Assure DeFi®

THE VERIFICATION **GOLD STANDARD**

Security Assessment

BSTRToken

Date: 10/06/2025

Audit Status: FAIL

Audit Edition: Advanced

ASSURE DEFI®

THE VERIFICATION **GOLD STANDARD**

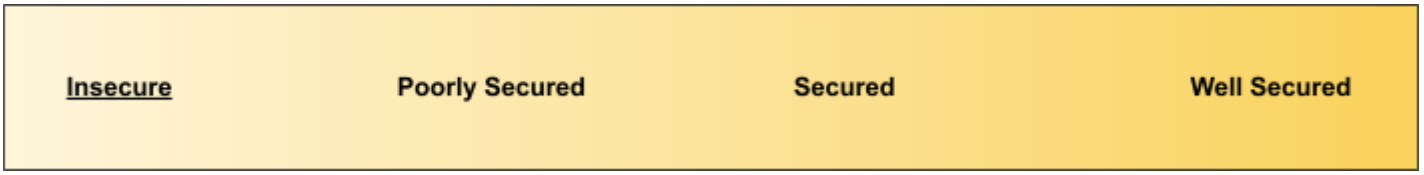
Risk Analysis

Vulnerability summary

Classification	Description
High	High-level vulnerabilities can result in the loss of assets or manipulation of data.
Medium	Medium-level vulnerabilities can be challenging to exploit, but they still have a considerable impact on smart contract execution, such as allowing public access to critical functions.
Low	Low-level vulnerabilities are primarily associated with outdated or unused code snippets that generally do not significantly impact execution, sometimes they can be ignored.
Informational	Informational vulnerabilities, code style violations, and informational statements do not affect smart contract execution and can typically be disregarded.

Executive Summary

According to the Assure assessment, the Customer's smart contract is **Insecure.**



Scope

Target Code And Revision

For this audit, we performed research, investigation, and review of the BSTRToken contracts followed by issue reporting, along with mitigation and remediation instructions outlined in this report.

Target Code And Revision

Project	Assure
Language	Solidity
Codebase	BSTRToken.sol [SHA256] - 57dcab963657c4a3361335bb3532bcdf41436e1bd6ec6c244adeea55ba5b7d94
Audit Methodology	Static, Manual

Attacks made to the contract

In order to check for the security of the contract, we tested several attacks in order to make sure that the contract is secure and follows best practices.

Category	Item
Code review & Functional Review	• Compiler warnings. • Race conditions and Reentrancy. Cross-function race conditions. • Possible delays in data delivery. • Oracle calls. • Front running. • Timestamp dependence. • Integer Overflow and Underflow. • DoS with Revert. • DoS with block gas limit. • Methods execution permissions. • Economy model. • Private user data leaks. • Malicious Event log. • Scoping and Declarations. • Uninitialized storage pointers. • Arithmetic accuracy. • Design Logic. • Cross-function race conditions. • Safe Zeppelin module. • Fallback function security. • Overpowered functions / Owner privileges

AUDIT OVERVIEW

1. Compilation & Ownership Initialization

Contract: BSTRToken

Function: constructor

Issue: Uses Ownable(msg.sender) but OpenZeppelin v4's Ownable has no constructor parameter—this won't compile or initialize ownership properly.

Recommendation: Remove the erroneous constructor argument; rely on OZ's default Ownable() which sets owner = msg.sender.

2. Unsafe ETH Transfer in Constructor

Contract: BSTRToken

Function: constructor

Issue: Unbounded feeReceiver_ transfer: payable(feeReceiver_).transfer(msg.value) may revert if receiver's fallback uses >2300 gas.

Recommendation: Use the Checks-Effects-Interactions pattern with .call{value: …}("") and handle the return boolean; or require a simple EOA that can't revert.

3. Unused taxRateUpdater Role

Contract: BSTRToken

Function: setTaxRates

Issue: Only onlyOwner enforced, but variable taxRateUpdater is never used—lack of owner/taxRateUpdater distinction means no delegated fee set rights.

Recommendation: Either remove taxRateUpdater entirely (dead code) or add logic so only taxRateUpdater can call setTaxRates, with an event, to honor the intended delegation.

MEDIUM

1. Unchecked ETH Transfer in Constructor

Contract: TaxableToken

Function: _update(Hooks)

Issue: Potential reentrancy: fees processing can trigger external DEX calls within a token transfer, without a reentrancy guard on _update itself.

Recommendation: Add nonReentrant to entry points that ultimately call _update, or restructure so that external calls occur after state updates and emit no further transfers.

2. Gas-Limit DoS in Fee Distribution

Contract: TaxDistributor

Function: distributeFees()

Issue: Large collector lists can exceed gas limits and DoS distribution.

Recommendation: Impose a max collector limit or implement batch distributions.

LOW

No low severity issues were found.

INFORMATIONAL

1. Custom Decimals Documentation

Contract: BSTRToken

Issue: decimals() returns 9, differing from the usual 18. This can confuse integrators if not documented.

Recommendation: Clearly document in the README and emit a DecimalsChanged event (if upgradeable) or provide a public constant.

2. Missing Events on State Changes

Contract: BSTRToken

Function: SetAutoProcessFees etc

Issue: Many admin functions lack corresponding events (e.g. AutoProcessFeesUpdated, CollectorAdded, etc.), impairing off-chain monitoring.

Recommendation: Emit a specific event in each setter to log the new state/value, aiding transparency and on-chain observability.

Technical Findings Summary

Findings

Vulnerability Level	Total	Pending	Not Apply	Acknowledged	Partially Fixed	Fixed
High	3					
Medium	2					
Low	0					
Informational	2					

Assessment Results

Score Results

Review	Score
Global Score	**70/100**
Assure KYC	https://projects.assuredefi.com/project/drvn-labo
Audit Score	65/100

The Following Score System Has been Added to this page to help understand the value of the audit, the maximum score is 100, however to attain that value the project must pass and provide all the data needed for the assessment. Our Passing Score has been changed to 84 Points for a higher standard, if a project does not attain 85% is an automatic failure. Read our notes and final assessment below. The Global Score is a combination of the evaluations obtained between having or not having KYC and the type of contract audited together with its manual audit.

Audit FAIL

Following our comprehensive security audit of the token contract for the BSTRToken project, the project did not fulfill the necessary criteria required to pass the security audit.

Disclaimer

Assure Defi has conducted an independent security assessment to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the reviewed code for the scope of this assessment. This report does not constitute agreement, acceptance, or advocating for the Project, and users relying on this report should not consider this as having any merit for financial adBSTRToken in any shape, form, or nature. The contracts audited do not account for any economic developments that the Project in question may pursue, and the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude, and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are entirely free of exploits, bugs, vulnerabilities or deprecation of technologies.

All information provided in this report does not constitute financial or investment adBSTRToken, nor should it be used to signal that any person reading this report should invest their funds without sufficient individual due diligence, regardless of the findings presented. Information is provided ‘as is, and Assure Defi is under no covenant to audit completeness, accuracy, or solidity of the contracts. In no event will Assure Defi or its partners, employees, agents, or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions or actions with regards to the information provided in this audit report.

The assessment serBSTRTokens provided by Assure Defi are subject to dependencies and are under continuing development. You agree that your access or use, including but not limited to any serBSTRTokens, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies with high levels of technical risk and uncertainty. The assessment reports could include false positives, negatives, and unpredictable results. The serBSTRTokens may access, and depend upon, multiple layers of third parties.