(uint256).max;
    }
    function approve(address, uint256) external pure returns (bool){
        return true;
    }
    function transferFrom(address sender, address recipient, uint256 amount) external
returns (bool){
         // Missing calculate allowance
        _transfer(sender, recipient, amount);
        return true;
    }
    function _transfer(address from, address recipient, uint256 amount) private {
         // Does not implement a transfer feature
        emit Transfer(from, recipient, amount);
    }
}
```

RECOMMENDATION

Remove the `_transfer`, `transfer`, `transferFrom`, `allowance` and `approve` functions.

UPDATES

Nov 21, 2023: The SpacePi team has been acknowledged the issue.

2.2.3. An admin can set the rate to zero to disable users exchanging their vouchers - INFORMATIVE

Affected contract: Distributor

The `setRate` function allows an admin to set the rate of a voucher and a token. The rate is used to calculate the amount of the voucher that a user can exchange for tokens.

However, the `setRate` function currently permits an admin to set the rate to zero. When the rate is set to zero, users will be unable to exchange their vouchers for tokens.

```
function setRate(uint256 _rate) public onlyOwner{
    rate = _rate;
}
```

RECOMMENDATION

The `setRate` function should be updated to disallow an admin from setting the rate to zero. Instead, it should allow an admin to specify the rate within a defined range that includes minimum and maximum values.

UPDATES

Nov 21, 2023: The SpacePi team has acknowledged the issue.

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Oct 30, 2023*	Private Report	Verichains Lab
1.1	*Nov 21, 2023*	Public Report	Verichains Lab

Table 2. Report versions history