verichains

SECURITY AUDIT OF

STMAN TOKEN SMART CONTRACTS

Public Report

Mar 21, 2022

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 07, 2022. We would like to thank the STMAN 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 STMAN Token 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 contract code.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About STMAN Token Smart Contracts

Anti-inflation Stickman's Battleground is an NFT game of survival with a free-to-play-to-earn mechanism.

1.2. Audit scope

This audit focused on identifying security flaws in code and the design of the ERC20 token and ERC721 token of the STMAN Token Smart Contracts. It was conducted on the source code provided by the STMAN team.

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

SHA256 Sum	File
d0563bf69a4dfa7f2a7379fcaa8f94aa17b8f481d1f993b4a840bf7b7ef97472	STMAN.SOL
912970e9ae697323b5d3a035f91455dfb63cd6e4a4bc9d9c23d879f399d0dae5	Hero.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 STMAN Token Smart Contracts was written in Solidity language.

2.1.1. STManToken contract

This table lists some properties of the audited STManToken (as of the report writing time).

PROPERTY	VALUE
Name	STICKMAN BATTLEGROUND
Symbol	STMan
Decimals	18
Max Supply	230,000,000 ($x10^{18}$) Note: the number of decimals is 18, so the total representation token will be 230,000,000.

Table 2. The STManToken properties

STManToken requires solidity version to be 0.6.12. The contract extends BEP20 and Ownable contracts. With Ownable, by default, Token Owner is contract deployer but he can transfer ownership to another address at any time.

The contract also supports bot protection by BPContract, which will make limit on buy/sell, add/remove blacklist, whitelist addresses and prevent sandwich attack. The owner can enable/disable bot protection at any time.

The owner can use mint function to increase total supply but it cannot beyond the hardcap.

2.1.2. Hero contract

Hero contract requires solidity version to be <=0.8.10. This is the NFT contract in the STMAN Token Smart Contracts, which extends ERC721Enumerable, AccessControlEnumerable, Ownable and IHero contract. With Ownable, by default, Token Owner is contract deployer, but he can transfer ownership to another address at any time.

The owner or the MINTER_ROLE role can mint new NFT by using mint function. Also, the owner can change the BaseURL by setBaseUrl function.

2.2. Findings

During the audit process, the audit team found no vulnerability in the given version of the STMAN Token Smart Contracts.

APPENDIX

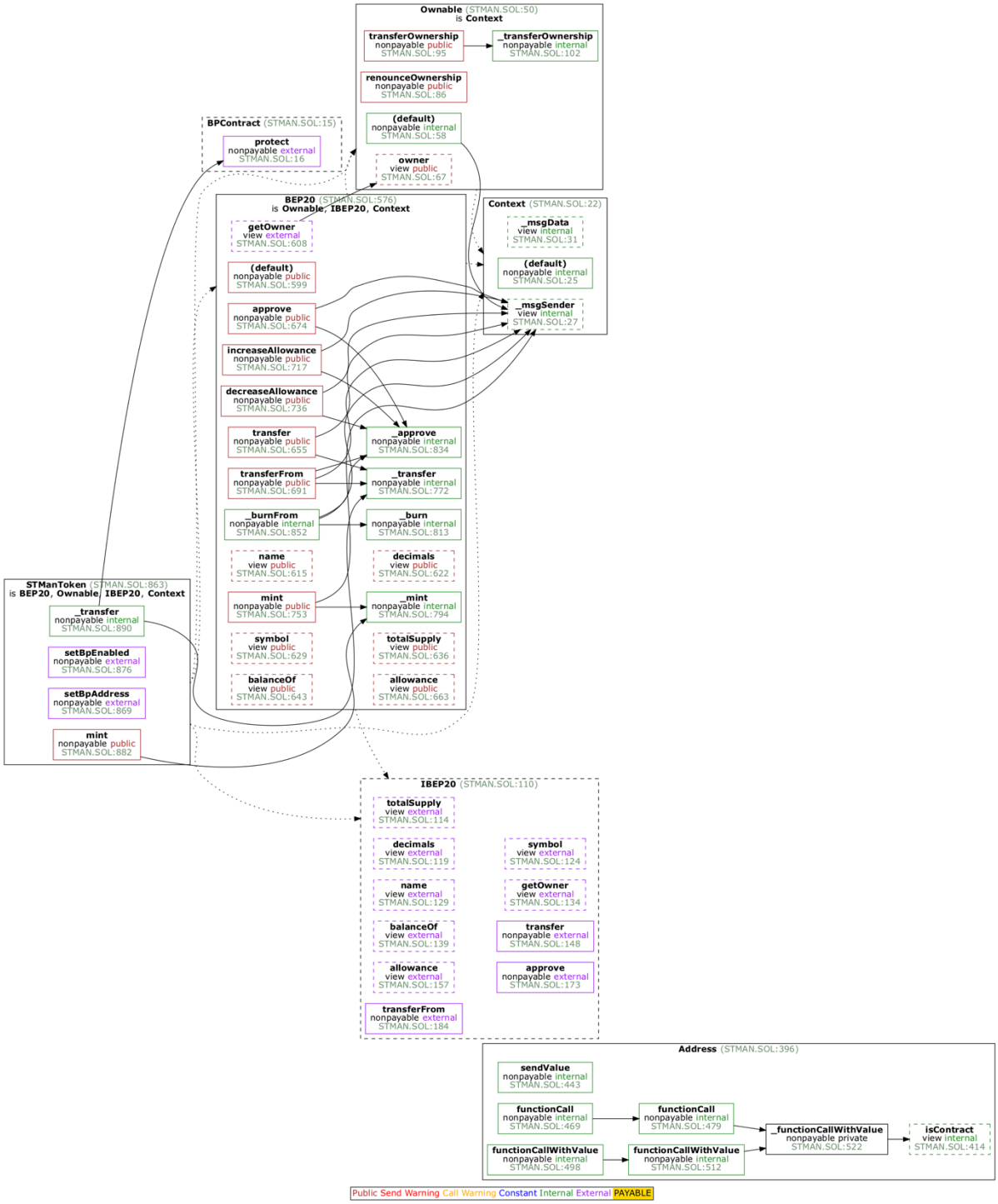

Image 1. STMAN Token call graph

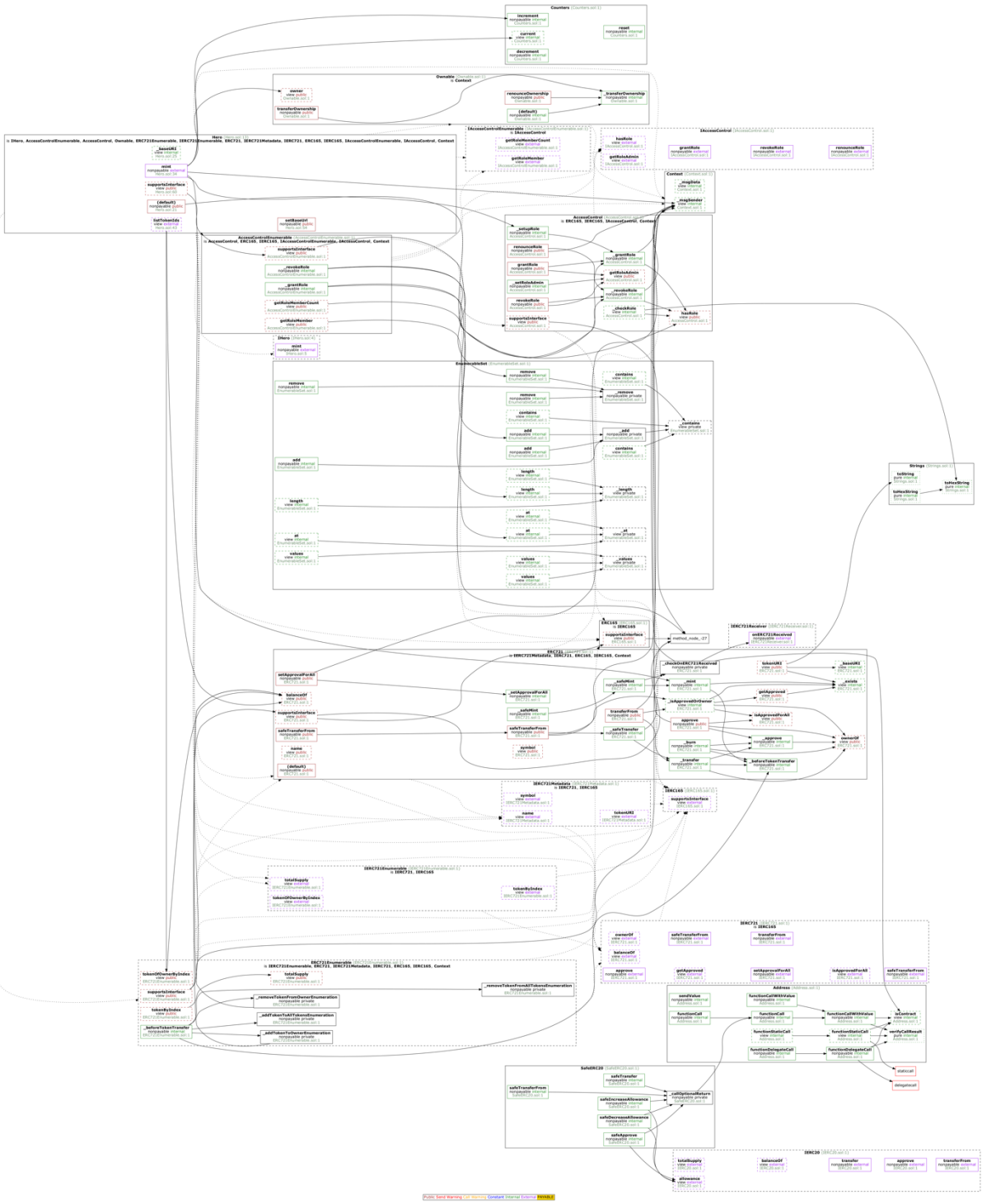

Image 2. Hero call graph

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Feb 11, 2022*	Public Report	Verichains Lab
1.1	*Feb 11, 2022*	Public Report	Verichains Lab
1.2	*Mar 07, 2022*	Public Report	Verichains Lab
1.3	*Mar 21, 2022*	Public Report	Verichains Lab

Table 3. Report versions history