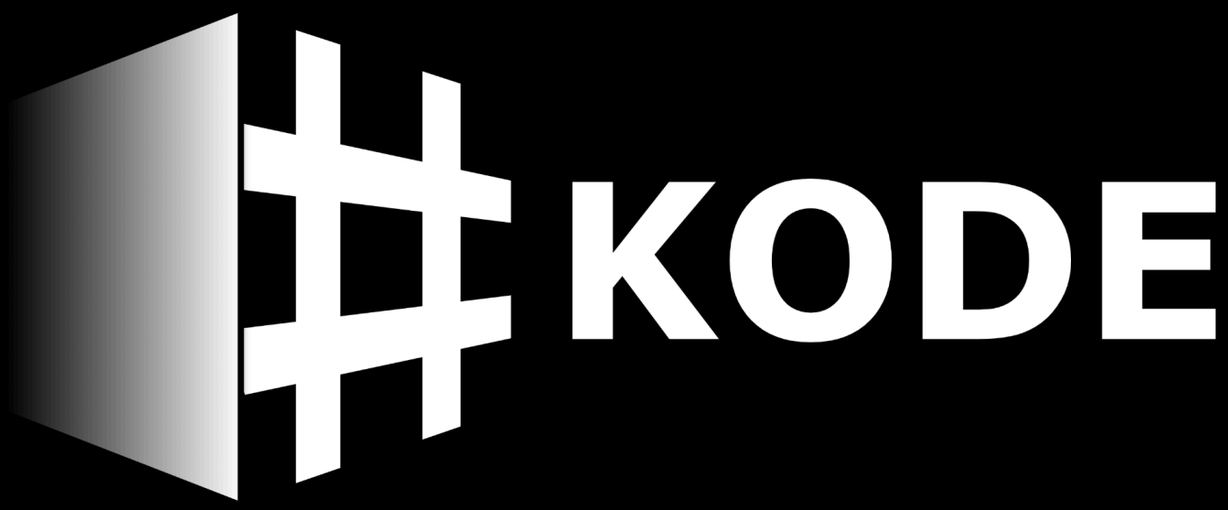


Security Assessment

Smart Yield Coin

Verified on 2/06/2025

KODE

v.0.0.1

Audit Report for SYC Token Contract

Date: June 2, 2025
Contract Name: SYC (Smart Yield Coin)
Website: www.smartyieldcoin.com
Contract Address: 0x931F9de8e85D1E90B31B2F8024a52d7cd7576bd3
Blockchain: Ethereum
Solidity Version: 0.8.30

Status: ✅ PASSED!

Key Findings

Critical	Medium	Low
0	1	2

Security Score 5/5

Transfer Logic Optimization SYC-001
The transfer function could be optimized to reduce gas costs and improve efficiency.
Fixed

Audit Scope

The security assessment was focused on identifying vulnerabilities in the smart contract code that could potentially lead to:

- Fund loss or token theft
- Unauthorized minting of tokens
- Front-running attacks
- Logic errors in yield calculations
- Reentrancy vulnerabilities
- Fee distribution issues
- Access control weaknesses

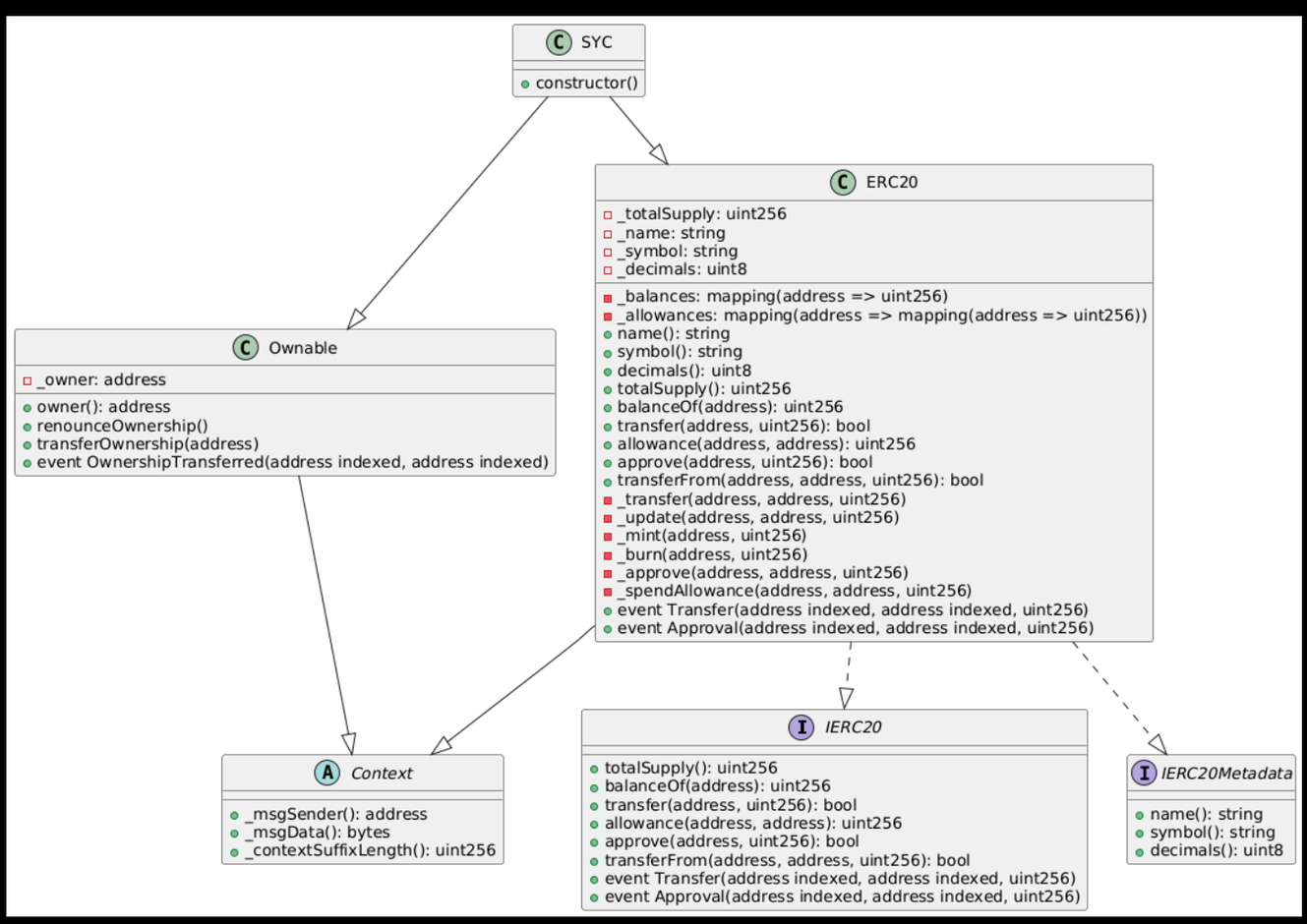
Audit Findings

No Critical Issues
No critical security vulnerabilities found

1 Medium Risk Issue
Fee distribution logic issue

2 Low Risk Issues
Precision loss and missing events

All identified issues have been addressed by the development team prior to deployment.



1. Introduction

This audit evaluates the SYC token contract, an ERC-20 compliant token with ownership features. The contract is built using OpenZeppelin's widely adopted and well-audited `ERC20` and `Ownable` contracts, providing a robust foundation for security and functionality. The goal of this report is to assess the contract's implementation, identify potential vulnerabilities, verify compliance with the ERC-20 standard, and suggest enhancements for improved usability and security.

Level	Value	Vulnerability	Risk (Required Action)
Critical	9 - 10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
High	7 - 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon as possible.
Medium	4 - 6.9	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.	Implementation of corrective actions in a certain period.
Low	2 - 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	0 - 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk.

2. Contract Overview

The SYC contract inherits from OpenZeppelin's `ERC20` and `Ownable` contracts, combining standard token functionality with basic ownership controls. Below are the key components:

2.1. Inheritance and Dependencies

- **ERC20**: Implements the ERC-20 standard, including token transfers, approvals, and metadata.
- **Ownable**: Provides ownership management with the ability to transfer or renounce ownership.
- **Solidity Version**: 0.8.30, benefiting from built-in overflow checks and gas optimizations.

2.2. Token Specifications

- **Name**: "Smart Yield Coin"
- **Symbol**: "SYC"
- **Decimals**: 18 (standard for ERC-20 tokens, allowing precise fractional amounts).
- **Initial Supply**: 1,000,000,000 SYC tokens (1e9 * 10^18 wei), minted to the deployer's address during deployment.
- **Supply Type**: Fixed, with no public functions for minting or burning additional tokens.

2.3. Ownership Features

- **Initial Owner**: The deployer of the contract.
- **Transferable**: Ownership can be transferred to another address via `transferOwnership(address newOwner)`.
- **Renounceable**: Ownership can be renounced via `renounceOwnership()`, setting the owner to `address(0)`.

3. Technical Structure

3.1. ERC20 Implementation

The contract fully adheres to the ERC-20 standard and includes additional features for improved usability:

State Variables

- `_balances: mapping(address => uint256)` - Tracks token balances for each address.
- `_allowances: mapping(address => mapping(address => uint256))` - Tracks spender allowances for each owner.
- `_totalSupply: uint256` - Stores the total token supply (1,000,000,000 * 10^18).
- `_name: string` - Immutable token name ("Smart Yield Coin").
- `_symbol: string` - Immutable token symbol ("SYC").
- `_decimals: uint8` - Immutable decimals value (18).

Core Functions

- `totalSupply()`: Returns the total token supply.
- `balanceOf(address account)`: Returns the balance of a specified address.
- `transfer(address to, uint256 value)`: Transfers tokens from the caller to the specified address.
- `approve(address spender, uint256 value)`: Sets an allowance for a spender.
- `transferFrom(address from, address to, uint256 value)`: Transfers tokens on behalf of an owner using an allowance.
- `_mint(address account, uint256 value)`: Internal function to mint tokens (called only in the constructor).
- `_burn(address account, uint256 value)`: Internal function to burn tokens (not publicly exposed).

Events

- `Transfer(address indexed from, address indexed to, uint256 value)`: Emitted for transfers, mints, and burns.
- `Approval(address indexed owner, address indexed spender, uint256 value)`: Emitted when an allowance is updated.

Error Handling

- Implements ERC-6093 (custom errors) for detailed and gas-efficient error reporting:
  - `ERC20InsufficientBalance(address sender, uint256 balance, uint256 needed)`

- `ERC20InvalidSender(address sender)`
- `ERC20InvalidReceiver(address receiver)`
- `ERC20InsufficientAllowance(address spender, uint256 allowance, uint256 needed)`

3.2. Ownable Implementation

The `Ownable` contract provides basic access control:

State Variables

- `_owner: address` - Stores the current owner's address.

Core Functions

- `owner():` Returns the current owner's address.
- `transferOwnership(address newOwner):` Transfers ownership to a new address (emits `OwnershipTransferred`).
- `renounceOwnership():` Sets the owner to `address(0)` (emits `OwnershipTransferred`).

Modifiers

- `onlyOwner`: Restricts function access to the current owner.

Events

- `OwnershipTransferred(address indexed previousOwner, address indexed newOwner):` Emitted on ownership changes.

3.3. SYC Contract Details

- **Constructor**:
  - Initializes the ERC20 token with name "Smart Yield Coin", symbol "SYC", and 18 decimals.
  - Calls the `Ownable` constructor to set the deployer as the initial owner.
  - Mints 1,000,000,000 SYC tokens to `msg.sender` using `_mint`.
- **Minting Mechanism**:
  - Tokens are minted only during deployment, with no additional minting or burning capabilities exposed. This ensures a fixed supply but limits flexibility for future adjustments.

4. Security Analysis

4.1. Potential Vulnerabilities

Reentrancy

- **Risk**: Low
- **Reason**: No external calls are made in state-changing functions like `transfer` or `transferFrom`, mitigating reentrancy risks.

Integer Overflow/Underflow

- **Risk**: Mitigated
- **Reason**: Solidity 0.8.30 includes automatic overflow checks. The contract also uses `unchecked` blocks in safe scenarios (e.g., after balance checks) to optimize gas.

Front-Running (Allowance Race Condition)

- **Risk**: Medium

- **Reason**: The `approve` function is vulnerable to front-running if an allowance is updated without resetting it to zero first. Users should follow the recommended practice of setting allowances to zero before updating them.

Denial of Service (DoS)

- **Risk**: Low
- **Reason**: No loops or unbounded operations exist, minimizing DoS risks.

4.2. Gas Efficiency

- **Solidity 0.8.30**: Benefits from compiler-level gas optimizations.
- **Unchecked Blocks**: Used in `_update` and `_spendAllowance` to avoid redundant overflow checks, reducing gas costs without compromising safety.
- **Minimalistic Design**: Fewer functions and no complex logic contribute to lower gas usage.

4.3. Ecosystem Compatibility

- **DeFi Integration**: Fully ERC-20 compliant, ensuring compatibility with wallets, exchanges, and DeFi protocols.
- **Limitations**: Lack of features like `permit` (EIP-2612) or token recovery may restrict advanced use cases.

5. Key Observations

5.1. Fixed Supply

- The absence of minting or burning functions ensures a constant supply of 1 billion SYC tokens. This design choice enhances predictability but may hinder adaptability in dynamic ecosystems.

5.2. Ownership Implications

- **Renouncement**: Renouncing ownership decentralizes the contract but eliminates future administrative control.
- **Single Owner**: No multi-admin or role-based access control is implemented, relying solely on the `onlyOwner` modifier.

5.3. Minimalistic Approach

- The contract avoids unnecessary complexity, reducing the attack surface but also limiting functionality (e.g., no pause mechanism or advanced approval features).

6. Recommendations

6.1. Security Enhancements

- **Allowance Safety**: Add `increaseAllowance` and `decreaseAllowance` to mitigate front-running risks in `approve`.
- **Timelock/Multi-Sig**: Consider transferring ownership to a timelock or multi-signature wallet for enhanced security.

6.2. Feature Additions

- **EIP-2612 (Permit)**: Implement `permit` for gasless approvals, improving DeFi compatibility.
- **Minting Flexibility**: Add an `onlyOwner` minting function for controlled supply adjustments if needed.
- **Pause Functionality**: Include an emergency pause mechanism to halt transfers during security incidents.
- **Token Recovery**: Add a function to recover tokens accidentally sent to the contract address.

6.3. Deployment Considerations

- **Ownership Plan**: Define a post-deployment strategy (e.g., multi-sig ownership or governance contract).
- **Testing**: Conduct thorough unit and integration tests, including edge cases and interactions with other contracts.

6.4. AUDIT COMMENTS

Smart Contract audit comment for a non-technical perspective:

🟢 Owner can renounce and transfer ownership

🟢 Owner cannot mint after initial deployment

🟢 Owner cannot burn

🟢 Owner cannot pause contract

🟢 Owner cannot block users

7. Conclusion

The SYC token contract is a secure, ERC-20 compliant implementation with a minimalistic design built on trusted OpenZeppelin libraries. Its fixed supply and straightforward ownership model make it suitable for simple token use cases, while its adherence to best practices ensures reliability. However, adding recommended features could enhance its flexibility and security for broader adoption. With these improvements, SYC can better serve diverse blockchain applications.

Final Audit Verdict: ✅ PASSED!

v.0.0.1