verichains

SECURITY AUDIT OF

LANDAUCTION SMART CONTRACT

Public Report

Mar 06, 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 Mar 06, 2024. We would like to thank the Super Sushi Samurai 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 LandAuction Smart Contract. 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 one vulnerable issue in the smart contracts code.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About LandAuction Smart Contract

Super Sushi Samurai is a social strategy focused idle game, played on the telegram app and powered by the Blast network. It's fully on-chain.

The LandAuction is a core part of the game, where players can buy land.

1.2. Audit scope

This audit focused on identifying security flaws in code and the design of the LandAuction Smart Contract.

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

SHA256 Sum	File
fe2d55856137f65734fbef71eb3035e63028b22e12fc9e348cd7d4908150a135	LandAuction.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

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

2. AUDIT RESULT

2.1. Overview

The LandAuction Smart Contract was written in `Solidity` language, with the required version to be `^0.8.20`.

The contract extends the `Ownable` contract from the `openzeppelin/contracts` library. By default, the owner of the contract is the deployer. The contract enables users to bid native tokens for land NFTs, with the price decreasing sequentially over time until the auction concludes or the land is sold. Once the auction ends, users can claim the number NFTs based on the bid amount divided by the last recorded price.

2.2. Findings

During the audit process, the audit team found one vulnerability issue in the given version of LandAuction Smart Contract.

2.2.1. Rounding error while calculating the number NFT to mint HIGH

By default, division in Solidity results in flooring. Consequently, the outcome of 2/3 is 0, and 1/3 yields 0 as well. This poses an issue when determining the number of NFTs to mint, as the division's reduction can result in fewer NFTs being minted compared to the totalNFTSell set by the owner. Consequently, this shortfall can lead to minting fewer NFTs than the owner intends to sell. In the current implementation, an attacker can stake various amounts across multiple accounts to minimize the actual number of NFTs minted, facilitating manipulation of market prices.

For example, with the default configuration where totalNFTSell equals 555 and totalBidAmount equals 123 ether, suppose there are three bidders with the following bid amounts: 3 ether, 40 ether, and 80 ether. With the current implementation, the number of NFTs minted for each bidder would be 13, 180, and 360, respectively (13.53, 180.48, 360.97 if not floored). The total NFTs minted would amount to 553, falling short of the totalNFTSell target of 555.

```
function mintNFT() nonReentrant external {
        // check if sold out
        uint256 soldOutPrice = clearPrice;

        // Check if auction is finished
        if(soldOutPrice == 0) {
            uint256 currentPrice = getCurrentPrice();
            uint256 totalNFTSold = totalBidAmount/currentPrice;
            if(totalNFTSold < totalNFTSell) {
                revert("Auction: Not sold out");
```

```
            } else {
                soldOutPrice = totalBidAmount/totalNFTSell; // @Verichains: the
soldOutPrice is calculated based on totalBidAmount and totalNFTSell
                clearPrice = soldOutPrice;
                uint256 steps = (startPrice - soldOutPrice) / stepPrice;
                endBlock = startBlock + steps * blockDelta;
            }
        }

        // calculate amount of NFT can mint
        uint256 amountOfNFTCanMint = bidders[msg.sender] / soldOutPrice; // @Verichains:
ammountOfNFTcantMint is floored for each user.
        uint256 mintedAmount = bidderNFTMintedAmounts[msg.sender];
        if(amountOfNFTCanMint == 0) {
            revert("Auction: Not enough SSS to mint NFT");
        }

        ...

        // limit to 10 NFTs per tx to prevent out of gas
        amountOfNFTCanMint = amountOfNFTCanMint - mintedAmount;
        amountOfNFTCanMint = amountOfNFTCanMint > 10?10:amountOfNFTCanMint;

        bidderNFTMintedAmounts[msg.sender] = mintedAmount + amountOfNFTCanMint;

        // mint NFT
        landDistributor.mintBatch(msg.sender, amountOfNFTCanMint);
    }
```

UPDATES

- **Mar 06, 2024**: The issue has been acknowledged by the Super Sushi Samurai team.

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Mar 06, 2024*	Public Report	Verichains Lab

Table 2. Report versions history