Assure DeFi®

THE VERIFICATION **GOLD STANDARD**

# Security Assessment

# BSTRToken

Date: 10/06/2025

Audit Status: PASS

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| 🟡 Low | Low-level vulnerabilities are primarily associated with outdated or unused code snippets that generally do not significantly impact execution, sometimes they can be ignored.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 **Well Secured.**

| | | | |
|---|---|---|---|
| Insecure | Poorly Secured | Secured | Well Secured |

# 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<br><br>Fixed version: https://github.com/DRVN-Labo/BSTR-Token-v2/commit/f8f055a9bde5d74d5232f930db9d2e12c84e7361<br><br>Deployed address: https://basescan.org/address/0xBFC5cD421bBC91A2Ca976C4AB1754748634b7D41#code |
| **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.<br>● Race conditions and Reentrancy. Cross-function race conditions.<br>● Possible delays in data delivery.<br>● Oracle calls.<br>● Front running.<br>● Timestamp dependence.<br>● Integer Overflow and Underflow.<br>● DoS with Revert.<br>● DoS with block gas limit.<br>● Methods execution permissions.<br>● Economy model.<br>● Private user data leaks.<br>● Malicious Event log.<br>● Scoping and Declarations.<br>● Uninitialized storage pointers.<br>● Arithmetic accuracy.<br>● Design Logic.<br>● Cross-function race conditions.<br>● Safe Zeppelin module.<br>● Fallback function security.<br>● Overpowered functions / Owner privileges |

# AUDIT OVERVIEW

◆ ◆ ◆ **HIGH**

**1. Compilation & Ownership Initialization  [Fixed ✅]**

**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.

**Fix**: Removed that argument entirely and rely on OZ's built-in Ownable() (which sets owner = msg.sender).

**2. Unsafe ETH Transfer in Constructor [Fixed ✅]**

**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.

Fix: Now the contract forwards all gas and checks the return boolean. safe ETH transfer implemented.

**3. Unused taxRateUpdater Role  [Fixed ✅]**

**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.
**Fix**: Now the contract initialize taxRateUpdater = _msgSender() in the ctor, introduce onlyTaxRateUpdater and apply it to setTaxRates and also expose setTaxRateUpdater() under onlyOwner so the owner can delegate that role.

## MEDIUM

**1. Unchecked ETH Transfer in Constructor [Acknowledge ✅]**

**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 [Fixed ✅]**

**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.

**Fix**: Max 50 entries.

**3. Reentrancy in Fee Distribution [Fixed ✅]**

**Contract**: TaxDistributor

**Function**: _distributeFees

**Issue**: TaxDistributor._distributeFees makes external calls without nonReentrant.

**Recommendation**: Replace low-level calls with SafeERC20.safeTransfer, wrap _distributeFees in nonReentrant (or adopt a pull-over-push model).

**Fix**: Reentrancy Guard added.

## LOW

**1. Zero-Share Configuration Can Lock Funds [Acknowledge]**

**Contract**: BSTRToken

**Issue**: SetFeeConfiguration allows (burn+liquidity+collectors)==0. In that state, _processFees will:

Swap/liquify portion (liquidityRatio=0 → skip)

if feesInToken, call _distributeFees with all tokens as collector amount, but shareSum=0 means no collectors get anything → tokens stay in contract

So if the owner accidentally (for example misconfigured fee update), funds are temporarily stuck (not lost, but unusable) and requires manual intervention (owner must reset shares)

**Recommendation**: Disallow a zero-sum share config entirely (i.e. require burn+liq+collectors == FEE_PRECISION) or, in _processFees, detect a zero total share and auto-route all fees to liquidity or burn instead of a no-op.

◇ ◆ ◆ **INFORMATIONAL**

**1. Custom Decimals Documentation [Fixed ✅]**

**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.

**Fix**: Public constant added.

**2. Missing Events on State Changes [Acknowledge ✅]**

**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 | | | | | 3 |
| 🟠 Medium | 3 | | | 1 | | 2 |
| 🟡 Low | 1 | | | 1 | | |
| 🟢 Informational | 2 | | | 1 | | 1 |

# Assessment Results

## Score Results

| Review | Score |
|---|---|
| **Global Score** | **90/100** |
| Assure KYC | [https://projects.assuredefi.com/project/drvn-labo](https://projects.assuredefi.com/project/drvn-labo) |
| Audit Score | 85/100 |

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.

## <u>Audit PASS</u>

Following our comprehensive security audit of the token contract for the BSTRToken project, the project did meet the necessary criteria required to pass the security audit.

## Disclaimer

ASSURE DEFI®
THE VERIFICATION **GOLD STANDARD**