Cyberscope

A TAC Security Company

Audit Report

JETUSD Token

January 2026

Network	BSC
Address	0x61107715903b52c1f4191Ef90B3b9E21748aF41f
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
●	RGF	Redundant getOwner Function	Unresolved
●	MLI	Missing License Identifier	Unresolved
●	OPSV	Outdated Pragma Solidity Version	Unresolved
●	ROF	Redundant Ownership Functionality	Unresolved
●	L09	Dead Code Elimination	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	JETUSD
Compiler Version	v0.5.16+commit.9c3226ce
Optimization	200 runs
Explorer	https://bscscan.com/address/0x61107715903b52c1f4191ef90b3b9e21748af41f
Address	0x61107715903b52c1f4191ef90b3b9e21748af41f
Network	BSC
Symbol	JETUSD
Decimals	18
Total Supply	99,999,999,999

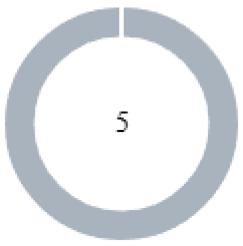
Audit Updates

Initial Audit	26 Jan 2026

Source Files

Filename	SHA256
JETUSD.sol	33bfea99c05d113bda63ba699fa644b1652af8402960f75d960ff52e519ae2ad

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

RGF - Redundant getOwner Function

Criticality	Minor / Informative
Location	JETUSD.sol#L364
Status	Unresolved

Description

The contract exposes the owner address via the inherited `owner()` function from Ownable. The JETUSD `getOwner()` function returns the same value by simply calling `owner()`. As a result, `getOwner()` is functionally redundant.

Shell
```
  function getOwner() external view returns
(address) {

     return owner()
```

Recommendation

The team is advised to remove redundant getter functions where possible to reduce bytecode size and keep the contract interface minimal.

MLI - Missing License Identifier

Criticality	Minor / Informative
Status	Unresolved

Description

The contract under review is missing an explicit SPDX license identifier, which is a crucial component for ensuring the legal clarity and compliance of the code. The license identifier specifies the terms under which the code can be used, modified, and redistributed, providing essential guidance to developers and end-users. Its absence creates ambiguity regarding the rights and obligations associated with the code, potentially leading to legal disputes or misuse.

Recommendation

The team is recommended to add an SPDX license identifier to the top of the smart contract to clearly specify the terms under which the code can be used, modified, and distributed.

OPSV - Outdated Pragma Solidity Version

Criticality	Minor / Informative
Location	JETUSD.sol#L5
Status	Unresolved

Description

The contract is using an outdated version of Solidity. Outdated versions of Solidity lack optimizations, and improvements found in more recent versions, which could affect the performance and efficiency of the contract.

```
Shell
pragma solidity 0.5.16;
```

Recommendation

The team should consider using a more recent version of Solidity for the reasons mentioned above.

ROF - Redundant Ownership Functionality

Criticality	Minor / Informative
Location	JETUSD.sol
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 {
```

Recommendation

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

L09 - Dead Code Elimination

Criticality	Minor / Informative
Location	JETUSD.sol#L460,476
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.

Shell
```
function _burn(address account, uint256 amount) internal {
    require(account != address(0), "BEP20: burn from the
zero address");

    _balances[account] = _balances[account].sub(amount,
"BEP20: burn amount exceeds balance");
    _totalSupply = _totalSupply.sub(amount);
    emit Transfer(account, address(0), amount);
  }

function _burnFrom(address account, uint256 amount)
internal {
    _burn(account, amount);
    _approve(account, _msgSender(),
_allowances[account][_msgSender()].sub(amount, "BEP20: burn
amount exceeds allowance"));
   }
```

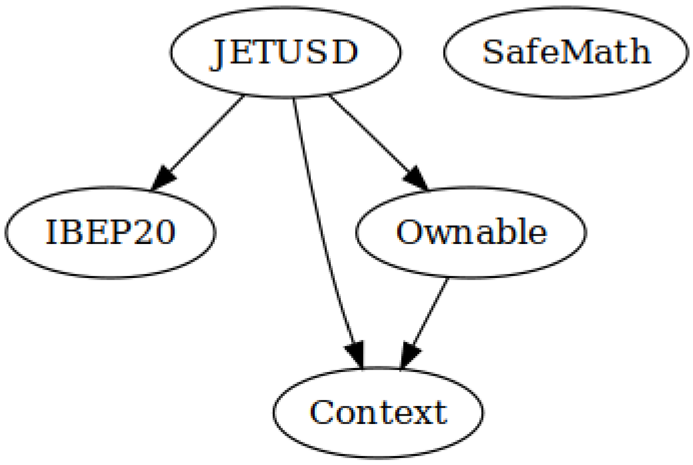
Recommendation

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.

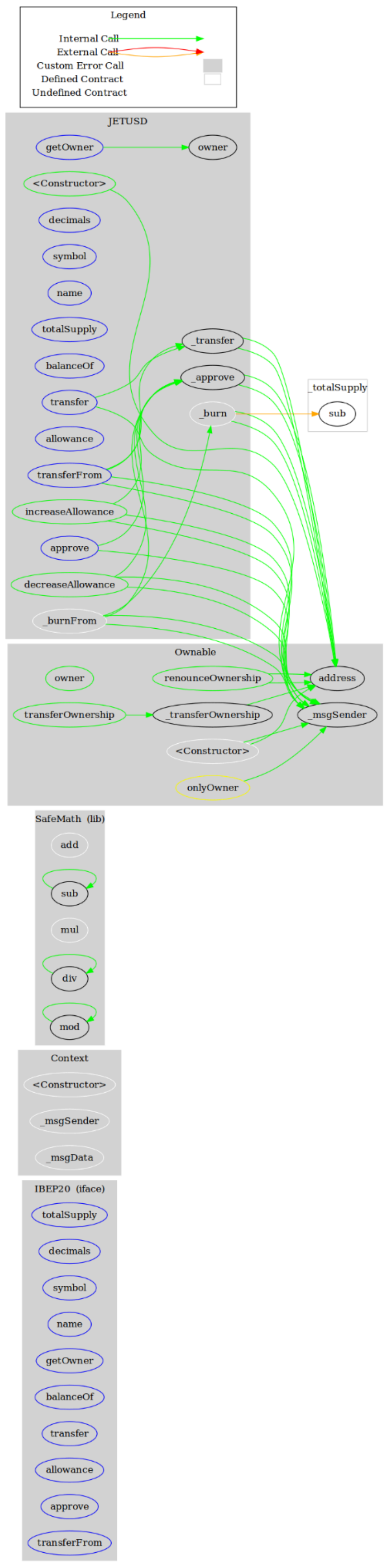
Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
JETUSD	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	✓	
	_burn	Internal	✓	
	_approve	Internal	✓	
	_burnFrom	Internal	✓	

Inheritance Graph



Flow Graph


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

Summary

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

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