verichains

SECURITY AUDIT OF

MERCURIAL

TOKEN CONTRACT

PUBLIC REPORT

MAY 19, 2021

Verichains Lab

info@verichains.io

https://www.verichains.io

Driving Technology > Forward

ACRONYMS AND 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.

EXECUTIVE SUMMARY

This Security Audit Report prepared by Verichains Lab on May 19, 2021. We would like to thank the Mercurial team for trusting Verichains Lab in audit 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 Mercurial Token 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.

The assessment did not identify any vulnerability issue in Mercurial Token smart contract code.

Overall, the code reviewed is of good quality, written with the awareness of smart contract development best practices.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About Mercurial and Mercurial Token

Mercurial (available at https://www.mercurial.finance) is building DeFi's first dynamic vaults for stable assets, providing the technical tools for users to easily deposit and mint stable assets, generating liquidity for their own requirements or offering them to ecosystem participants with such demands.

Mercurial Token (MER) is the ERC-20 token of Mercurial, with an initial total supply of 50 million.

1.2. Audit scope

This audit focused on identifying security flaws in code and the design of the Mercurial Token Contract.

The audited contract is the Mercurial Token Contract that was deployed on Ethereum Mainnet at address *0x825ec55a67dF0c9DfbFC37Af6C32c6C0A9C07d64*. The details of the deployed smart contract are listed in Table 1.

FIELD	VALUE
Contract Name	Mercurial
Contract Address	0x825ec55a67dF0c9DfbFC37Af6C32c6C0A9C07d64
Complier Version	v0.8.1+commit.df193b15
Optimization Enabled	No with 200 runs
Explorer	https://etherscan.io/address/0x825ec55a67df0c9dfbfc37af6c32c6c0a9c07d64

Table 1: The deployed smart contract details

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 2, 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 2: Vulnerability 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 Mercurial Token Contract is an ERC-20 Token Contract[1], which implements a standard interface for token as defined in EIP-20[2]. This standard provides basic functionality to transfer tokens, as well as allow tokens to be approved so they can be spent by another on-chain third party.

Table 3 lists some properties of the audited Mercurial Token Contract (as of the report writing time).

PROPERTY	VALUE
Name	Mercurial
Symbol	MER
Decimals	6
Owner	0xabe6ecefcf2455c7189024e471484382261bd734 (Mercurial: Deployer)
Total Supply	50.000.000.000.000 (5×10^{13}) Note: the number of decimals is 6, so the total representation tokens will be 5×10^{7}, or 50 million.

Table 3: The Mercurial Token properties

Besides the standard interface of an ERC-20 token, the Mercurial Token Contract also implements some additional functional logics, that make the token contract:

- Ownable: the contract has a basic access control mechanism, where there is an account (an owner) that can be granted exclusive access to specific functions. By default, the owner account will be the one that deploys the contract. That owner can be changed later.
- Mintable: the contract can create a number of new tokens and assigns them to an account, increasing the total token supply.
- Burnable: the token holders can destroy both their own tokens and those that they have an allowance for, in a way that can be recognized off-chain (via event analysis).
- Pausable: the contract has an emergency stop mechanism that can be triggered by an authorized account (which is the contract owner is this case). In the Mercurial Token Contract, when the contract is paused, all the token transfers will be stopped, includes minting and burning.

[1] https://ethereum.org/en/developers/docs/standards/tokens/erc-20
[2] https://eips.ethereum.org/EIPS/eip-20

2.2. Contract codes

The Mercurial Token Contract was written in Solidity language[3], with the minimum required version is `0.8.0`.

The source codes consist of two contracts, four abstract contracts and two interfaces. Almost all source codes in the Mercurial Token Contract are imported from OpenZeppelin's implementation of ERC20-related contracts[4].

2.2.1. IERC20 interface

This is the interface of ERC-20 standard as defined in EIP-20. The source code is stored in the `IERC20.sol` file, which is identical to OpenZeppelin's implementation at https://github.com/OpenZeppelin/openzeppelin-contracts/blob/c3ae4790c7/contracts/token/ERC20/IERC20.sol.

2.2.2. IERC20Metadata interface

This is the interface for the optional metadata functions from the ERC-20 standard. The source code is stored in the `IERC20Metadata.sol` file, which is identical to OpenZeppelin's implementation at https://github.com/OpenZeppelin/openzeppelin-contracts/blob/c3ae4790c7/contracts/token/ERC20/extensions/IERC20Metadata.sol.

2.2.3. Context abstract contract

This abstract contract provides information about the current execution context, including the sender of the transaction and its data. The source code is stored in the `Context.sol` file, which is identical to OpenZeppelin's implementation at https://github.com/OpenZeppelin/openzeppelin-contracts/blob/c3ae4790c7/contracts/utils/Context.sol.

2.2.4. ERC20Burnable abstract contract

This abstract contract is an extension of ERC-20 that allows the Mercurial Token to be burnable. The source code is stored in the `ERC20Burnable.sol` file, which is identical to OpenZeppelin's implementation at https://github.com/OpenZeppelin/openzeppelin-contracts/blob/c3ae4790c7/contracts/token/ERC20/extensions/ERC20Burnable.sol.

2.2.5. Ownable abstract contract

This abstract contract makes the Mercurial Token Contract ownable. The source code is stored in the `Ownable.sol` file, which is identical to OpenZeppelin's implementation at

[3] https://docs.soliditylang.org/en/latest
[4] https://docs.openzeppelin.com/contracts/4.x/erc20

https://github.com/OpenZeppelin/openzeppelin-contracts/blob/c3ae4790c7/contracts/access/Ownable.sol.

2.2.6. Pausable abstract contract

This abstract contract makes the Mercurial Token Contract pausable. The source code is stored in the *Pausable.sol* file, which is identical to OpenZeppelin's implementation at https://github.com/OpenZeppelin/openzeppelin-contracts/blob/c3ae4790c7/contracts/security/Pausable.sol

2.2.7. ERC20 contract

This contract follows the ERC-20 standard by implementing the *IERC20* and *IERC20Metadata* interfaces. The source code is stored in the *ERC20.sol* file, which is almost identical to OpenZeppelin's implementation at https://github.com/OpenZeppelin/openzeppelin-contracts/blob/c3ae4790c7/contracts/token/ERC20/ERC20.sol, except that the *decimals()* function returns *6* instead of *18*, and some typos in the comments.

2.2.8. Mercurial contract

This is the main contract in the Mercurial Token Contract, which extends the *ERC20*, *ERC20Burnable*, *Pausable* and *Ownable* abstract contracts. This contract consists of five functions:

- *constructor()*: specifies the name, symbol and initial total token supply for the Mercurial Token.
- *pause()*: triggers stop state (all token transfers will be stopped).
- *unpause():* returns to normal state.
- *mint(address to, uint256 amount):* creates *`amount`* tokens and assigns them to *`to`* address.
- *_beforeTokenTransfer(address from, address to, uint256 amount):* is an override function to makes the token transfer pausable.

The source code is stored in the *mercurial.sol* file.

2.3. Discovered issues

During the audit process, the audit team did not discover any security vulnerability issue in the Mercurial Token Contract. Only some minor issues (typos) were found and listed in the following section. These issues do not affect the functional logic of the contract.

2.3.1. Typos [VERY LOW]

There are some typos in the comments in *mercurial.sol* file:

- Line 46: *The defaut value of {decimals} is 18.*
- Line 232: *`to` cannot be the zero address.*

RECOMMENDED FIXES

Update the above comments:

- Line 46: *The default value of {decimals} is 6.*
- Line 232: *`account` cannot be the zero address.*

3. VERSION HISTORY

Version	Date	Status/Changes	Created by
1.0	May 19, 2021	Initial report	Verichains Lab
1.1	May 19, 2021	Public report	Verichains Lab

APPENDIX A: FUNCTION CALL GRAPH

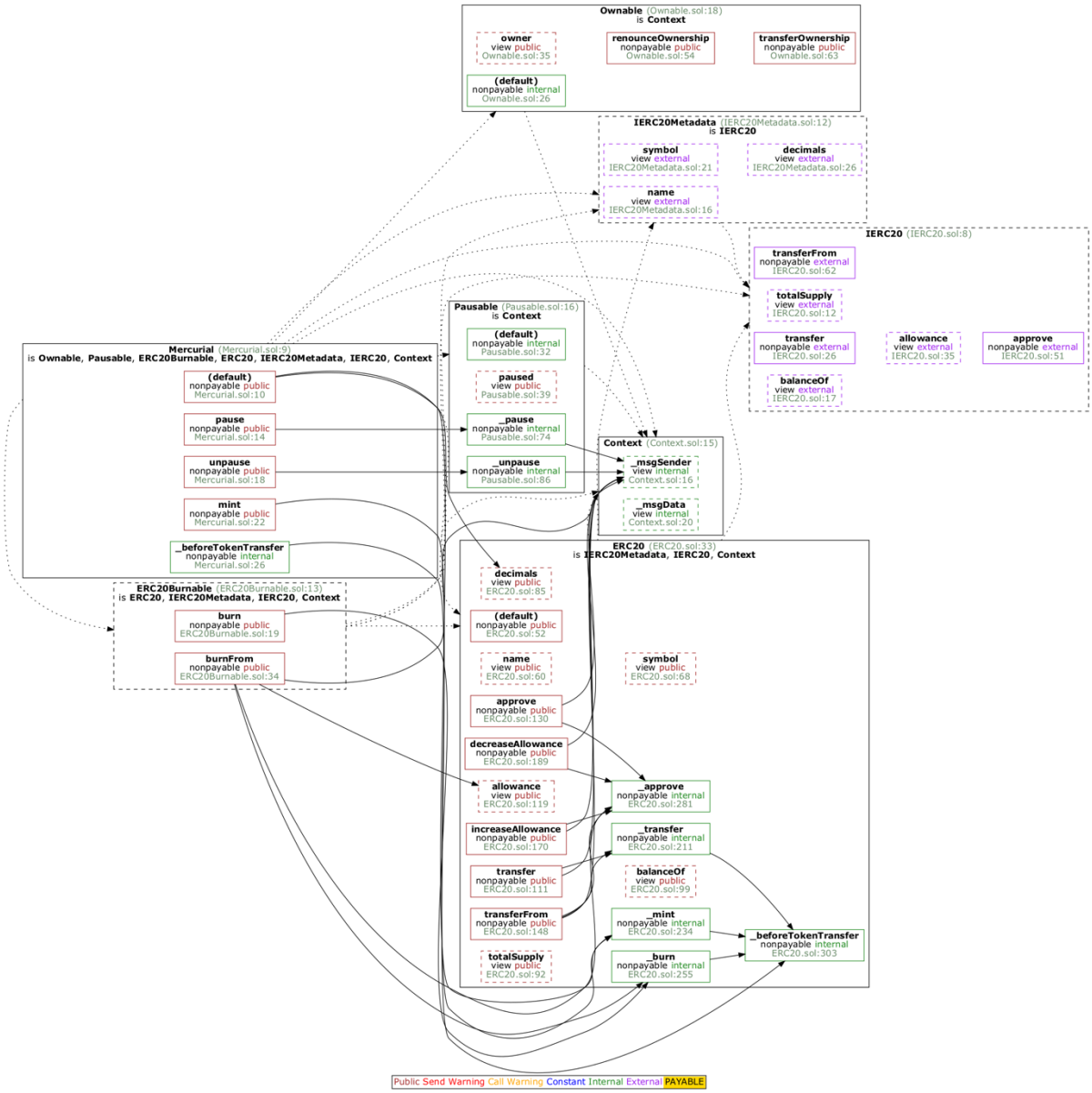

Figure 1: The function call graph of Mercurial Token smart contract