SECURITY AUDIT OF

ETHE KING SMART CONTRACTS

Public Report

Jan 17, 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 Jan 17, 2021. We would like to thank the ETHE King 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 ETHE King 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 issue in the smart contracts code.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About ETHE King Smart Contracts

EHTE King is a game metaverse based on the NFT platform and uses ETHE which is a BSC-based Cryptocurrencies. The first ETHE King Saga has two phases: Future Kings and Clash of Kings(COK)

1.2. Audit scope

This audit focused on identifying security flaws in code and the design of ETHE King Smart Contracts. It was conducted on commit 83b3b9b2283a85ecd05e4c6486c612cc39e460a0 from git repository *https://github.com/NS8888/EtheToken/*.

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

SHA256 SUM	FILE
d7964315929578101a60b512aedb1b6f0567d5d29eebfd23fe661147c62f1e41	Address.sol
d57a87486b2a47da36c0e094aa079d83bd6660ef2d17daa41647b229156461c6	ERC165.sol
45ae0e676100c23e75b39603bc7d8ce4d6e64868993ab50262039bdc54103e8c	BEP20.sol
e9075996c3560f2607b8c2af7fdadd3172a4d100a93856eff2f350d64c23e190	BEP20Detailed.sol
365dfec5dab662f369923aa27950dce681c20bd5eb74e2020a8c775df8450bf1	BEPContext.sol
195e71a92a844951fa48d39758069989ee3259b067f917de220024002acf7587	BEPOwnable.sol
3769353b5680ec57e949d9493797a273a3be770b5b22148807361bad10cab3ec	BEPPausable.sol
7a4ea73cd98185f37872c309f177b77d8013878ea8cee7a3aca5667ea7487c8f	ERC721.sol
9e98a870e40ff543266fdb0ed3a04ebdd441c3ec9811e502740c36fff4533a51	ERC721Enumerable.sol

SHA256 SUM	FILE
7085e05419f6bac3537c5d4514f9f7e1a0cef80522d36156f13e2cdaba8f42ce	ETHEToken.sol
15e595496b6268faf1a7b92f3107a9cbcdfbb54e092fc8634597a5b6c9635224	IBEP20.sol
846cb969f4b27d05d43e69809f851d5c3643361addd93624b3bed805b7037aaa	LANDNFT.sol
5d14192adfc4920220985461b73476cb75fc61461ff185deb52c62f68c91fade	MRTNFT.sol
48de099057597f26778a3815a18c267f37189d71e4446644f6c57387fa9e57e4	Lock-All-ETHE-And-Vesting.sol
71c37232113f52433042d788efd366ceaecf78d412f009b8a88d776cb6934646	SafeERC20.sol
a4e43f384ea6a1578301d10e3b695dcfc989df173ebbe59b0bfe2e506923f521	IERC20.sol
004f2aa969c0563bf991cbeb424574d19245e9d18a846c0e8c0126071828ca4f	Math.sol
92762629f91532d937e795ceee7391d5e4e9db0ca8eba233da3dd1e95ce9d792	IERC165.sol
f7e53102ef90c09c9ccd8ecf0b750c1093b50b5abbb32b8eb66762d1f3e654b8	IERC721.sol
aa10a56cb89f948a75ca6aeadcb7b63b88176500d701e7561d55a675dc46ad91	IERC721Enumerable.sol
49ad003da954541241dad9a32166fa75bcffbbfb360c5d3bc5a4fd36ed5ef0a8	IERC721Metadata.sol
c56e2d94c65a19174c1b4bc6c943e504c9f8aa42d0e0ee739b8be932909e9799	IERC721Receiver.sol
c6337888a0819d44219a7828c96df4b5b7116bb5e7508ab079238948e3cd7b40	SafeMath.sol

SHA256 SUM	FILE
5258e2f47ca2c91476eef42adea79c9d72448374d58b6db87711718bbe0c9c55	Strings.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.

SEVERITY LEVEL	DESCRIPTION
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 initial review was conducted on Jan 2022 and a total effort of 3 working days was dedicated to identifying and documenting security issues in the code base of the ETHE King Smart Contracts.

2.1.1. Contract code

The ETHE King Smart Contracts was written in Solidity language, with the required version to be ^0.8.0. The source code was written based on OpenZeppelin's library.

2.1.2. IBEP20 interface

The interface of the BEP20 standard as defined in the EIP.

2.1.3. BEP20 contract

The base contract which implements all functions of the IBEP20 interface.

2.1.4. BEP20Detailed abstract contract

The abstract contract which implements view functions allows users to get some metadata of the BEP20 contract.

2.1.5. BEPContext contract

Provides information about the current execution context, including the sender of the transaction and its data.

2.1.6. ERC165 contract

Creates a standard method to publish and detect what interfaces a smart contract implements.

2.1.7. IERC165 interface

Interface of the ERC165 standard.

2.1.8. BEPPausable abstract contract

The contract module allows children to implement an emergency stop mechanism that can be triggered by an authorized account.

2.1.9. SafeERC20 library

Wrappers around ERC20 operations that throw on failure (when the token contract returns false). Tokens that return no value (and instead revert or throw on failure) are also supported, non-reverting calls are assumed to be successful.

2.1.10. BEPOwnable abstract contract

Contract module which provides a basic access control mechanism, where there is an account (an owner) that can be granted exclusive access to specific functions.

2.1.11. ERC721 contract

This is the contract that implements ERC271 which was defined in IERC721 and IERC721Metadata interface. The contract inherits BEPPausable so the owner can pause or unpause the activities of the contract.

2.1.12. ERC721Enumerable abstract contract

The abstract contract extends ERC271 to tracking the ERC271 token through the execution. The contract inherits ERC271 contract so the contract also inherits BEPPausable.

2.1.13. ETHEToken contract

ETHEToken contract is a BEP20 token contract. The contract implements theburn external function so the users can burn their tokens which can change the totalSupply of the contract.

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

PROPERTY	VALUE
Name	EtheKing
Symbol	ETHE
Decimals	18
Total Supply	100,000,000 (x10^{18}) Note: the number of decimals is 18, so the total representation token will be 100,000,000 or 100 million.

Table 2. The ETHE King Smart Contracts properties

2.1.14. LANDNFT contract

LANDNFT contract is an ERC721 contract. The contract inherits the ERC721Enumerable abstract contract so the contract also inherits all of the parent ERC721Enumerable including the BEPPausable contract. The owner of the contract can pause or unpause the activities of the contract. The owner also creates new tokens through the mint public function which affects to the totalSupply value.

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

PROPERTY	VALUE
Name	Moon Land ETHE
Symbol	MoonLand
baseURI	*https://etheking.io/api/nft/moonland/*

2.1.15. MRTNFT contract

MRTNFT contract is an ERC721 contract. The contract inherits the ERC721Enumerable abstract contract so the contract also inherits all of the parent ERC721Enumerable including the BEPPausable contract. The owner of the contract can pause or unpause the activities of the contract. The owner also creates new tokens through the mint public function which affects to the totalSupply value.

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

PROPERTY	VALUE
Name	Moon Rocket Ticket
Symbol	MoonTicket
baseURI	*https://etheking.io/api/nft/rocketticket/*

2.1.16. VestingWallet contract

This contract handles the vesting of Eth and ERC20 tokens for a given beneficiary. Custody of multiple tokens can be given to this contract, which will release the token to the beneficiary following a given vesting schedule. The contract does not allow anyone to claim the tokens early than expected including the owner of contract.

2.2. Findings

During the audit process, the audit team found no vulnerability issue in the given version of ETHE King Smart Contracts.

2.3. Additional notes and recommendations

2.3.1. BEP20.sol - Unnecessary usage of SafeMath library in Solidity 0.8.0+ INFORMATIVE

All safe math usage in the contract are for overflow checking, solidity 0.8.0+ already do that by default, the only usage of safemath now is to have a custom revert message which isn't the case in the auditing contracts. We suggest using normal operators for readability and gas saving.

RECOMMENDATION

We suggest changing all methods from SafeMath library to the normal arithmetic operator in the contract and removing the SafeMath import statement.

UPDATES

- *Jan 17, 2021*: This issue has been acknowledged by the ETHE King team.

2.3.2. Lock-All-ETHE-And-Vesting.sol - Unused release internal function INFORMATIVE

Currently, the contract does not have any logic to use the release internal function. There is no function calls it and this function is also set internal so the normal users can't call it.

```
function release() internal virtual {
        uint256 releasable = vestedAmount(uint64(block.timestamp)) - …
released();
        _released += releasable;
        emit EtherReleased(releasable);
        Address.sendValue(payable(beneficiary()), releasable);
    }
```

Snippet 1. Lock-All-ETHE-And-Vesting.sol - Unused `release` internal function

RECOMMENDATION

We suggest removing this function for readability.

UPDATES

- *Jan 17, 2021*: This issue has been acknowledged by the ETHE King team.

APPENDIX

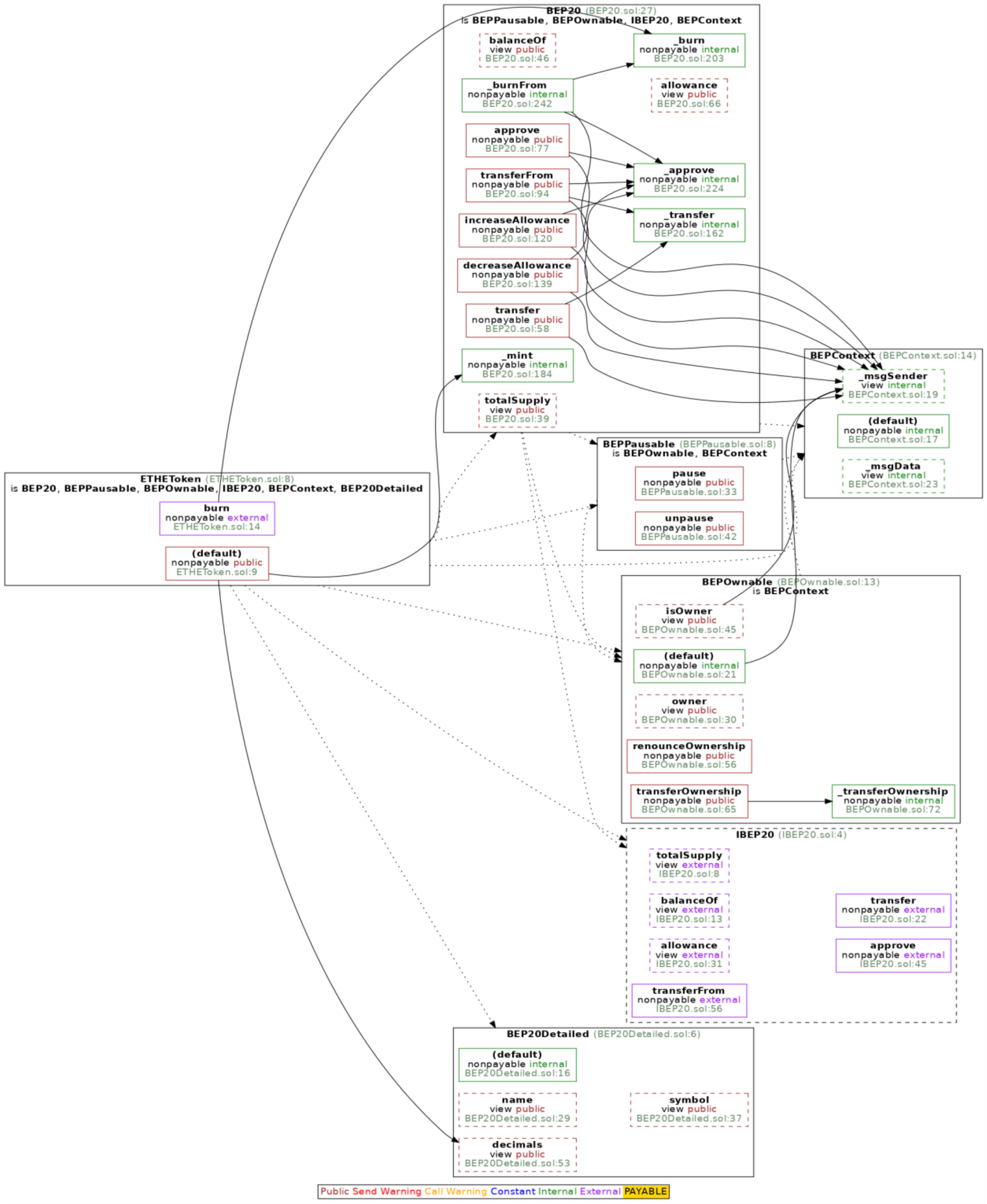
BEP20 (BEP20.sol:27)
is BEPPausable, BEPOwnable, IBEP20, BEPContext
balanceOf view public BEP20.sol:46
_burn nonpayable internal BEP20.sol:203
_burnFrom nonpayable internal BEP20.sol:242
allowance view public BEP20.sol:66
approve nonpayable public BEP20.sol:77
_approve nonpayable internal BEP20.sol:224
transferFrom nonpayable public BEP20.sol:94
_transfer nonpayable internal BEP20.sol:162
increaseAllowance nonpayable public BEP20.sol:120
decreaseAllowance nonpayable public BEP20.sol:139
transfer nonpayable public BEP20.sol:58
_mint nonpayable internal BEP20.sol:184
totalSupply view public BEP20.sol:39
BEPContext (BEPContext.sol:14)
_msgSender view internal BEPContext.sol:19
(default) nonpayable internal BEPContext.sol:17
_msgData view internal BEPContext.sol:23
BEPPausable (BEPPausable.sol:8)
is BEPOwnable, BEPContext
pause nonpayable public BEPPausable.sol:33
unpause nonpayable public BEPPausable.sol:42
ETHEToken (ETHEToken.sol:8)
is BEP20, BEPPausable, BEPOwnable, IBEP20, BEPContext, BEP20Detailed
burn nonpayable external ETHEToken.sol:14
(default) nonpayable public ETHEToken.sol:9
BEPOwnable (BEPOwnable.sol:13)
is BEPContext
isOwner view public BEPOwnable.sol:45
(default) nonpayable internal BEPOwnable.sol:21
owner view public BEPOwnable.sol:30
renounceOwnership nonpayable public BEPOwnable.sol:56
transferOwnership nonpayable public BEPOwnable.sol:65
_transferOwnership nonpayable internal BEPOwnable.sol:72
IBEP20 (IBEP20.sol:4)
totalSupply view external IBEP20.sol:8
balanceOf view external IBEP20.sol:13
transfer nonpayable external IBEP20.sol:22
allowance view external IBEP20.sol:31
approve nonpayable external IBEP20.sol:45
transferFrom nonpayable external IBEP20.sol:56
BEP20Detailed (BEP20Detailed.sol:6)
(default) nonpayable internal BEP20Detailed.sol:16
name view public BEP20Detailed.sol:29
symbol view public BEP20Detailed.sol:37
decimals view public BEP20Detailed.sol:53
Public Send Warning Call Warning Constant Internal External PAYABLE

Image 1. ETHE token Smart contracts call graph

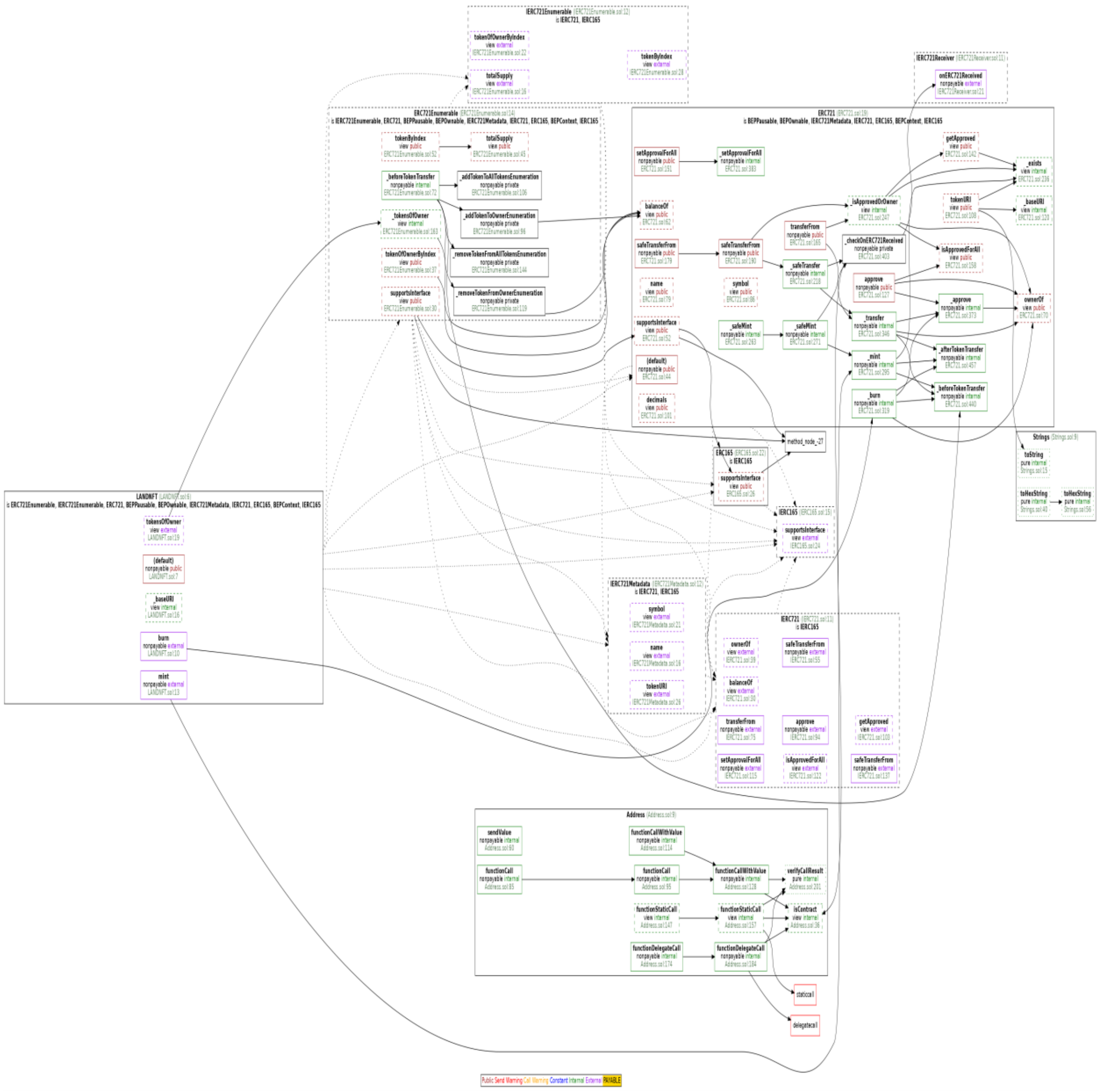

Image 2. LANDNFT Smart contracts call graph

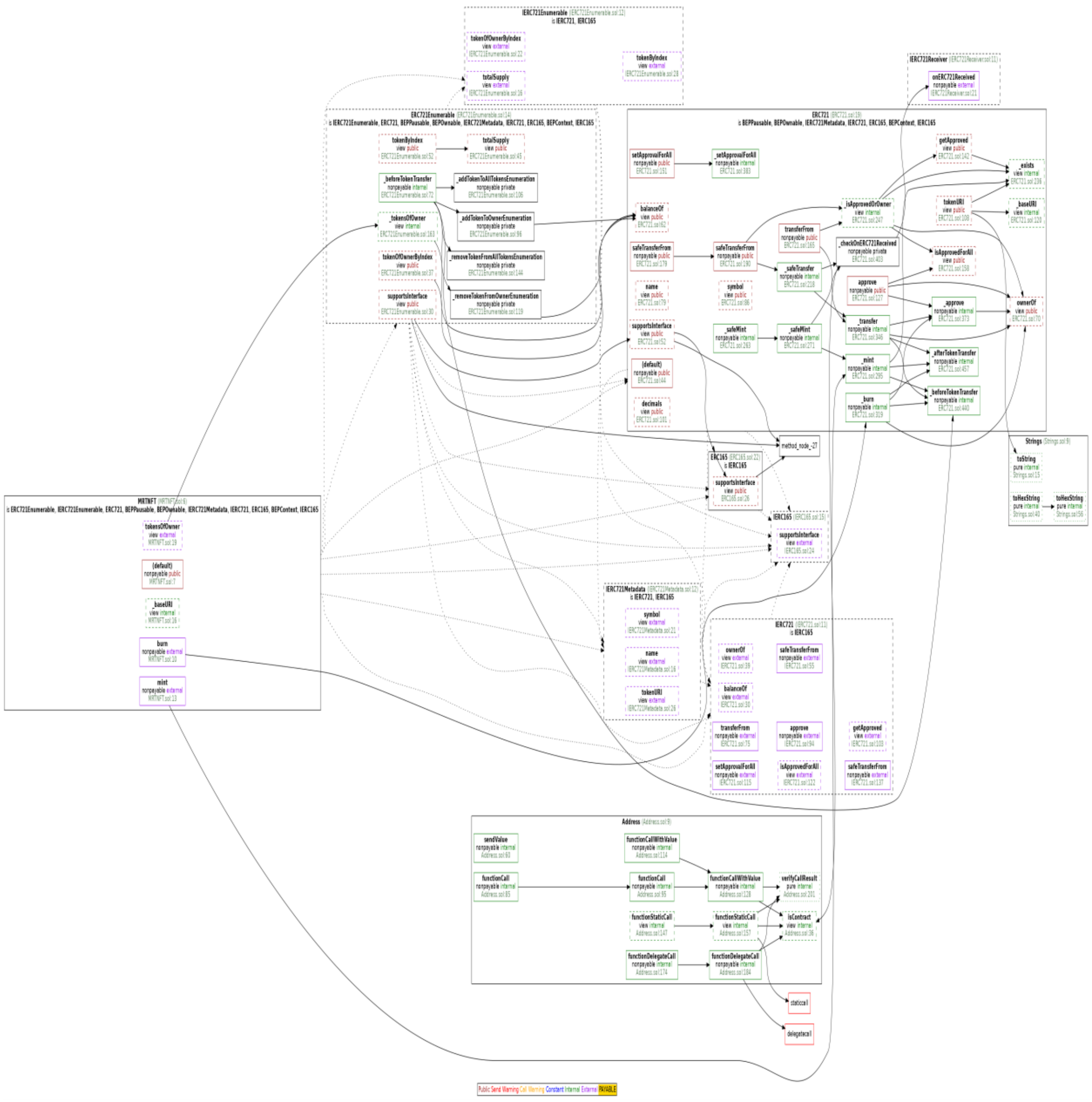

Image 3. MRTNFT Smart contracts call graph

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Jan 14, 2021*	Public Report	Verichains Lab
1.1	*Jan 17, 2021*	Public Report	Verichains Lab

Table 3. Report versions history