SECURITY AUDIT OF

SLIME-ROYALE TOKEN SMART CONTRACT

Public Report

Jul 07, 2022

Verichains Lab

info@verichains.io

https://www.verichains.io

Driving Technology > Forward

ABBREVIATIONS

Name	Description
BSC	Binance Smart Chain or BSC is an innovative solution for introducing interoperability and programmability on Binance Chain.
BNB	A cryptocurrency whose blockchain is generated by the Binance Smart Chain platform. BNB is used for payment of transactions and computing services in the Binance Smart Chain 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 Jul 07, 2022. We would like to thank the Slime-Royale 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 Slime-Royale Token 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 no vulnerable issue** in the smart contracts code.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About Slime-Royale Token Smart Contract

Slime Royale is not a Gamefi project, they are a Mobile game with blockchain technology. GameFi is usually regarded as money-oriented projects that majorly focus on the financial factors, while the gaming gets set aside. The core of Slime Royale, in contrast, is formed around designing an engaging gameplay and then applying blockchain to resolve the payment problem and attract new players.

In short, Slime Royale leverages blockchain technology to create a different business model in the Mobile Game industry, allowing players to both enjoy the game and earn money from their skills.

SRG token is an ERC20 token which Slime Royale players use in the game.

1.2. Audit scope

This audit focused on identifying security flaws in code and the design of the Slime-Royale Token Smart Contract. It was conducted on commit `b033fcbaa4b40239e8f2ae182fc57d716a3adff1` from git repository *https://github.com/Slime-Royale/token-contract*.

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

SHA256 Sum	File
`ed0e525b6c3f905cf59bf9ecedb6c0a49b1f9bdf51108ea80d616a6c9ce22b6e`	`SRG.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 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 Slime-Royale Token Smart Contract was written in `Solidity` language, with the required version to be `0.8.0`.

2.1.1. SRG token contract

The contract extends `AccessControl`, `ERC20` and `ERC20Snapshot` contracts. With `AccessControl`, the contract creates the `OWNER_ROLE` which may control the `snapshot` function and `BPContract` settings. The `ERC20Snapshot` contract helps `OWNER_ROLE` take a snapshot of the balances and total supply at a time for later access.

PROPERTY	VALUE
Name	Slime Royale Gold
Symbol	SRG
Decimals	18
Total Supply	1,000,000,000 (x10^{18}) Note: the number of decimals is 18, so the total representation token will be 1,000,000,000 or 1 billion.

Table 2. The SRG contract properties

2.2. Findings

During the audit process, the audit team **found no vulnerability** in the given version of Slime-Royale Token Smart Contract.

2.3. Additional notes and recommendations

2.3.1. BPContract function INFORMATIVE

The contract uses `BPContract` to control the `transfer` action. Since we do not control the logic of the `BPContract`, there is no guarantee that `BPContract` will not contain any security related issues. With the current context, in case the `BPContract` is compromised, there is not yet a way to exploit the Slime-Royale Token Smart Contract, but we still note that here as a warning for avoiding any related issue in the future.

By the way, if having any issue, the `BPContract` function can be easily disabled anytime by the `OWNER_ROLE` user using the `togglePreventBotMode` function.

2.3.2. Best-practice in `togglePreventBotMode` function INFORMATIVE

The contract includes the `togglePreventBotMode` function.

```
function togglePreventBotMode() public onlyRole(OWNER_ROLE) {
        isInPreventBotMode = !isInPreventBotMode;
    }
```

The `togglePreventBotMode` is used to toggle the `isInPreventBotMode` state variable.

With current logic, the `OWNER_ROLE` could not exactly control the next value, if the `OWNER_ROLE` didn't check the previous `isInPreventBotMode` value. With a mistake, the transaction is sent twice, the value will be idempotent.

The `togglePreventBotMode` will be more convenient if the `OWNER_ROLE` passes the exact value.

APPENDIX

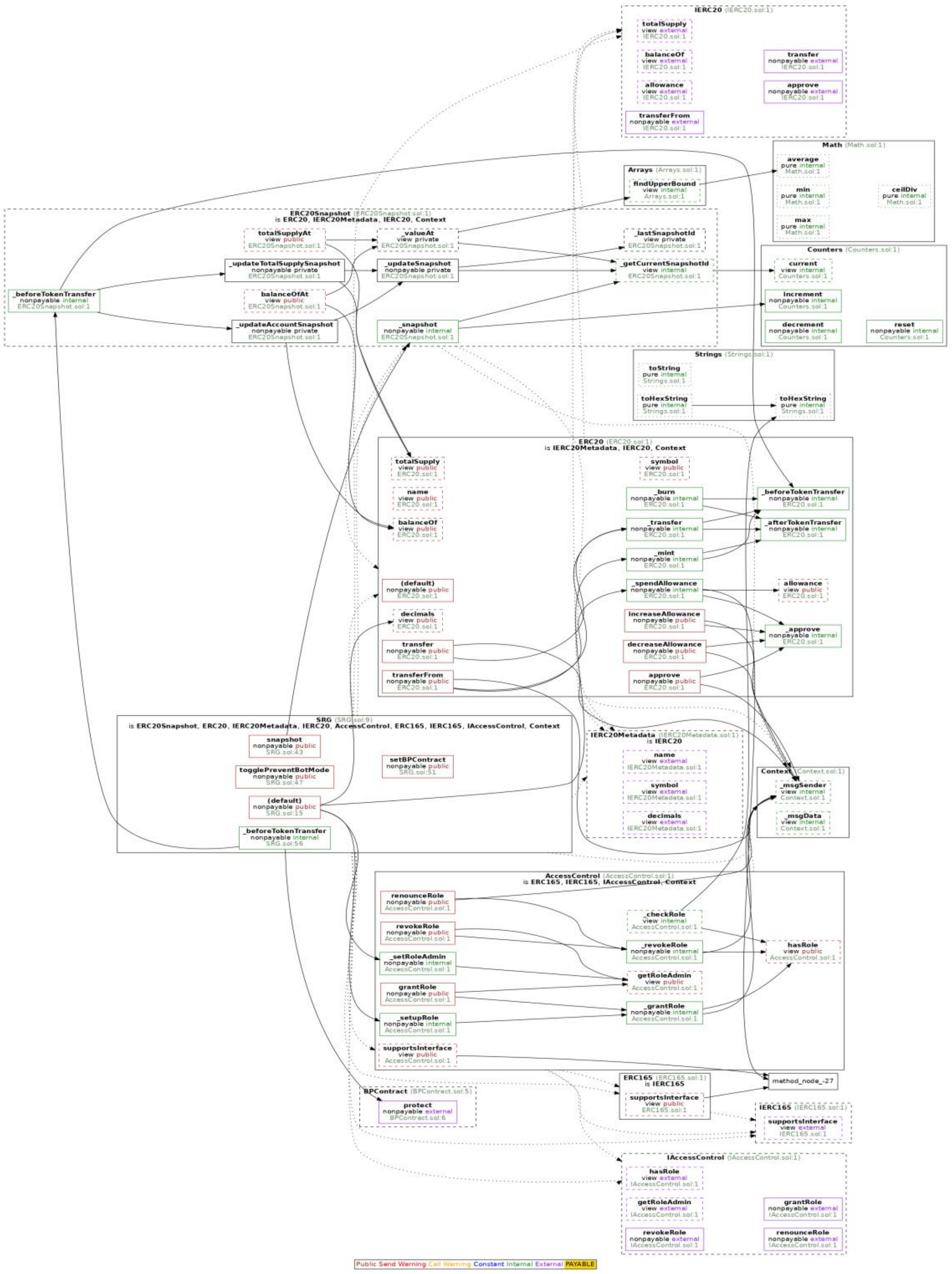

Image 1. SRG contract call graph

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Jun 28, 2022*	Public Report	Verichains Lab
1.1	*Jul 07, 2022*	Public Report	Verichains Lab

Table 3. Report versions history