SECURITY AUDIT OF

STMAN VAULT AND BOXVAULT SMART CONTRACT

Public Report

Apr 15, 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 prepared by Verichains Lab on Apr 15, 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 Vault and BoxVault 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 issues in the contract code.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About STMAN Vault and BoxVault Smart Contract

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 STMAN Vault and BoxVault Smart Contract. 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
a8aec3b5587391392e7687fc5917f500e1fb2df394125435496dfcb9c60c7305	**BoxVault.sol**
f3381f00a7656db59d0e52b5b7886c226a27d2b915d84b79d1a084b4bceeaa44	**Vault.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 Vault and BoxVault Smart Contract was written in Solidity language, with the required version to be ^0.8.10

2.1.1. BoxVault contract

The contract allows users to deposit tokens following the strategy of the STMAN team. All the deposited tokens may be withdrawn by the owner or the WITHDRAWER_ROLE user.

2.1.2. Vault contract

The contract has logic similar to the BoxVault contract which is mentioned above. The only difference is the deposit function of the Vault contract doesn't include the emit statement.

2.2. Findings

During the audit process, the audit team found no vulnerability in the given version of STMAN Vault and BoxVault Smart Contract.

APPENDIX

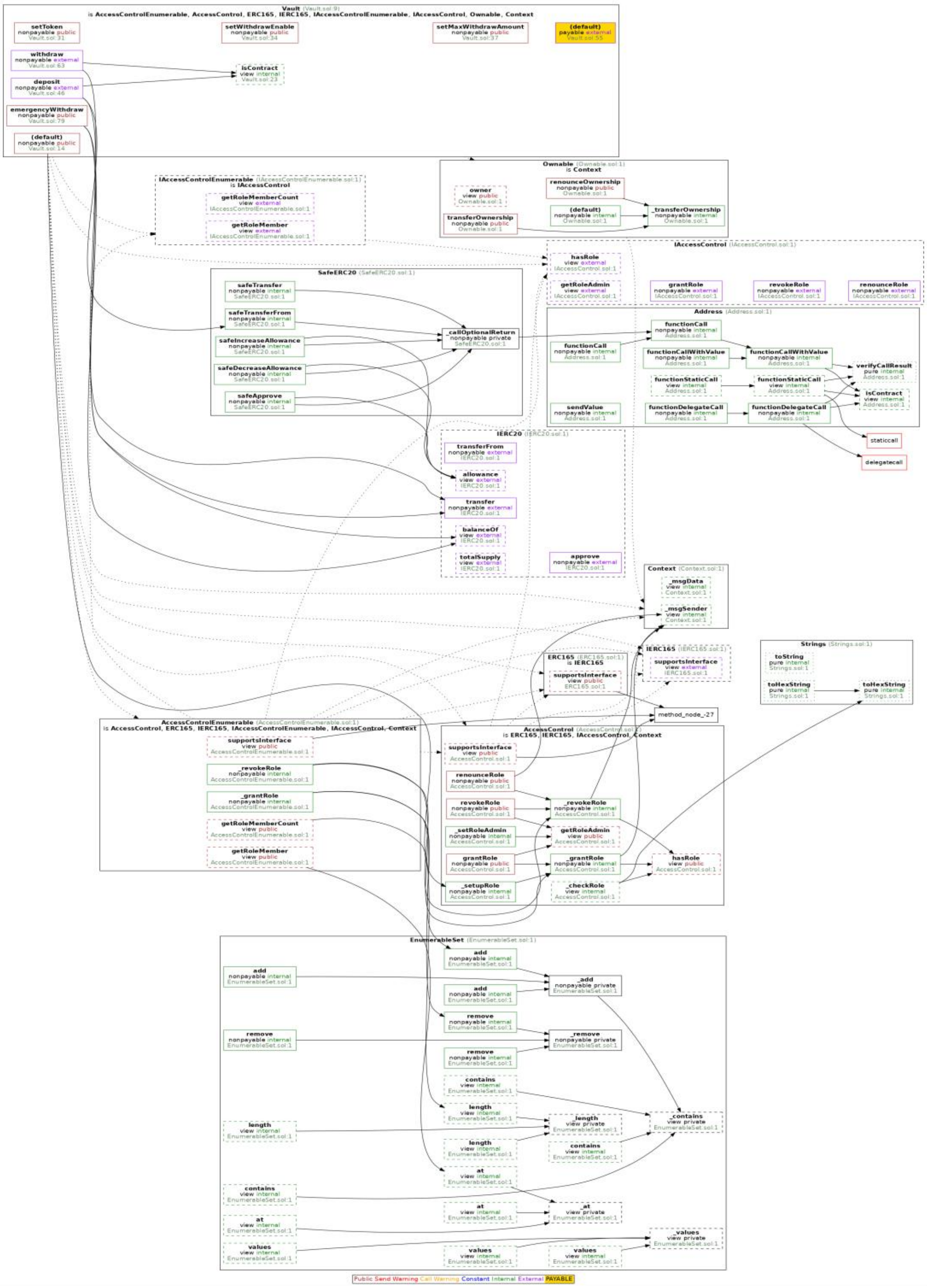
Vault (Vault.sol:9)
is AccessControlEnumerable, AccessControl, ERC165, IERC165, IAccessControlEnumerable, IAccessControl, Ownable, Context
setToken nonpayable public Vault.sol:31
setWithdrawEnable nonpayable public Vault.sol:34
setMaxWithdrawAmount nonpayable public Vault.sol:37
(default) payable external Vault.sol:55
withdraw nonpayable external Vault.sol:63
isContract view internal Vault.sol:23
deposit nonpayable external Vault.sol:46
emergencyWithdraw nonpayable public Vault.sol:79
(default) nonpayable public Vault.sol:14
Ownable (Ownable.sol:1) is Context
owner view public Ownable.sol:1
renounceOwnership nonpayable public Ownable.sol:1
(default) nonpayable internal Ownable.sol:1
transferOwnership nonpayable public Ownable.sol:1
_transferOwnership nonpayable internal Ownable.sol:1
IAccessControlEnumerable (IAccessControlEnumerable.sol:1) is IAccessControl
getRoleMemberCount view external IAccessControlEnumerable.sol:1
getRoleMember view external IAccessControlEnumerable.sol:1
IAccessControl (IAccessControl.sol:1)
hasRole view external IAccessControl.sol:1
getRoleAdmin view external IAccessControl.sol:1
grantRole nonpayable external IAccessControl.sol:1
revokeRole nonpayable external IAccessControl.sol:1
renounceRole nonpayable external IAccessControl.sol:1
SafeERC20 (SafeERC20.sol:1)
safeTransfer nonpayable internal SafeERC20.sol:1
safeTransferFrom nonpayable internal SafeERC20.sol:1
safeIncreaseAllowance nonpayable internal SafeERC20.sol:1
safeDecreaseAllowance nonpayable internal SafeERC20.sol:1
safeApprove nonpayable internal SafeERC20.sol:1
_callOptionalReturn nonpayable private SafeERC20.sol:1
Address (Address.sol:1)
functionCall nonpayable internal Address.sol:1
functionCall nonpayable internal Address.sol:1
functionCallWithValue nonpayable internal Address.sol:1
functionCallWithValue nonpayable internal Address.sol:1
verifyCallResult pure internal Address.sol:1
functionStaticCall view internal Address.sol:1
functionStaticCall view internal Address.sol:1
isContract view internal Address.sol:1
sendValue nonpayable internal Address.sol:1
functionDelegateCall nonpayable internal Address.sol:1
functionDelegateCall nonpayable internal Address.sol:1
staticcall
delegatecall
IERC20 (IERC20.sol:1)
transferFrom nonpayable external IERC20.sol:1
allowance view external IERC20.sol:1
transfer nonpayable external IERC20.sol:1
balanceOf view external IERC20.sol:1
totalSupply view external IERC20.sol:1
approve nonpayable external IERC20.sol:1
Context (Context.sol:1)
_msgData view internal Context.sol:1
_msgSender view internal Context.sol:1
Strings (Strings.sol:1)
toString pure internal Strings.sol:1
toHexString pure internal Strings.sol:1
toHexString pure internal Strings.sol:1
ERC165 (ERC165.sol:1) is IERC165
supportsInterface view public ERC165.sol:1
IERC165 (IERC165.sol:1)
supportsInterface view external IERC165.sol:1
method_node_-27
AccessControlEnumerable (AccessControlEnumerable.sol:1)
is AccessControl, ERC165, IERC165, IAccessControlEnumerable, IAccessControl, Context
supportsInterface view public AccessControlEnumerable.sol:1
_revokeRole nonpayable internal AccessControlEnumerable.sol:1
_grantRole nonpayable internal AccessControlEnumerable.sol:1
getRoleMemberCount view public AccessControlEnumerable.sol:1
getRoleMember view public AccessControlEnumerable.sol:1
AccessControl (AccessControl.sol:1)
is ERC165, IERC165, IAccessControl, Context
supportsInterface view public AccessControl.sol:1
renounceRole nonpayable public AccessControl.sol:1
revokeRole nonpayable public AccessControl.sol:1
_revokeRole nonpayable internal AccessControl.sol:1
_setRoleAdmin nonpayable internal AccessControl.sol:1
getRoleAdmin view public AccessControl.sol:1
grantRole nonpayable public AccessControl.sol:1
_grantRole nonpayable internal AccessControl.sol:1
hasRole view public AccessControl.sol:1
_setupRole nonpayable internal AccessControl.sol:1
_checkRole view internal AccessControl.sol:1
EnumerableSet (EnumerableSet.sol:1)
add nonpayable internal EnumerableSet.sol:1
add nonpayable internal EnumerableSet.sol:1
add nonpayable internal EnumerableSet.sol:1
_add nonpayable private EnumerableSet.sol:1
remove nonpayable internal EnumerableSet.sol:1
remove nonpayable internal EnumerableSet.sol:1
remove nonpayable internal EnumerableSet.sol:1
_remove nonpayable private EnumerableSet.sol:1
contains view internal EnumerableSet.sol:1
length view internal EnumerableSet.sol:1
length view internal EnumerableSet.sol:1
_length view private EnumerableSet.sol:1
_contains view private EnumerableSet.sol:1
length view internal EnumerableSet.sol:1
contains view internal EnumerableSet.sol:1
at view internal EnumerableSet.sol:1
contains view internal EnumerableSet.sol:1
at view internal EnumerableSet.sol:1
_at view private EnumerableSet.sol:1
at view internal EnumerableSet.sol:1
_values view private EnumerableSet.sol:1
values view internal EnumerableSet.sol:1
values view internal EnumerableSet.sol:1
values view internal EnumerableSet.sol:1
Public Send Warning Call Warning Constant Internal External PAYABLE

Image 1. Vault call graph

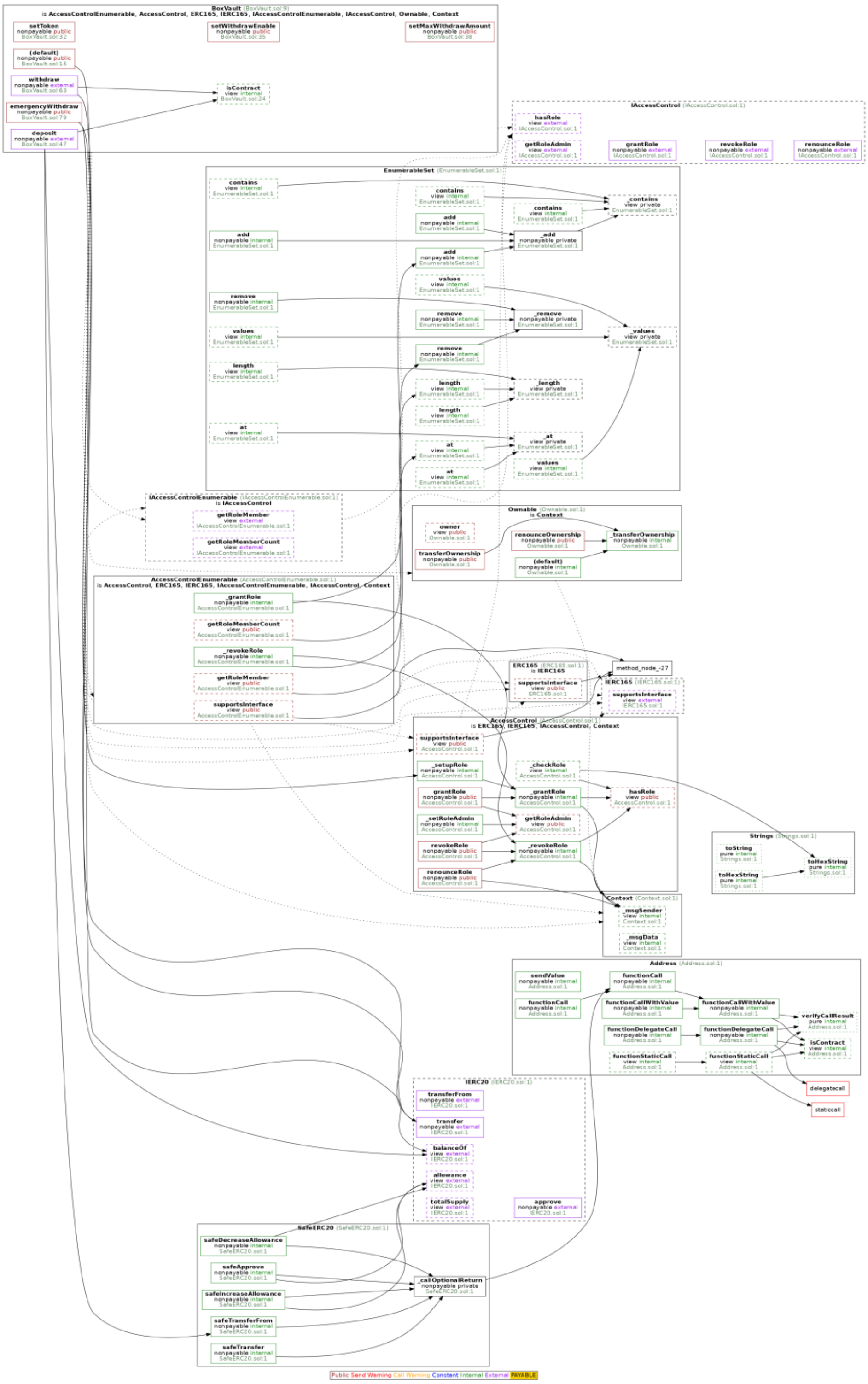

Image 2. BoxVault call graph

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Apr 15, 2022*	Public Report	Verichains Lab

Table 2. Report versions history