SECURITY AUDIT OF

ELEMON SMART CONTRACTS

Public Report

Dec 08, 2021

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 prepared by Verichains Lab on Dec 08, 2021. We would like to thank the Elemon 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 Elemon Smart Contracts. 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 no vulnerable issues in the application, along with some recommendations.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About Elemon Smart Contracts

Elemon is an innovative game project that combines NFT technology with the new generation IDLE RPG trend. It is the IDLE mechanism of the game that will be the bright spot to help NFT gamers always stay in a leisurely state to earn a lot of money without having to plug in for hours in front of the screen, hang up overnight and damage the device. With just a few simple steps, the player's squad will automatically fight, even when the player is offline, the player will still receive the usual rewards. Possessing a compelling storyline with thousands of diverse NFT characters, the world of Elemon will open a meaningful journey of discovery and combat.

1.2. Audit scope

This audit focused on identifying security flaws in code and the design of the smart contracts of Elemon Smart Contracts. It was conducted on the source code provided by the Elemon team.

It was conducted on commit ab3523f4a17e25b9fb899545b57d2e511edb6444 from git repository *https://github.com/elemongame/elemon-contracts/*

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 was 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

Please note that security auditing cannot uncover all existing vulnerabilities, and even an audit in which no vulnerabilities are found is not a guarantee for a 100% secure smart contract. However, auditing allows discovering vulnerabilities that were unobserved, overlooked during development and areas where additional security measures are necessary.

2. AUDIT RESULT

2.1. Overview

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

SHA256 SUM	FILE
3c7897de810850a0a5cc21198c44b36582202cf7d67a576e2a89896f58799e6f	ElemonDistributor.sol
5055737c7f7c9b486444b1c8730a8b19700ffeb4ebe08aea7dc83d257623b829	ElemonGetingNFTStakingInitializer.sol
b06da57e140fac3644d2660300ddc0df259700b61940bb08b7e9f79a7f8f0344	ElemonInfo.sol
b98919ebc891a49e5a62ef7cc7765495c9849ba810ec21d0534ccdb584603b53	ElemonMysteryBox.sol
1944b5a3553f2ca5fb16624dad31b034f7e6525760c5599ef44aff93fac5b810	ElemonMysteryBoxNFT.sol
909ee5383b00b8b6059ebed903b392b04a2273c4faa9deba8e2e6923d45b8d0f	ElemonNFT.sol
68c09e01306b8089c7699f90f7a661e92892b9da5b1776df538818e47a52af34	ElemonStakingInitializer.sol
d7e30ebdcc22948caf3fcfc9e3577deea1f0726292468d8b3014312a8b974f26	ElemonSummon.sol
d69412dabfc0dbb9d486a38be1f0c847bd25882aa39b215cf312d3ee3a251864	ERC165.sol
fbf13168f101f3803f97ecaffe51f181f34f461cd86077f868a2376a136938b9	Context.sol
8be096aed597f7efd40774024a65ea833453fe45713cc2bc70a49ad23a671e42	Ownable.sol

SHA256 SUM	FILE
3914172e119a4beed08dcd1e7cba51ae397d60082bedbaf e7644affc0fb11e2e	ReentrancyGuard.sol
44071440519f4c615a7ed7e02b9bd69ecbdc90a0a794d3c 760ba66ff07623520	Runnable.sol

2.2. Findings

The Elemon Smart Contracts was written in Solidity language, with the required version to be ^0.8.9. The source code was written based on OpenZeppelin's library.

This section contains a detailed analysis of all the vulnerabilities that were discovered by the audit team during the audit process.

2.2.1. ElemonGetingNFTStakingInitializer.sol - Unlogical minting conditions on withdraw INFORMATIVE

withdraw function will mint an elemon once for callers who staked longer than _stakingDuration (userInfo.lastStakingTime + _stakingDuration <= block.timestamp):

```
if(userInfo.elemonTokenId == 0 && userInfo.lastStakingTime + _stakingDura…
  tion <= block.timestamp){
    //Mint Elemon for user
   userInfo.elemonTokenId = _elemonNFT.mint(_msgSender());

   //Set Elemon info with rarity
   _elemonInfo.setRarity(userInfo.elemonTokenId, _rarity);

   emit ElemonDistributed(_msgSender(), userInfo.elemonTokenId);
}
```

This minting logic is unfair for ones who stake many times and for ones who stake high total amount. Current contract's source code does not contain any document for this behavior. Audit team can not sure if the final behavior will be provided to end users or not so that they can make their staking strategics correctly.

RECOMMENDATION

Possible method maybe that staking will accumulate and user will be credited using staking token, and they can exchange staking token into elemon, but it's complex.

2.2.2. ElemonStakingInitializer.sol - claimVestingReward optimize INFORMATIVE

Current signature of claimVestingReward is:

```
claimVestingReward(uint256 count)
```

As vestingRewards is storage variable, reading its element's unlockedTime field still cost gas, there's still possibility that there is not enough gas to finish the operation, or that user is needed to pay unrequired gas for already claimed rewards.

RECOMMENDATION

Add another function claimVestingReward(uint256 from, uint256 count).

2.2.3. ElemonStakingInitializer.sol - Invalid revert message INFORMATIVE

The revert message in setDuration is wrong.

```
function setDuration(uint256 duration) external onlyOwner{
    require(duration > 0, "Zero address");
    _stakingDuration = duration;
}
```

RECOMMENDATION

Change the revert message to duration must be greater than zero.

```
function setDuration(uint256 duration) external onlyOwner{
    require(duration > 0, "duration must be greater than zero");
    _stakingDuration = duration;
}
```

2.2.4. ElemonStakingInitializer.sol - Use immutable for SMART_CHEF_FACTORY INFORMATIVE

Variable SMART_CHEF_FACTORY is only set in constructor and then use for reading so we recommend using immutable instead of storage variable (The value will be stored directly in the code thus saving gas and avoid mistake assign).

```
address public SMART_CHEF_FACTORY;
```

RECOMMENDATION

Change SMART_CHEF_FACTORY to immutable.

```
address public immutable SMART_CHEF_FACTORY;
```

2.2.5. ElemonSummon.sol - Use calldata instead of memory for gas saving INFORMATIVE

In external function with array arguments, using memory will force solidity to copy that array to memory thus wasting more gas than using directly from calldata. Unless you want to write to the variable, always using calldata for external function.

```
function setRarities(uint256[] memory rarities) external onlyOwner
function setRarityAbilities(uint256 level, uint256[] memory rarities, uin…
  t256[] memory abilities) external onlyOwner
function setBaseCardIds(uint256 level, uint256[] memory baseCardIds) exte…
  rnal onlyOwner
...
```

RECOMMENDATION

Change memory to calldata for gas saving in all external functions.

```
function setRarities(uint256[] calldata rarities) external onlyOwner
function setRarityAbilities(uint256 level, uint256[] calldata rarities, u…
  int256[] calldata abilities) external onlyOwner
function setBaseCardIds(uint256 level, uint256[] calldata baseCardIds) ex…
  ternal onlyOwner
...
```

2.2.6. Typo in _recepient INFORMATIVE

There are some typo in recepient, the correct should be recipient.

RECOMMENDATION

Fix the typo.

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Dec 07, 2021*	Private Report	Verichains Lab
1.1	*Dec 08, 2021*	Private Report	Verichains Lab

Table 2. Report versions history