verichains

SECURITY AUDIT OF

ANCIENT8 STAKING

Public Report

Jul 05, 2024

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.
Ether (ETH)	A cryptocurrency whose blockchain is generated by the Ethereum platform. Ether is used for payment of transactions and computing services in the Ethereum network.
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 Jul 05, 2024. We would like to thank the Ancient8 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 Ancient8 Staking. 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 a vulnerable issue in the smart contracts code. The issue is fixed and the code is updated.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About Ancient8 Staking

Ancient8 is the gaming L2 on ethereum backed by Dragonfly, Pantera, Hashed, Coinbase Ventures and Makers Fund.

Core products:

- Ancient8 Chain: OP Stack Ethereum L2, Built on Optimism Superchain.
- Space3: Gaming Users Acquisition Engine.
- Reneverse: In-game and IP Asset Interoperability Protocol

1.2. Audit scope

This audit focused on identifying security flaws in code and the design of the Ancient8 Staking. It was conducted on commit `70479e54b25496a64f5c4bcb0a4488e3fbb744db` from git repository link *https://github.com/ancient8-gg/a8-staking-contracts*.

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

SHA256 Sum	File
`1687f139925448c0bff5bab1331c99b1abfa85f60ed13cab56861d1786819660`	`contracts/token-staking/TokenStakingControllable.sol`
`25ae7538549fa0f1c88e90564ece7eaa252ab6cac0c4a4ca03e8bc296e1b4a50`	`contracts/token-staking/TokenStakingCore.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

Ancient8 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. Ancient8 understands and accepts that despite rigorous auditing, certain vulnerabilities may remain undetected. Therefore, Ancient8 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 Ancient8 will be considered an Acceptance Minute. Within 7 days, if no any further responses or reports is received from the Ancient8, the final report will be considered fully accepted by the Ancient8 without the signature.

2. AUDIT RESULT

2.1. Overview

The Ancient8 Staking was written in `Solidity` language, with the deployed compiler version is `0.8.19`.

The contract extends the `AccessControl` contract from the `OpenZeppelin` library. The `AccessControl` contract is used to manage roles and permissions in the contract.

2.1.1. Roles

- `DEFAULT_ADMIN_ROLE`: Has the highest level of access and can assign other roles.
- `CONTROLLER_ROLE`: Can stake tokens on behalf of users that have approved amounts.

2.1.2. Business Logic

- **Staking**: allows users to stake tokens by specifying an amount and a lock-up configuration index. This function calls the internal _stake function, which performs several checks via the canStake modifier. These checks ensure the lock-up configuration is valid, the apocalypse date hasn't passed, the user hasn't reached the maximum active sections, and the amount is greater than zero. If all checks pass, the contract transfers the specified amount of tokens from the user to the contract and creates a new section to record the stake details.
- **Unstaking**: enables users to unstake tokens by providing the section ID they wish to unstake. This function calls the internal _unstake function, which performs checks via the canUnstake modifier. These checks verify the section ID is valid, the section hasn't been unstaked already, and the lock-up period has ended. If the checks are successful, the contract updates the section's unstaked date, removes the section ID from the user's active sections, and transfers the staked amount of tokens back to the user.

2.2. Findings

During the audit process, the audit team found a vulnerability in the given version of Ancient8 Staking.

2.2.1. MEDIUM - Unsafe Transfer Token

Positions:

- contracts/token-staking/TokenStakingCore.sol#L198
- contracts/token-staking/TokenStakingCore.sol#L255
- contracts/token-staking/TokenStakingControllable.sol#74

Description:

The contract uses the `transfer` and `transferfrom` functions to transfer in/out tokens. If the token does not revert when the transfer fails (e.g., due to insufficient balance) and only return a true/false value, the contract will not handle the failure and continue execution. This can lead to unexpected behavior and loss of funds.

```
function _stake(
    uint256 amount,
    uint256 lockUpConfigIdx
) internal canStake(amount, lockUpConfigIdx) returns (uint256) {
    IERC20(tokenAddress).transferFrom(msg.sender, address(this), amount);
    return
        _createSection(
        //...
        );
}
```

RECOMMENDATION

Use the `SafeERC20` library to handle token transfers. The `SafeERC20` library provides a set of functions that prevent the contract from continuing execution if the token transfer fails.

UPDATES

- ***Jul 05, 2024***: The issue has been acknowledged and fixed by Ancient8 team.

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Jul 05, 2024*	Public Report	Verichains Lab

Table 2. Report versions history