SECURITY AUDIT OF

ASHWARD TOKEN AND IDO SMART CONTRACTS

Public Report

Mar 18, 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 Mar 18, 2022. We would like to thank the ASHWARD 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 ASHWARD Token and IDO 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 ASHWARD Token and IDO Smart Contracts

Ashward is a virtual world where players can own, and monetize their gaming experiences in the Binance Smart Chain (BSC) network using ASC, the platform's utility token. Players will be able to explore the fantasy medieval land, gather the bravest warriors to defeat aggressive bosses or engage in fierce battles against other players.

1.2. Audit scope

This audit focused on identifying security flaws in code and the design of the smart contracts of ASHWARD Token and IDO Smart Contracts. It was conducted on commit a589c98bb89b7d939f8c2c8e5c72a90dbcc59327 from git repository *https://github.com/ashward-game/contract-ashward*.

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

SHA256 SUM	FILE
3ce476b8d97dc91d4f8af145a3c1f6c3eb2eb403a1778125209cc28c4a0917ad	IDO.sol
7e452b9742e40b8ad36bbe7614cfa403db4b00f229c388c9f97aded4600d3095	Token.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 RESULTs

2.1. Overview

The ASHWARD Token and IDO Smart Contracts was written in Solidity language, with the required version to be ^0.8.9.

2.1.1. Token contract

The token extends ERC1363, Pausable, AccessControl contracts. The ERC1363 is an implementation of the ERC-1363 Payable Token that defines a token interface for ERC-20 tokens that supports executing recipient code after transfer or transferFrom, or spender code after approve. AccessControl allows the contract to implement role-based access control mechanisms. There are 3 roles: DEFAULT_ADMIN_ROLE, SELLING_ROLE, FREETAX_ROLE. The user has DEFAULT_ADMIN_ROLE may use the pause/unpause function from Pausable to control the activities of the contract. Users can only transfer tokens when the contract is not paused.

When normal users transfer the tokens to SELLING_ROLE addresses, they must pay an amount of tax. Only FREETAX_ROLE users were set by DEFAULT_ADMIN_ROLE user may skip this action.

2.1.2. IDO contract

The IDO extends AccessControl, Pausable, contracts. AccessControl allows the contract to implement role-based access control mechanisms. There are 3 roles: DEFAULT_ADMIN_ROLE, SUBSCRIBER_ROLE. The user has DEFAULT_ADMIN_ROLE may use the pause/unpause function from Pausable to control the activities of the contract. The DEFAULT_ADMIN_ROLE may also grant SUBSCRIBER_ROLE for other users.

The ASHWARD team uses this contract for selling the tokens. The tokens are sold in packages following 2 phases. In the first phase, only users who were granted the SUBSCRIBER_ROLE may buy packages with a limited number. In the second phase, anyone may purchase packages which are available in the pool.

When users purchase a package, the user won't receive the tokens immediately. The contract will create an emit event to log the number of the tokens that users purchased.

2.2. Findings

During the audit process, the audit team found no vulnerability issue in the given version of ASHWARD Token and IDO Smart Contracts.

APPENDIX

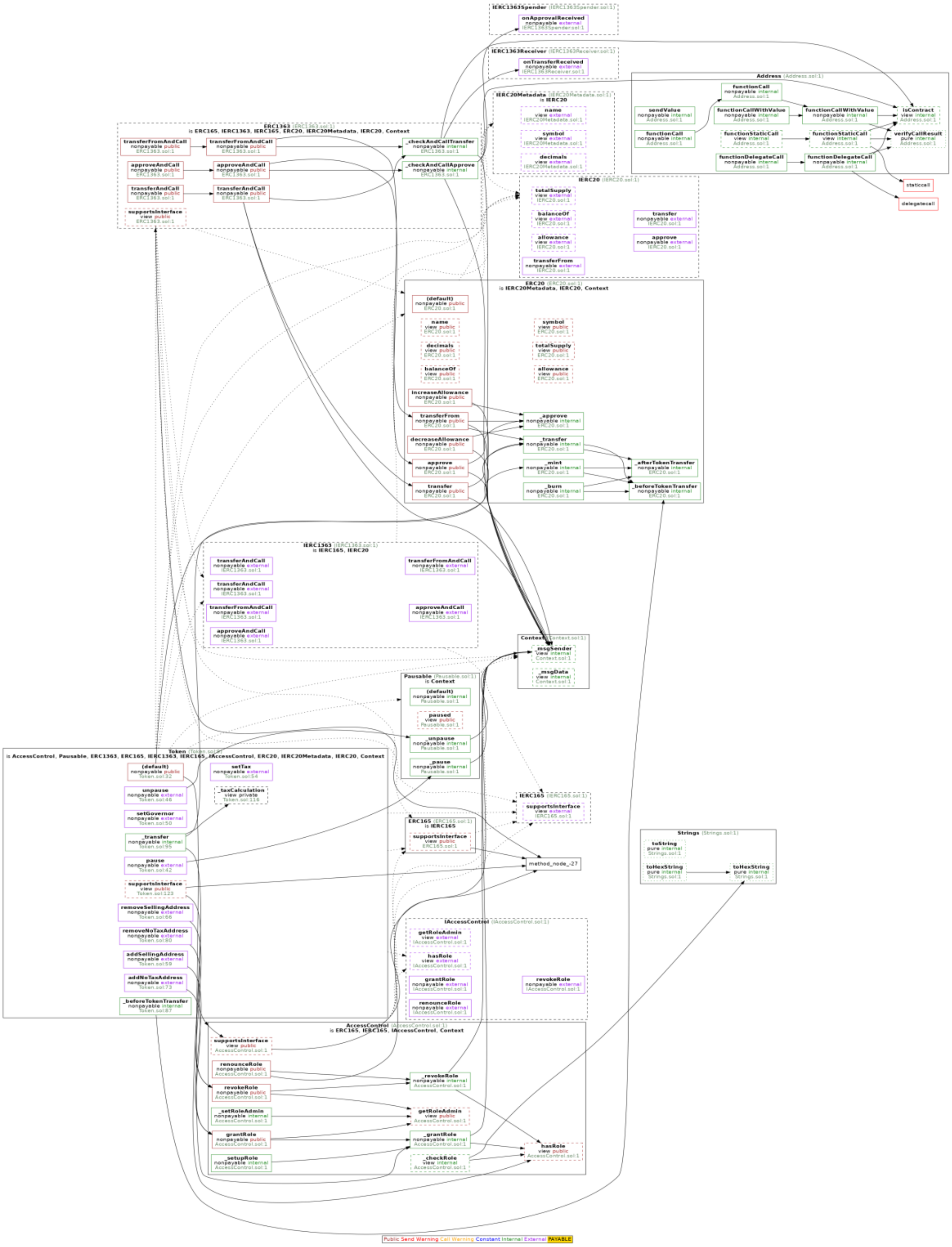
IERC1363Spender (IERC1363Spender.sol:1)
onApprovalReceived nonpayable external IERC1363Spender.sol:1
IERC1363Receiver (IERC1363Receiver.sol:1)
onTransferReceived nonpayable external IERC1363Receiver.sol:1
Address (Address.sol:1)
functionCall nonpayable internal Address.sol:1
sendValue nonpayable internal Address.sol:1
functionCallWithValue nonpayable internal Address.sol:1
functionCallWithValue nonpayable internal Address.sol:1
isContract view internal Address.sol:1
functionCall nonpayable internal Address.sol:1
functionStaticCall view internal Address.sol:1
functionStaticCall view internal Address.sol:1
verifyCallResult pure internal Address.sol:1
functionDelegateCall nonpayable internal Address.sol:1
functionDelegateCall nonpayable internal Address.sol:1
staticcall
delegatecall
IERC20Metadata (IERC20Metadata.sol:1) is IERC20
name view external IERC20Metadata.sol:1
symbol view external IERC20Metadata.sol:1
decimals view external IERC20Metadata.sol:1
ERC1363 (ERC1363.sol:1) is ERC165, IERC1363, IERC165, ERC20, IERC20Metadata, IERC20, Context
transferFromAndCall nonpayable public ERC1363.sol:1
transferFromAndCall nonpayable public ERC1363.sol:1
_checkAndCallTransfer nonpayable internal ERC1363.sol:1
approveAndCall nonpayable public ERC1363.sol:1
approveAndCall nonpayable public ERC1363.sol:1
_checkAndCallApprove nonpayable internal ERC1363.sol:1
transferAndCall nonpayable public ERC1363.sol:1
transferAndCall nonpayable public ERC1363.sol:1
supportsInterface view public ERC1363.sol:1
IERC20 (IERC20.sol:1)
totalSupply view external IERC20.sol:1
transfer nonpayable external IERC20.sol:1
balanceOf view external IERC20.sol:1
allowance view external IERC20.sol:1
approve nonpayable external IERC20.sol:1
transferFrom nonpayable external IERC20.sol:1
ERC20 (ERC20.sol:1) is IERC20Metadata, IERC20, Context
(default) nonpayable public ERC20.sol:1
name view public ERC20.sol:1
symbol view public ERC20.sol:1
decimals view public ERC20.sol:1
totalSupply view public ERC20.sol:1
balanceOf view public ERC20.sol:1
allowance view public ERC20.sol:1
increaseAllowance nonpayable public ERC20.sol:1
transferFrom nonpayable public ERC20.sol:1
_approve nonpayable internal ERC20.sol:1
decreaseAllowance nonpayable public ERC20.sol:1
_transfer nonpayable internal ERC20.sol:1
approve nonpayable public ERC20.sol:1
_mint nonpayable internal ERC20.sol:1
_afterTokenTransfer nonpayable internal ERC20.sol:1
transfer nonpayable public ERC20.sol:1
_burn nonpayable internal ERC20.sol:1
_beforeTokenTransfer nonpayable internal ERC20.sol:1
IERC1363 (IERC1363.sol:1) is IERC165, IERC20
transferAndCall nonpayable external IERC1363.sol:1
transferFromAndCall nonpayable external IERC1363.sol:1
transferAndCall nonpayable external IERC1363.sol:1
transferFromAndCall nonpayable external IERC1363.sol:1
approveAndCall nonpayable external IERC1363.sol:1
approveAndCall nonpayable external IERC1363.sol:1
Context (Context.sol:1)
_msgSender view internal Context.sol:1
_msgData view internal Context.sol:1
Pausable (Pausable.sol:1) is Context
(default) nonpayable internal Pausable.sol:1
paused view public Pausable.sol:1
_unpause nonpayable internal Pausable.sol:1
_pause nonpayable internal Pausable.sol:1
Token (Token.sol:9) is AccessControl, Pausable, ERC1363, ERC165, IERC1363, IERC165, IAccessControl, ERC20, IERC20Metadata, IERC20, Context
(default) nonpayable public Token.sol:32
setTax nonpayable external Token.sol:54
unpause nonpayable external Token.sol:46
_taxCalculation view private Token.sol:116
setGovernor nonpayable external Token.sol:50
_transfer nonpayable internal Token.sol:95
pause nonpayable external Token.sol:42
supportsInterface view public Token.sol:123
removeSellingAddress nonpayable external Token.sol:66
removeNoTaxAddress nonpayable external Token.sol:80
addSellingAddress nonpayable external Token.sol:59
addNoTaxAddress nonpayable external Token.sol:73
_beforeTokenTransfer nonpayable internal Token.sol:87
IERC165 (IERC165.sol:1)
supportsInterface view external IERC165.sol:1
ERC165 (ERC165.sol:1) is IERC165
supportsInterface view public ERC165.sol:1
method_node_-27
Strings (Strings.sol:1)
toString pure internal Strings.sol:1
toHexString pure internal Strings.sol:1
toHexString pure internal Strings.sol:1
IAccessControl (IAccessControl.sol:1)
getRoleAdmin view external IAccessControl.sol:1
hasRole view external IAccessControl.sol:1
grantRole nonpayable external IAccessControl.sol:1
revokeRole nonpayable external IAccessControl.sol:1
renounceRole nonpayable external IAccessControl.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 internal AccessControl.sol:1
revokeRole nonpayable public 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
Public Send Warning Call Warning Constant Internal External PAYABLE

Image 1. Token call graph

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Mar 18, 2022*	Public Report	Verichains Lab

Table 2. Report versions history