SECURITY AUDIT OF

AVAIL (WORMHOLE) ERC20 TOKEN

Public Report

Nov 19, 2024

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 Nov 19, 2024. We would like to thank the Avail 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 Avail (Wormhole) ERC20 Token. 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 Avail (Wormhole) ERC20 Token

Avail is a dedicated unification layer for Web3, designed to decouple the data availability layer from execution and settlement processes. By doing so, it allows developers to focus on building scalable, efficient, and fast applications. Avail is compatible with any execution environment that supports blockchain scaling, making it a versatile and future-proof solution.

At its core, the Avail token serves as a unifying bridge across blockchain ecosystems. As blockchains expand rapidly, challenges such as interoperability and connectivity have become increasingly evident. Avail addresses these issues by providing a decentralized data layer that enables seamless storage and verification of data across blockchains in a secure and efficient manner.

Avail aims to solve critical problems in the blockchain space, including data fragmentation, scalability limitations, and the lack of interaction between blockchains. By acting as an intermediary, Avail facilitates data sharing and verification without relying on centralized infrastructure, thus enhancing transparency, trust, and decentralization.

What sets Avail apart is its ability to create a scalable and flexible data system. This capability supports a wide range of applications, from DeFi (decentralized finance) to supply chain management solutions. By improving performance and fostering sustainable blockchain growth, Avail contributes to the vision of a unified and interconnected blockchain future.

1.2. Audit Scope

This audit focused on identifying security flaws in code and the design of the Avail (Wormhole) ERC20 Token. It was conducted on commit `81ecf8c8826feeb2cdbe0cee77bee60214aea4a1` on branch `feat/wormhole` from git repository: https://github.com/availproject/contracts/

SHA256 Sum	File
`1a314913e684fd759e91e4234cdf5c6aabdef533a061e9201eadf3e080f5a91b`	`src/AvailWormhole.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 were 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

Avail acknowledges that the security services provided by Verichains, are conducted to the best of their professional abilities but cannot guarantee 100% coverage of all security vulnerabilities. Avail understands and accepts that despite rigorous auditing, certain vulnerabilities may remain undetected. Therefore, Avail agrees that Verichains shall not be held responsible or liable, and shall not be charged for any hacking incidents that occur due to security vulnerabilities not identified during the audit process.

1.5. Acceptance Minute

This final report served by Verichains to the Avail will be considered an Acceptance Minute. Within 7 days, if no any further responses or reports is received from the Avail, the final report will be considered fully accepted by the Avail without the signature.

2. AUDIT RESULT

2.1. Overview

The Avail (Wormhole) ERC20 Token was written in `Solidity` language, with the required version to be `^0.8.25`.

The contract extends `AccessControlDefaultAdminRulesUpgradeable` and `ERC20PermitUpgradeable` from the `OpenZeppelin` library, it also extends `INttToken` from the native-token-transfers interface. The token contract is an ERC20 token with a mint and burn mechanism and an access control mechanism that sets `governance` to be an admin. Only user who has `MINTER_ROLE` can mint and any user can burn their amounts.

Below table describes some properties of the audited Avail (Wormhole) ERC20 Token (as of the report writing time):

PROPERTY	VALUE
Name	Avail (Wormhole)
Symbol	AVAIL.W

Table 2. Avail (Wormhole) Token's properties

Note: The scope of the audit is limited to the source code files provided. All contracts in the scope are **upgradeable** contracts, the contract owner can change the contract logic at any time in the future.

2.2. Findings

During the audit process, the audit team found no vulnerability in the given version of Avail (Wormhole) ERC20 Token.

APPENDIX

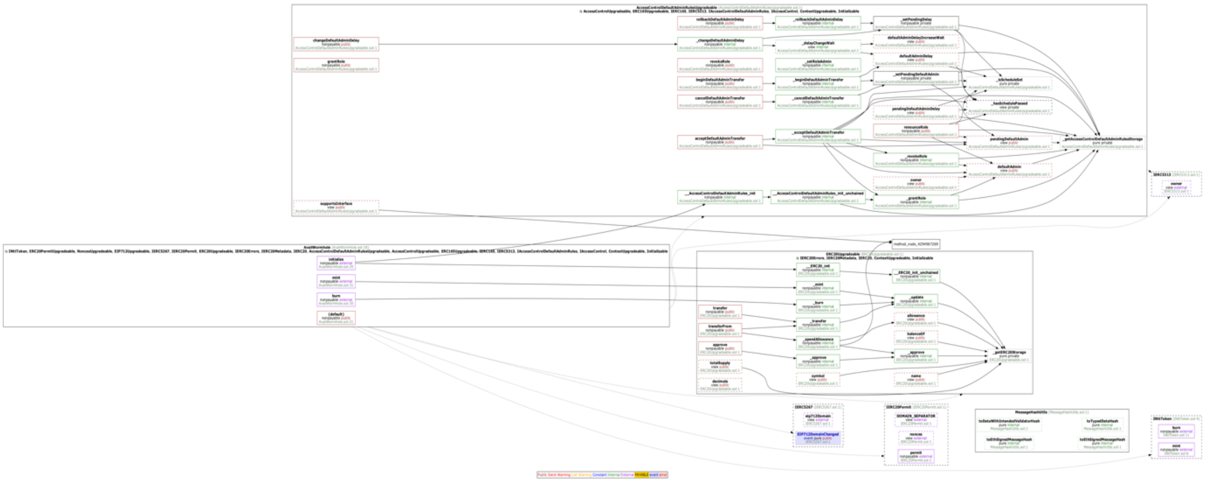
AccessControlDefaultAdminRulesUpgradeable (AccessControlDefaultAdminRulesUpgradeable.sol:1)
is AccessControlUpgradeable, ERC165Upgradeable, IERC165, IERC5313, IAccessControlDefaultAdminRules, IAccessControl, ContextUpgradeable, Initializable
changeDefaultAdminDelay nonpayable public
grantRole nonpayable public
supportsInterface view public
rollbackDefaultAdminDelay nonpayable public
_changeDefaultAdminDelay nonpayable internal
revokeRole nonpayable public
beginDefaultAdminTransfer nonpayable public
cancelDefaultAdminTransfer nonpayable public
acceptDefaultAdminTransfer nonpayable public
__AccessControlDefaultAdminRules_init nonpayable internal
_rollbackDefaultAdminDelay nonpayable internal
_delayChangeWait view internal
_setRoleAdmin nonpayable internal
_beginDefaultAdminTransfer nonpayable internal
_cancelDefaultAdminTransfer nonpayable internal
_acceptDefaultAdminTransfer nonpayable internal
__AccessControlDefaultAdminRules_init_unchained nonpayable internal
_setPendingDelay nonpayable private
defaultAdminDelayIncreaseWait view public
defaultAdminDelay view public
_setPendingDefaultAdmin nonpayable private
pendingDefaultAdminDelay view public
renounceRole nonpayable public
_revokeRole nonpayable internal
owner view public
_grantRole nonpayable internal
_isScheduleSet pure private
_hasSchedulePassed view private
pendingDefaultAdmin view public
defaultAdmin view public
_getAccessControlDefaultAdminRulesStorage pure private
IERC5313 (IERC5313.sol:1)
owner view external
method_node_4294967269
AvailWormhole (AvailWormhole.sol:18)
is INttToken, ERC20PermitUpgradeable, NoncesUpgradeable, EIP712Upgradeable, IERC5267, IERC20Permit, ERC20Upgradeable, IERC20Errors, IERC20Metadata, IERC20, AccessControlDefaultAdminRulesUpgradeable, AccessControlUpgradeable, ERC165Upgradeable, IERC165, IERC5313, IAccessControlDefaultAdminRules, IAccessControl, ContextUpgradeable, Initializable
initialize nonpayable external
mint nonpayable external
burn nonpayable external
(default) nonpayable public
ERC20Upgradeable (ERC20Upgradeable.sol:1)
is IERC20Errors, IERC20Metadata, IERC20, ContextUpgradeable, Initializable
transfer nonpayable public
transferFrom nonpayable public
approve nonpayable public
totalSupply view public
decimals view public
__ERC20_init nonpayable internal
_mint nonpayable internal
_burn nonpayable internal
_transfer nonpayable internal
_spendAllowance nonpayable internal
_approve nonpayable internal
symbol view public
__ERC20_init_unchained nonpayable internal
_update nonpayable internal
allowance view public
balanceOf view public
_approve nonpayable internal
name view public
_getERC20Storage pure private
IERC5267 (IERC5267.sol:1)
eip712Domain view external
EIP712DomainChanged event pure public
IERC20Permit (IERC20Permit.sol:1)
DOMAIN_SEPARATOR view external
nonces view external
permit nonpayable external
MessageHashUtils (MessageHashUtils.sol:1)
toDataWithIntendedValidatorHash pure internal
toTypedDataHash pure internal
toEthSignedMessageHash pure internal
toEthSignedMessageHash pure internal
INttToken (INttToken.sol:4)
burn nonpayable external
mint nonpayable external
Public Send Warning Call Warning Constant Internal External PAYABLE event error

Image 1. Avail (Wormhole) ERC20 Token call graph

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Nov 19, 2024*	Public Report	Verichains Lab

Table 3. Report versions history