Cyberscope

Audit Report

EXIT Designer Token

February 2024

Network	BSC
Address	0xdEbd6e2da378784A69Dc6Ec99Fe254223b312287
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
	RRS	Redundant Require Statement	Unresolved
	RSML	Redundant SafeMath Library	Unresolved
	L09	Dead Code Elimination	Unresolved

Table of Contents

Review

Contract Name	Btoken
Compiler Version	v0.8.13+commit.abaa5c0e
Optimization	200 runs
Explorer	https://bscscan.com/address/0xdebd6e2da378784a69dc6ec99fe254223b312287
Address	0xdebd6e2da378784a69dc6ec99fe254223b312287
Network	BSC
Symbol	EXIT
Decimals	18
Total Supply	1,000,000,000

Audit Updates

Initial Audit	06 Feb 2024

Source Files

Filename	SHA256
Btoken.sol	48618b7259960abe48027ce90b31fe4511eccb6aca55c01b3bf847b02dc82d56

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

IDI - Immutable Declaration Improvement

Criticality	Minor / Informative
Location	Btoken.sol#L157
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.

```
_decimals
```

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.

RRS - Redundant Require Statement

Criticality	Minor / Informative
Location	Btoken.sol#L59,269
Status	Unresolved

Description

The contract utilizes a `require` statement within the `add` function aiming to prevent overflow errors. This function is designed based on the SafeMath library's principles. In Solidity version 0.8.0 and later, arithmetic operations revert on overflow and underflow, making the overflow check within the function redundant. This redundancy could lead to extra gas costs and increased complexity without providing additional security.

Furthermore, the `_burn` internal function utilizes a `require` statement to check for the zero address. However, this function is invoked only from the `burn` public function, which can be called only by the contract owner, since the `_burnFrom` internal function is not called by anywhere in the contract. As a result, the check for the zero address is redundant.

```
function add(uint256 a, uint256 b) internal pure returns (uint256) {
      uint256 c = a + b;
      require(c >= a, "SafeMath: addition overflow");
      return c;
}

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

      _balances[account] = _balances[account].sub(
            amount,
            "EXIT: burn amount exceeds balance"
      );
      _totalSupply = _totalSupply.sub(amount);
      emit Transfer(account, address(0), amount);
}
```

Recommendation

It is recommended to remove the `require` statement from the `add` function since the contract is using a Solidity pragma version equal to or greater than 0.8.0. By doing so, the contract will leverage the built-in overflow and underflow checks provided by the Solidity language itself, simplifying the code and reducing gas consumption. This change will uphold the contract's integrity in handling arithmetic operations while optimizing for efficiency and cost-effectiveness.

It is also recommended to remove the redundant `require` statement from the `_burn` function. This change will improve the code's clarity and gas usage.

RSML - Redundant SafeMath Library

Criticality	Minor / Informative
Location	Btoken.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.

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

Recommendation

The team is advised to remove the SafeMath library. 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/v0.8.16/080-breaking-changes.html#solidity-v0-8-0-breaking-changes.

L09 - Dead Code Elimination

Criticality	Minor / Informative
Location	Btoken.sol#L292
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 _burnFrom(address account, uint256 amount) internal {
    _burn(account, amount);
    _approve(
        account,
        _msgSender(),
        _allowances[account][_msgSender()].sub(
            amount,
            "EXIT: burn amount exceeds allowance"
        )
    );
}
```

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.

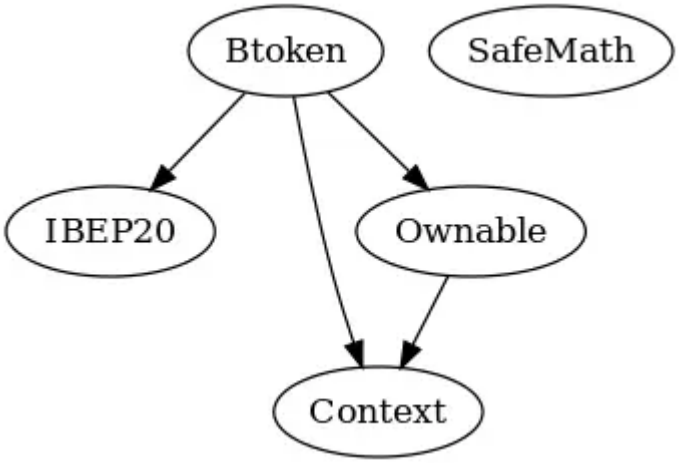
Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
IBEP20	Interface			
	totalSupply	External		-
	decimals	External		-
	symbol	External		-
	name	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
Context	Implementation			
		Public	✓	-
	_msgSender	Internal		
	_msgData	Internal		
SafeMath	Library			
	add	Internal		
	sub	Internal		

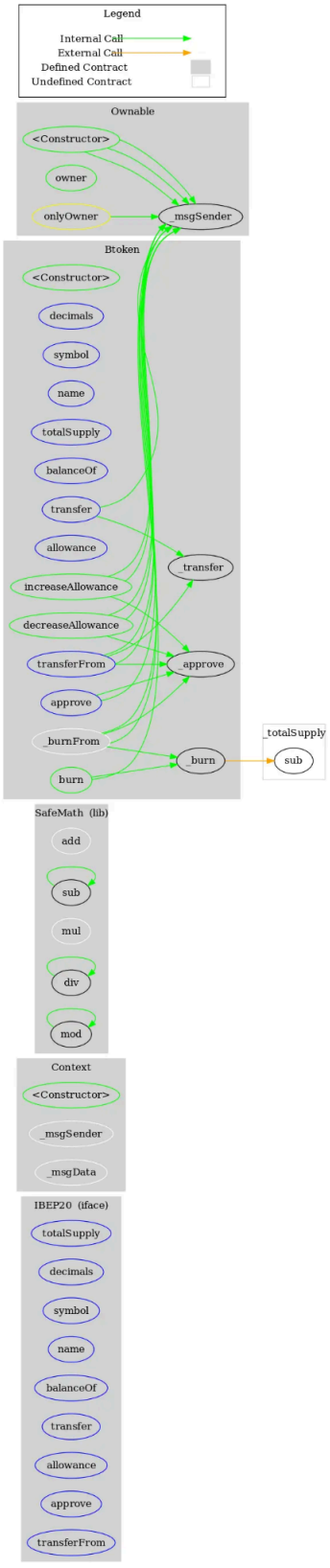
	sub	Internal		
	mul	Internal		
	div	Internal		
	div	Internal		
	mod	Internal		
	mod	Internal		
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
Btoken	Implementation	Context, IBEP20, Ownable		
		Public	✓	-
	decimals	External		-
	symbol	External		-
	name	External		-
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
	increaseAllowance	Public	✓	-

	decreaseAllowance	Public	✓	-
	burn	Public	✓	onlyOwner
	_transfer	Internal	✓	
	_burn	Internal	✓	
	_approve	Internal	✓	
	_burnFrom	Internal	✓	

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
Ownable
<Constructor>
owner
onlyOwner
_msgSender
Btoken
<Constructor>
decimals
symbol
name
totalSupply
balanceOf
transfer
allowance
increaseAllowance
decreaseAllowance
transferFrom
approve
_burnFrom
burn
_transfer
_approve
_burn
_totalSupply
sub
SafeMath (lib)
add
sub
mul
div
mod
Context
<Constructor>
_msgSender
_msgData
IBEP20 (iface)
totalSupply
decimals
symbol
name
balanceOf
transfer
allowance
approve
transferFrom

Summary

EXIT Designer Token contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. EXIT Designer Token is an interesting project that has a friendly and growing community. The Smart Contract analysis reported no compiler error 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