Cyberscope

Audit Report

Year Staking Token

November 2023

Network	ETH
Address	0x88021a02c0fe99dff7ec8fbc61f939243355a5fa
Audited by	© cyberscope

Analysis

Critical Medium Minor / Informative Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Passed
●	OTUT	Transfers User's Tokens	Passed
●	ELFM	Exceeds Fees Limit	Passed
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Passed

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
	ZDT	Zero Decimals Token	Unresolved
	L09	Dead Code Elimination	Unresolved
	L18	Multiple Pragma Directives	Unresolved
	L19	Stable Compiler Version	Unresolved

Table of Contents

Review

Contract Name	YearStakingToken
Compiler Version	v0.8.18+commit.87f61d96
Optimization	200 runs
Explorer	https://etherscan.io/address/0x88021a02c0fe99dff7ec8fbc61f939243355a5fa
Address	0x88021a02c0fe99dff7ec8fbc61f939243355a5fa
Network	ETH
Symbol	YST

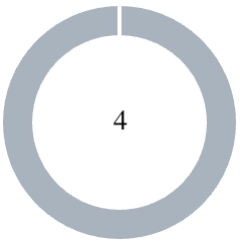
Audit Updates

Initial Audit	03 Nov 2023

Source Files

Filename	SHA256
YearStakingToken.sol	3939356a1936ec1efb5316de45ca44a34a236dad625f5aaf624fa884c911d0ec

Findings Breakdown



Critical	0
Medium	0
Minor / Informative	4

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	0	0	0
Medium	0	0	0	0
Minor / Informative	4	0	0	0

ZDT - Zero Decimals Token

Criticality	Minor / Informative
Location	YearStakingToken.sol#L231
Status	Unresolved

Description

The token contians 0 decimal precision, which might lead to potential problems for users and applications interacting with the token. While this is not necessarily a critical issue, it does introduce certain complexities and limitations that should be considered when designing and using the token.

The number of decimal places in an ERC20 token is a crucial parameter, as it determines the token's divisibility and the level of precision in its values. A standard ERC20 token typically has 18 decimal places, allowing for highly granular fractional ownership and precise transfers. However, when a token is configured with 0 decimal places, each unit of the token becomes indivisible, effectively acting as a whole unit, similar to an integer.

Potential Problems:

1. Limited divisibility: A token with 0 decimals cannot be divided into fractional units. This could create difficulties when dealing with smaller amounts of the token, especially in situations where precise calculations are required.
2. User experience: End-users may find it challenging to work with a token that lacks decimal places. Transactions may result in unexpected rounding errors or require manual adjustments to match desired values.
3. Compatibility with DApps and Exchanges: Many decentralized applications (DApps) and exchanges are optimized for tokens with the standard 18 decimal places. Using a token with 0 decimals in such environments may lead to issues or unexpected behavior.
4. Interoperability: Integration with other tokens or systems may be complicated, as most tokens follow the 18-decimal-place convention. This could hinder the token's overall usability.

L09 - Dead Code Elimination

Criticality	Minor / Informative
Location	YearStakingToken.sol#L417
Status	Unresolved

Description

In Solidity, dead code is code that is written in the contract, but is never executed or reached during normal contract execution. Dead code can occur for a variety of reasons, such as:

- Conditional statements that are always false.
- Functions that are never called.
- Unreachable code (e.g., code that follows a return statement).

Dead code can make a contract more difficult to understand and maintain, and can also increase the size of the contract and the cost of deploying and interacting with it.

```
function _burn(address account, uint256 amount) internal virtual {
        require(account != address(0), "ERC20: burn from the zero
address");

        _beforeTokenTransfer(account, address(0), amount);

        uint256 accountBalance = _balances[account];
...
            _totalSupply -= amount;
        }

        emit Transfer(account, address(0), amount);

        _afterTokenTransfer(account, address(0), amount);
    }
```

Recommendation

To avoid creating dead code, it's important to carefully consider the logic and flow of the contract and to remove any code that is not needed or that is never executed. This can help improve the clarity and efficiency of the contract.

L18 - Multiple Pragma Directives

Criticality	Minor / Informative
Location	YearStakingToken.sol#L7,34,115,145,510
Status	Unresolved

Description

If the contract includes multiple conflicting pragma directives, it may produce unexpected errors. To avoid this, it's important to include the correct pragma directive at the top of the contract and to ensure that it is the only pragma directive included in the contract.

```
pragma solidity ^0.8.0;
pragma solidity ^0.8.9;
```

Recommendation

It is important to include only one pragma directive at the top of the contract and to ensure that it accurately reflects the version of Solidity that the contract is written in.

By including all required compiler options and flags in a single pragma directive, the potential conflicts could be avoided and ensure that the contract can be compiled correctly.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	YearStakingToken.sol#L7,34,115,145,510
Status	Unresolved

Description

The `^` symbol indicates that any version of Solidity that is compatible with the specified version (i.e., any version that is a higher minor or patch version) can be used to compile the contract. The version lock is a mechanism that allows the author to specify a minimum version of the Solidity compiler that must be used to compile the contract code. This is useful because it ensures that the contract will be compiled using a version of the compiler that is known to be compatible with the code.

```
pragma solidity ^0.8.0;
pragma solidity ^0.8.9;
```

Recommendation

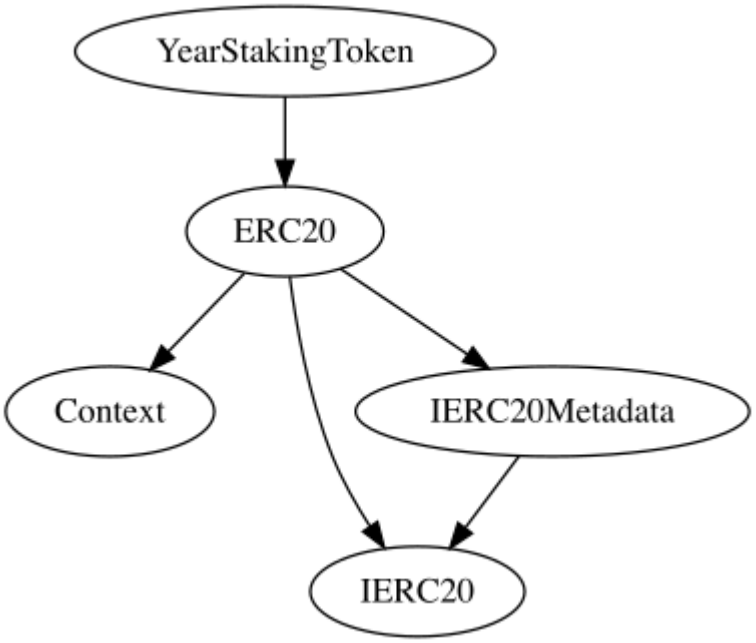
The team is advised to lock the pragma to ensure the stability of the codebase. The locked pragma version ensures that the contract will not be deployed with an unexpected version. An unexpected version may produce vulnerabilities and undiscovered bugs. The compiler should be configured to the lowest version that provides all the required functionality for the codebase. As a result, the project will be compiled in a well-tested LTS (Long Term Support) environment.

Functions Analysis

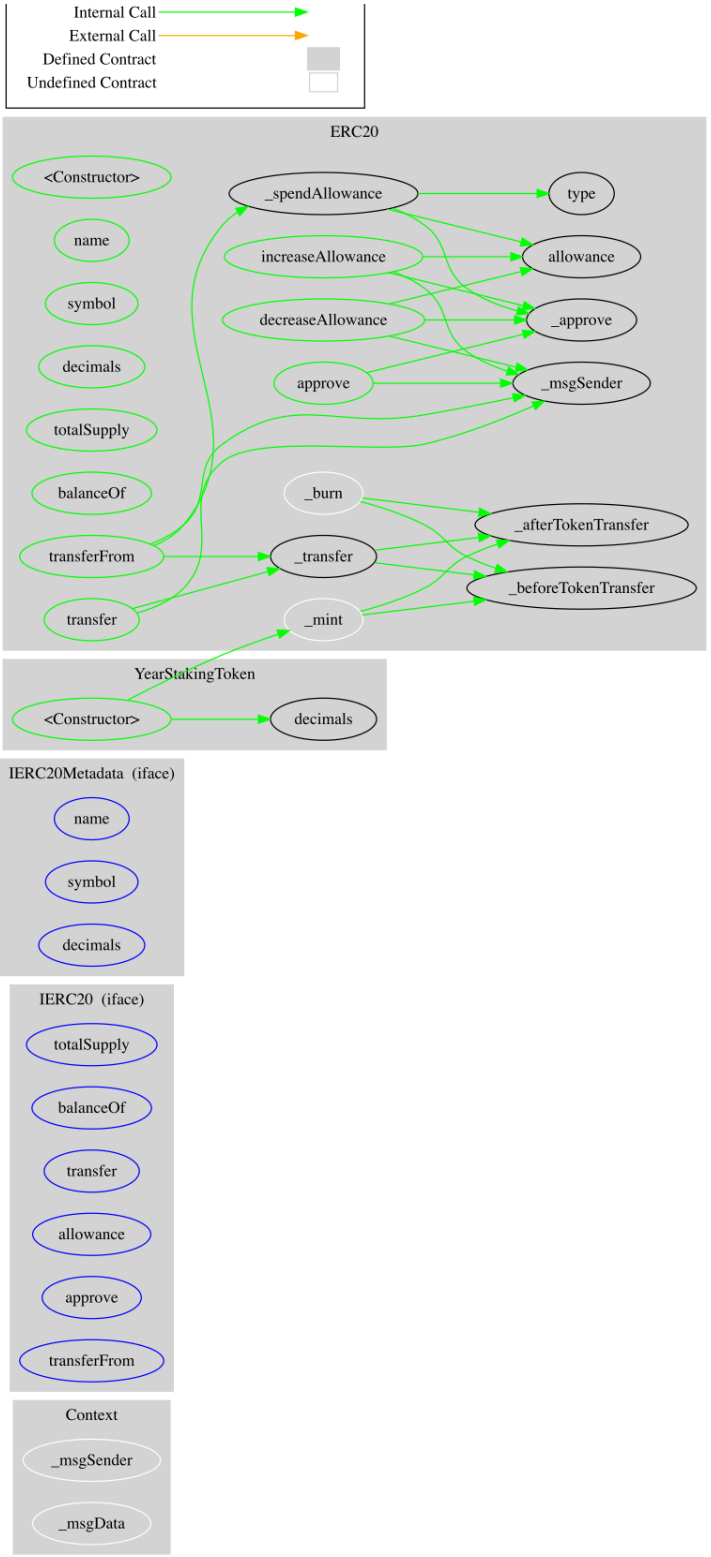
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
IERC20Metadata	Interface	IERC20		
	name	External		-
	symbol	External		-
	decimals	External		-
ERC20	Implementation	Context, IERC20, IERC20Metadata		

		Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	_transfer	Internal	✓	
	_mint	Internal	✓	
	_burn	Internal	✓	
	_approve	Internal	✓	
	_spendAllowance	Internal	✓	
	_beforeTokenTransfer	Internal	✓	
	_afterTokenTransfer	Internal	✓	
YearStakingToken	Implementation	ERC20		
		Public	✓	ERC20

Inheritance Graph



Flow Graph


Internal Call
External Call
Defined Contract
Undefined Contract
ERC20
<Constructor>
name
symbol
decimals
totalSupply
balanceOf
transferFrom
transfer
_spendAllowance
increaseAllowance
decreaseAllowance
approve
_burn
_transfer
_mint
type
allowance
_approve
_msgSender
_afterTokenTransfer
_beforeTokenTransfer
YearStakingToken
<Constructor>
decimals
IERC20Metadata (iface)
name
symbol
decimals
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
Context
_msgSender
_msgData

Summary

The year Staking Token (YST) contract implements a token mechanism. This audit investigates security issues, business logic concerns, and potential improvements. YST is an interesting project that has a friendly and growing community. The Smart Contract analysis reported no compiler errors or critical issues. The contract Owner can access some admin functions that can not be used maliciously to disturb the users' transactions. The contract contains 0 decimals. While it does not pose a direct security threat, it can impact the usability, compatibility, and user experience of the token. Developers and token issuers should carefully assess whether the lack of decimal places aligns with their project's objectives and the needs of their users.

Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Cyberscope's prior written consent. This report is not nor should be considered an "endorsement" or "disapproval" of any particular project or team. This report is not nor should be regarded as an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Cyberscope to perform a security assessment. This document does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors' business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk Cyberscope's position is that each company and individual are responsible for their own due diligence and continuous security Cyberscope's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. You agree that your access and/or use including but not limited to any services reports and materials will be at your sole risk on an as-is where-is and as-available basis Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.

The Cyberscope team

https://www.cyberscope.io