Cyberscope

A TAC Security Company

Audit Report

USDTR Token

April 2026

Network BSC

Address 0x855eC60F55900dc94Cc31f8A7E38a2A8F884a76a

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
	IDI	Immutable Declaration Improvement	Unresolved
	MTN	Misleading Token Naming	Unresolved
	ROF	Redundant Ownable Functionality	Unresolved
	RSML	Redundant SafeMath Library	Unresolved
	L01	Public Function could be Declared External	Unresolved

Table of Contents

Risk Classification

The criticality of findings in Cyberscope's smart contract audits is determined by evaluating multiple variables. The two primary variables are:

1. **Likelihood of Exploitation**: This considers how easily an attack can be executed, including the economic feasibility for an attacker.
2. **Impact of Exploitation**: This assesses the potential consequences of an attack, particularly in terms of the loss of funds or disruption to the contract's functionality.

Based on these variables, findings are categorized into the following severity levels:

1. **Critical**: Indicates a vulnerability that is both highly likely to be exploited and can result in significant fund loss or severe disruption. Immediate action is required to address these issues.
2. **Medium**: Refers to vulnerabilities that are either less likely to be exploited or would have a moderate impact if exploited. These issues should be addressed in due course to ensure overall contract security.
3. **Minor**: Involves vulnerabilities that are unlikely to be exploited and would have a minor impact. These findings should still be considered for resolution to maintain best practices in security.
4. **Informative**: Points out potential improvements or informational notes that do not pose an immediate risk. Addressing these can enhance the overall quality and robustness of the contract.

Severity	Likelihood / Impact of Exploitation
Critical	Highly Likely / High Impact
Medium	Less Likely / High Impact or Highly Likely/ Lower Impact
Minor / Informative	Unlikely / Low to no Impact

Review

Contract Name	BEP20USDTR
Compiler Version	v0.8.20+commit.a1b79de6
Optimization	200 runs
Explorer	https://bscscan.com/address/0x855ec60f55900dc94cc31f8a7e38a2a8f884a76a
Address	0x855ec60f55900dc94cc31f8a7e38a2a8f884a76a
Network	BSC
Symbol	USDTR
Decimals	18
Total Supply	27.011.985.555

Audit Updates

Initial Audit	17 Apr 2026

Source Files

Filename	SHA256
BEP20USDTR.sol	e5bfcad16694e657bd77c86ebc62398a4bdd2ef8d55b28fe4258ed0dda173d64

Findings Breakdown

Critical	0
Medium	0
Minor / Informative	5

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	0	0	0
Medium	0	0	0	0
Minor / Informative	5	0	0	0

IDI - Immutable Declaration Improvement

Criticality	Minor / Informative
Location	BEP20USDTR.sol#L103,104
Status	Unresolved

Description

The contract declares state variables that their value is initialized once in the constructor and are not modified afterwards. The `immutable` is a special declaration for this kind of state variables that saves gas when it is defined.

```
Shell
_decimals

_totalSupply
```

Recommendation

By declaring a variable as immutable, the Solidity compiler is able to make certain optimizations. This can reduce the amount of storage and computation required by the contract, and make it more gas-efficient.

MTN - Misleading Token Naming

Criticality	Minor / Informative
Location	BEP20USDTR.sol#L101
Status	Unresolved

Description

The contract declares a token with the name `Tether USD Bridged Wormhole` and the symbol `USDTR`. The name is identical to that of the Wormhole-bridged Tether USD token, a recognized cross-chain representation of USDT, while the symbol closely resembles `USDT` through the addition of a single character. Deploying a token with this name and symbol can cause confusion among users. This can also create potential issues when interfacing with third-party applications, exchanges, and wallets that recognize the original Wormhole-bridged USDT.

```
Shell
_name = "Tether USD Bridged Wormhole";
_symbol = "USDTR";
```

Recommendation

It is recommended to change the name and symbol of this token to clearly reflect its actual nature.

ROF - Redundant Ownable Functionality

Criticality	Minor / Informative
Location	BEP20USDTR.sol#L90,109
Status	Unresolved

Description

The contract implements functionality to define an owner. This functionality is normally implemented in contracts that need some form of authority and access control. However, excluding the `transferOwnership` and `renounceOwnership` there is no function in the contract that needs to be called only by the owner. Therefore the ownable functionality is redundant.

```
Shell
modifier onlyOwner() {}

function renounceOwnership() public onlyOwner {}

function transferOwnership(address newOwner) public
onlyOwner {}

function getOwner() external view returns (address) {
    return owner();

}
```

Recommendation

It is recommended to remove the ownable functionality to increase code optimization and readability.

RSML - Redundant SafeMath Library

Criticality	Minor / Informative
Location	BEP20USDTR.sol
Status	Unresolved

Description

SafeMath is a popular Solidity library that provides a set of functions for performing common arithmetic operations in a way that is resistant to integer overflows and underflows.

Starting with Solidity versions that are greater than or equal to 0.8.0, the arithmetic operations revert to underflow and overflow. As a result, the native functionality of the Solidity operations replaces the SafeMath library. Hence, the usage of the SafeMath library adds complexity, overhead and increases gas consumption unnecessarily in cases where the explanatory error message is not used.

```shell
library SafeMath {...}
```

Recommendation

The team is advised to remove the SafeMath library in cases where the revert error message is not used. Since the version of the contract is greater than `0.8.0` then the pure Solidity arithmetic operations produce the same result.

If the previous functionality is required, then the contract could exploit the `unchecked { ... }` statement.

Read more about the breaking change on https://docs.soliditylang.org/en/stable/080-breaking-changes.html#solidity-v0-8-0-breaking-changes.

L01 - Public Function could be Declared External

Criticality	Minor / Informative
Location	BEP20USDTR.sol#L78,83,153,158
Status	Unresolved

Description

A public function is a function that can be called from external contracts or from within the contract itself. An external function is a function that can only be called from external contracts, and cannot be called from within the contract itself.

It's generally a good idea to declare functions as external if they are only intended to be called from external contracts, as this can help make the contract's code easier to understand and maintain. Declaring a function as external can also help to improve the contract's performance and gas consumption.

```
Shell
function renounceOwnership() public onlyOwner {
function transferOwnership(address newOwner) public
onlyOwner {
function increaseAllowance(address spender, uint256
addedValue) public returns (bool) {

function decreaseAllowance(address spender, uint256
subtractedValue) public returns (bool) {
```

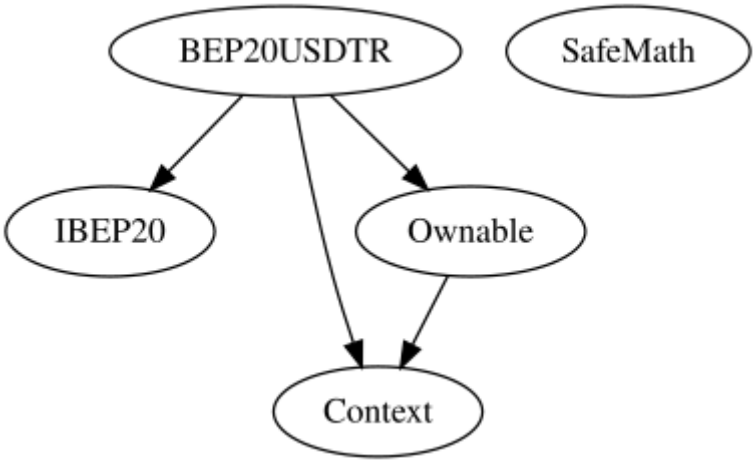
Recommendation

It's important to choose the appropriate visibility for each function based on how it is intended to be used. Declaring a function as external when it should be public, or vice versa can lead to unnecessary gas consumption.

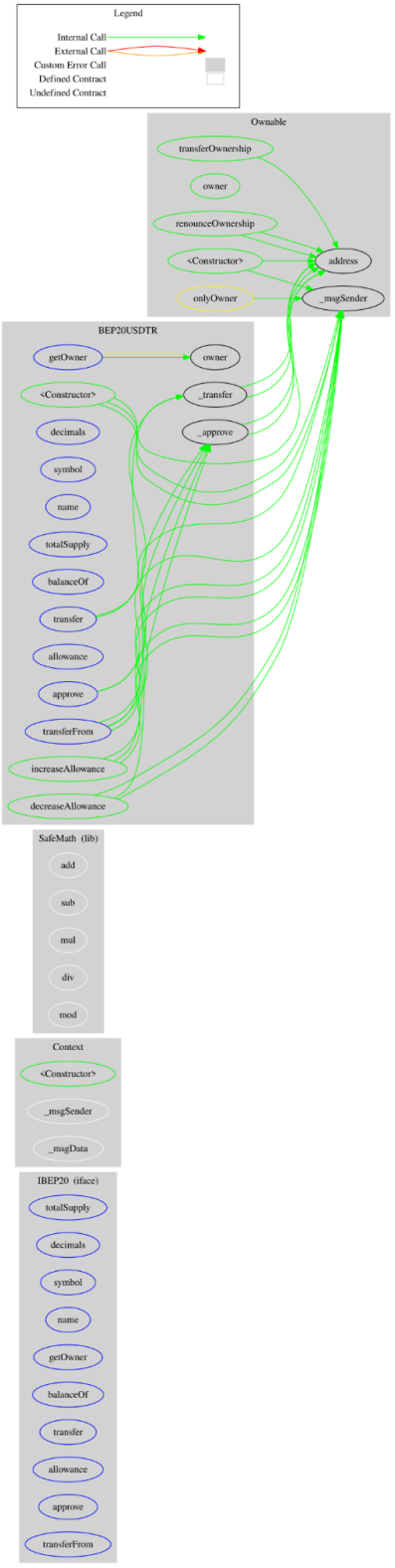
Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
BEP20USDTR	Implementation	Context, IBEP20, Ownable		
		Public	✓	-
	getOwner	External		-
	decimals	External		-
	symbol	External		-
	name	External		-
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	_transfer	Internal	✓	
	_approve	Internal	✓	

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Custom Error Call
Defined Contract
Undefined Contract
Ownable
transferOwnership
owner
renounceOwnership
<Constructor>
onlyOwner
address
_msgSender
BEP20USDTR
getOwner
owner
<Constructor>
_transfer
decimals
_approve
symbol
name
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
increaseAllowance
decreaseAllowance
SafeMath (lib)
add
sub
mul
div
mod
Context
<Constructor>
_msgSender
_msgData
IBEP20 (iface)
totalSupply
decimals
symbol
name
getOwner
balanceOf
transfer
allowance
approve
transferFrom

Summary

Tether USD Bridged Wormhole contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements..

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 TAC 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.

A **TAC Security** Company

The Cyberscope team

cyberscope.io