Cyberscope

Audit Report

Tunnel

December 2023

Network BSC

Address 0xf0147183aA4E0A5332d6adb72f0B380611200179

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
●	RID	Redundant Interface Declaration	Unresolved
●	L09	Dead Code Elimination	Unresolved
●	L19	Stable Compiler Version	Unresolved

Table of Contents

Review

Contract Name	StandardBEP20
Compiler Version	v0.8.23+commit.f704f362
Optimization	200 runs
Explorer	https://bscscan.com/address/0xf0147183aa4e0a5332d6adb72f0b380611200179
Address	0xf0147183aa4e0a5332d6adb72f0b380611200179
Network	BSC
Symbol	TNL
Decimals	18
Total Supply	100,000,000,000

Audit Updates

Initial Audit	03 Dec 2023

Source Files

Filename	SHA256
StandardBEP20.sol	d24da063a281cdb628a3dca260037af1f329df71af3ac436634fb80dfb0a2148

Findings Breakdown

Critical	0
Medium	0
Minor / Informative	3

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	0	0	0
Medium	0	0	0	0
Minor / Informative	3	0	0	0

RID - Redundant Interface Declaration

Criticality	Minor / Informative
Location	StandardBEP20.sol#L197,255
Status	Unresolved

Description

The contract features an interface declaration that remains unused in the implementation. While this doesn't directly compromise the security or functionality of the contract, it introduces unnecessary complexity, potentially hindering comprehension. This increased complexity may lead to challenges in maintenance and could pose security risks over time.

```
interface IERC721Errors {}

interface IERC1155Errors {}
```

Recommendation

It's recommended to avoid using redundant interface declarations. This approach enhances code clarity and reduces unnecessary complexity in the system.

L09 - Dead Code Elimination

Criticality	Minor / Informative
Location	StandardBEP20.sol#L653
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 value) internal {
    if (account == address(0)) {
        revert ERC20InvalidSender(address(0));
    }
    _update(account, address(0), value);
}
```

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.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	StandardBEP20.sol#L12,40,141,307,390,418,735,760,781,812,831
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 ^0.8.20;
```

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

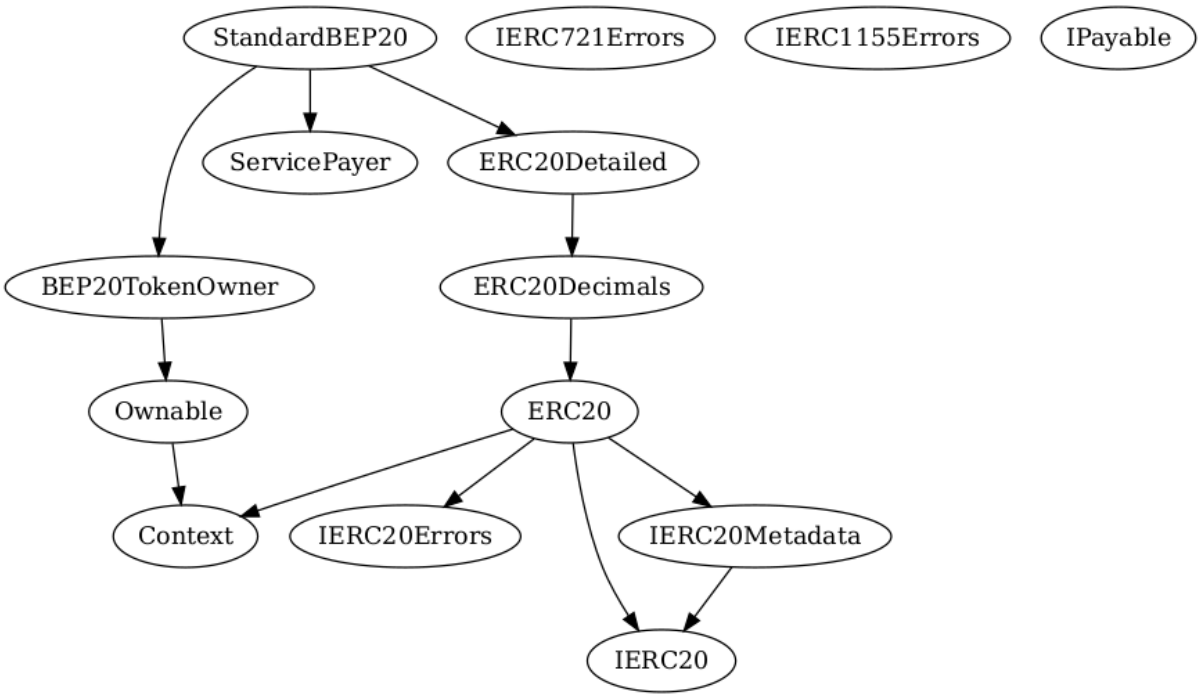
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	_checkOwner	Internal		
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	
IERC20Errors	Interface			
IERC721Errors	Interface			
IERC1155Errors	Interface			

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, IERC20Errors		
		Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-

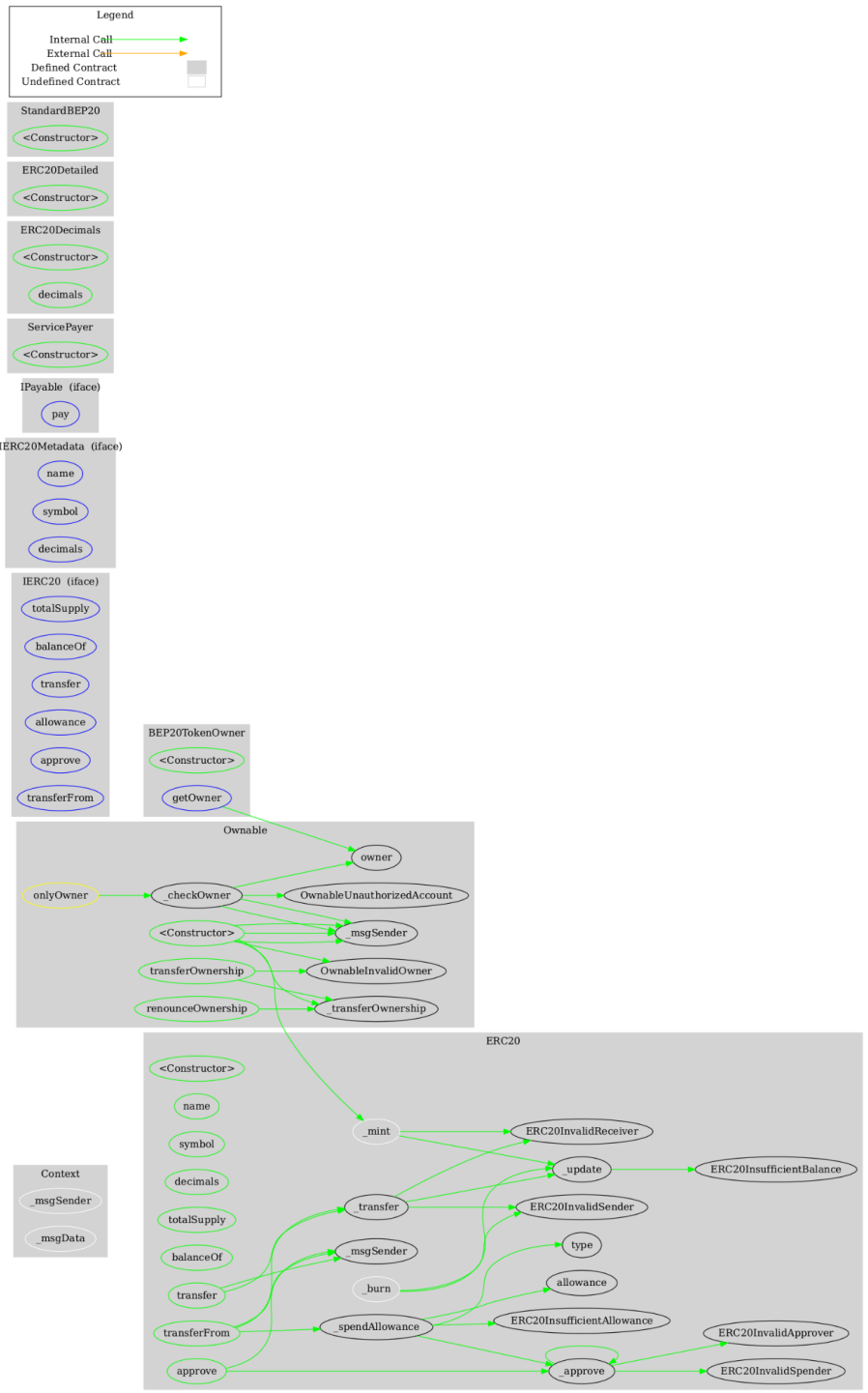
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	_transfer	Internal	✓	
	_update	Internal	✓	
	_mint	Internal	✓	
	_burn	Internal	✓	
	_approve	Internal	✓	
	_approve	Internal	✓	
	_spendAllowance	Internal	✓	
BEP20TokenOwner	Implementation	Ownable		
		Public	✓	Ownable
	getOwner	External		-
IPayable	Interface			
	pay	External	Payable	-
ServicePayer	Implementation			
		Public	Payable	-
ERC20Decimals	Implementation	ERC20		
		Public	✓	-

	decimals	Public		-
ERC20Detailed	Implementation	ERC20Decimals		
		Public	✓	ERC20 ERC20Decimals
StandardBEP20	Implementation	ERC20Detailed, BEP20TokenOwner, ServicePayer		
		Public	Payable	ERC20Detailed ServicePayer

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
StandardBEP20
<Constructor>
ERC20Detailed
<Constructor>
ERC20Decimals
<Constructor>
decimals
ServicePayer
<Constructor>
IPayable (iface)
pay
IERC20Metadata (iface)
name
symbol
decimals
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
BEP20TokenOwner
<Constructor>
getOwner
Ownable
owner
onlyOwner
_checkOwner
OwnableUnauthorizedAccount
_msgSender
<Constructor>
transferOwnership
OwnableInvalidOwner
renounceOwnership
transferOwnership
ERC20
<Constructor>
name
symbol
decimals
totalSupply
balanceOf
transfer
transferFrom
approve
_mint
ERC20InvalidReceiver
_update
ERC20InsufficientBalance
transfer
ERC20InvalidSender
_msgSender
type
_burn
allowance
_spendAllowance
ERC20InsufficientAllowance
ERC20InvalidApprover
_approve
ERC20InvalidSpender
Context
_msgSender
_msgData

Summary

Tunnel contract implements a token mechanism. This audit investigates security issues, business logic concerns, and potential improvements. The Tunnel Token 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 in a malicious way to disturb the users' transactions.

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