SECURITY AUDIT OF

POCO TOKEN SMART CONTRACT

PUBLIC REPORT

AUG 20, 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
Binance Chain	Binance Chain is a blockchain software system developed by Binance and its community.
Binance Smart Chain (BSC)	Binance Smart Chain (BSC) is a blockchain network built for running smart contract-based applications. BSC runs in parallel with Binance's native Binance Chain (BC), which allows users to get the best of both worlds: the high transaction capacity of BC and the smart contract functionality of BSC.
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 Aug 20, 2021. We would like to thank the Poco team 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 Poco 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 Poco 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 Poco and Poco Token

Poco brings you into the new gaming world. Let's immerse yourself in Pocoland when leading the powerful team with 5 Poco warrants owning different elements, defeat your enemies then collect the huge reward by POCO token.

Poco Token (POCO) is the BEP-20 token of Poco, with an initial total supply of 180 million.

1.2. Audit scope

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

The audited contract is the Poco Token Contract that was deployed on Binance Smart Chain Mainnet at address *0xfec70a44c683cc86397abed3700e115f0e956505*. The details of the deployed smart contract are listed in Table 1.

FIELD	VALUE
Contract Name	Poco Token
Contract Address	0xfec70a44c683cc86397abed3700e115f0e956505
Compiler Version	v0.5.16+commit.9c3226ce
Optimization Enabled	Yes with 200 runs
Explorer	https://bscscan.com/address/0xfec70a44c683cc86397abed3700e115f0e956505

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 Poco Token Contract is an BEP-20 Token Contract[1], which implements a standard interface for token as defined in IBEP-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 Poco Token Contract (as of the report writing time).

PROPERTY	VALUE
Name	Poco Token
Symbol	POCO
Decimals	18
Owner	0x8165af169efda9be54418f7c35a031137e6f0d8e
Total Supply	180,000,000 ($\times 10^{18}$) Note: the number of decimals is 18, so the total representation tokens will be 180,000,000, or 180 million.

Table 3: The Poco Token properties

Besides the standard interface of an BEP-20 token, the Poco Token Contract also implements some additional functional logics by extending the following contracts:

- Context: provides information about the current execution context, including the sender of the transaction and its data. While these are generally available via msg.sender and msg.data, they should not be accessed in such a direct manner, since when dealing with GSN meta-transactions the account sending and paying for execution may not be the actual sender (as far as an application is concerned).
- Ownable: this 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. The current owner can renounce or transfer ownership to a new owner.

[1] https://docs.binance.org/smart-chain/developer/BEP20.html
[2] https://docs.binance.org/smart-chain/developer/IBEP20.sol

2.2. Contract codes

The Poco Token Contract was written in Solidity language[3], with the required version to be *0.5.16*.

The source codes consist of three contracts, one interface and one library. Almost all source codes in the Poco Token Contract are imported from Binance's implementation template of BEP20-related contracts[4] and OpenZeppelin Contracts[5].

2.2.1. IBEP20 interface

This is the interface of the BEP-20 token standard. The source code is referenced from the official Binance's documentation.

2.2.2. Context contract

This contract provides information about the current execution context, including the sender of the transaction and its data. The source code is referenced from OpenZeppelin's implementation.

2.2.3. SafeMath library

This library provides wrappers over Solidity's arithmetic operations with added overflow checks. The source code is referenced from OpenZeppelin's implementation.

2.2.4. Ownable contract

This contract makes the Poco Token Contract ownable. The source code is referenced from OpenZeppelin's implementation.

2.2.5. Poco Token contract

This is the main contract in the Poco Token Contract, which extends the *IBEP20*, *Context* and *Ownable* contracts. Below is a summary of some important functions in this contract:

- *constructor()*: specifies the name, symbol and initial total token supply for the Poco Token.
- *getOwner()*: returns the bep token owner.
- *symbol()*: returns the token symbol.
- *name()*: returns the token name.
- *totalSupply():* returns the total supply.
- *balanceOf(address account):* returns the balance of the input account.

[3] https://docs.soliditylang.org/
[4] https://docs.binance.org/smart-chain/developer/BEP20Token.template
[5] https://openzeppelin.com/contracts/

- *`transfer(address recipient, uint256 amount):`* transfers `amount` tokens from the sender to `recipient`.
- *`allowance(address owner, address spender):`* returns the `amount` which `spender` is still allowed to withdraw from `owner`.
- *`approve(address spender, uint256 amount):`* allows `spender` to withdraw from the caller's account multiple times, up to the `amount`. If this function is called again it overwrites the current allowance with `amount`.
- *`transferFrom(address sender, address recipient, uint256 amount):`* transfers `amount` of tokens from address `sender` to address `recipient`.
- *`increaseAllowance(address spender, uint256 addedValue):`* increases the allowance amount for `spender` by `addedValue`.
- *`decreaseAllowance(address spender, uint256 subtractedValue):`* decreases the allowance amount for `spender` by `subtractedValue`.
- *`burnFrom(uint256 amount):`* destroys `amount` tokens from `account`. `amount` is then deducted from the caller's allowance.

2.3. Findings

During the audit process, the audit team did not discover any security vulnerability issue in the Poco Token Contract.

2.4. Additional notes and recommendations

2.4.1. Outdated version of Solidity

The required version of Solidity in current Poco Token contract source code is `0.5.16`, which is quite outdated. Updating the Solidity version from `0.5.16` to `0.8.0` will introduce many features like:

- Automatic integer overflow/underflow checking for arithmetic operations.
- Abstract contract support.

RECOMMENDATION

Update the Solidity version to `0.8.0` using *`pragma solidity ^0.8.0`*.

2.4.2. Unused _mint function

The *`_mint`* internal function is unused in this contract.

RECOMMENDATION

Remove the *`_mint`* function.

3. VERSION HISTORY

Version	Date	Status/Changes	Created by
1.0	Aug 19, 2021	Initial private report	Verichains Lab
1.1	Aug 20, 2021	Public report	Verichains Lab

APPENDIX A: FUNCTION CALL GRAPH

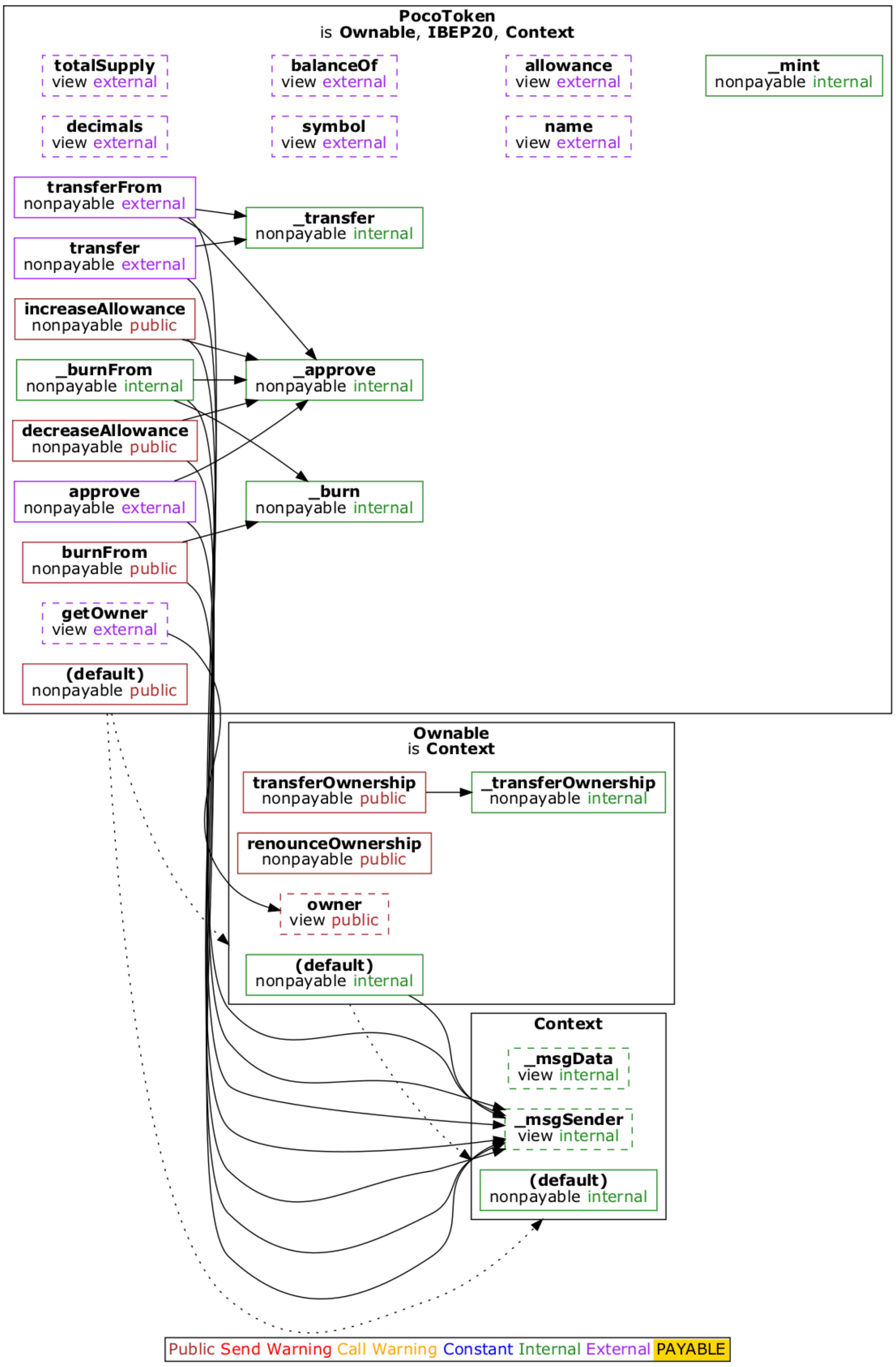

Figure 1: The function call graph of Poco Token smart contract