Assure DeFi®

THE VERIFICATION GOLD STANDARD

VERIFIED BY ASSURE DEFI ®

INTEGRITY ✦ TRUST ✦ CREDIBILITY

Security Assessment

DRVN Labo

Date: 05/06/2025

Audit Status: FAIL

Audit Edition: Advanced

DRVN labo

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🟡 Low	Low-level vulnerabilities are primarily associated with outdated or unused code snippets that generally do not significantly impact execution, sometimes they can be ignored.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

Insecure	**Poorly Secured**	**Secured**	**Well Secured**

Scope

Target Code And Revision

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

Target Code And Revision

Project	Assure
Language	Solidity
Codebase	PENDING ADDRESS
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

◆ ◆ ◆ **HIGH**

1. Reentrancy Race in Auto-Swap

Contract: TaxableToken

Function: autoProcessFees race-condition

Issue: Simultaneous transfers may cause two swaps at once, leading to reentrancy or swap failures.

Recommendation: Add a reentrancy lock (e.g., ReentrancyGuard) around _processFees and check _inSwap before invoking.

2. Reentrancy in Manual Fee Distribution

Contract: BSTRToken

Function: distributeFees(uint256 amount, bool inToken)

Issue: No reentrancy protection: malicious collector can reenter.

Recommendation: Wrap _distributeFees in nonReentrant or use Checks-Effects-Interactions.

3. Inheritance Hook Ordering Issue

Contract: BSTRToken

Function: _update() override precedence

Issue: Inheritance linearization between ERC20Base._update and TaxableToken._update may not enforce intended fee order, potentially bypassing fees.

Recommendation: Explicitly call TaxableToken._update then ERC20Base._update (or vice versa), avoiding implicit super.

4. Use of tx.origin (Potential)

Contract: TaxableToken

Function: Reliance on msg.sender vs. tx.origin

Issue: IIf tx.origin is used, fee exclusion can be bypassed.

Recommendation: Review code, ensure only msg.sender is used for access/exclusion checks.

MEDIUM

1. Unchecked ETH Transfer in Constructor

Contract: BSTRToken

Function: Constructor (feeReceiver_ transfer)

Issue: Unchecked reliance on transfer in constructor may fail if feeReceiver_ is a non-payable contract or has a fallback that consumes >2300 gas.

Recommendation: Use Address.sendValue instead of transfer, and check feeReceiver_ != address(0).

2. Slippage Vulnerability in processFees

Contract: BSTRToken

Function: processFees()

Issue: External call to swap Router (_processFees) can be manipulated (e.g., flash loans, front-runs) causing unexpected slippage and potential fund loss.

Recommendation: Add slippage checks (minAmountOut ≤ getAmountsOut), emit swap events, and require the owner to compute minOut via on-chain query.

3. Rounding “Dust” in Fee Calculation

Contract: TaxableToken

Function: Fee calculation on transfer

Issue: Rounding-down in percentage fee computation can create “dust” tokens in the contract balance, eventually locking them.

Recommendation: Track truncation remainders and distribute or burn dust periodically.

4. 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.

5. Division by Zero in Share Sum

Contract: TaxDistributor

Function: Division by zero in share sum

Issue: If totalShares is zero, distribution reverts.

Recommendation: require(totalShares > 0, "Total shares must be non-zero"); in constructor and share updates

6. Residual Wei/Token Dust

Contract: TaxDistributor

Function: Unchecked loss of precision when distributing ETH

Issue: Remainders in wei accumulate in contract.

Recommendation: Send leftovers to the owner or burn, e.g., after loop: remainder = amount - sumDistributed.

7. Malicious Swap Router Change

Contract: BSTRToken

Function: setSwapRouter()

Issue: Changing to a malicious router can drain fees.

Recommendation: Emit event, enforce timelock delay, set pendingRouter, confirmSwapRouter().

8. Insufficient Slippage Check in processFees

Contract: TaxableToken

Function: _processFees() insufficient slippage check

Issue: Swap may execute at zero/outstanding slippage.

Recommendation: Compare minAmountOut against getAmountsOut and revert on low output, or refund tokens.

9. Unbounded numTokensToSwap

Contract: BSTRToken

Function: setNumTokensToSwap(uint256 amount)

Issue: No bounds; zero triggers swap on every tx, > totalSupply disables swap.

Recommendation: require(amount > 0 && amount <= totalSupply(), "Invalid threshold"); and emit NumTokensToSwapUpdated.

10. Unbounded Collector Growth

Contract: TaxDistributor

Function: Collector list can grow unbounded

Issue: Too many collectors cause distribution gas limit.

Recommendation: Impose a hard cap on collector count (e.g., 100) or allow chunked, batched distribution.

LOW

1. Residual Wei/Token Dust

Contract: TaxDistributor

Function: Unchecked loss of precision when distributing ETH

Issue: Remainders in wei accumulate in contract.

Recommendation: Send leftover to owner or burn, e.g., after loop: remainder = amount - sumDistributed.

2. Unvalidated Liquidity Owner

Contract: BSTRToken

Function: setLiquidityOwner(address newOwner)

Issue: No validation; can set to zero or malicious.

Recommendation: require(newOwner != address(0), "Invalid owner"); and emit LiquidityOwnerUpdated.

3. Missing Events on State Changes

Contract: TaxableToken

Function: No event emission when toggling autoProcessFees

Issue: State changes are not logged.

Recommendation: Emit AutoProcessFeesUpdated(bool old, bool new), and similar for other setters.

INFORMATIONAL

No informational issues were found.

Technical Findings Summary

Findings

Vulnerability Level	Total	Pending	Not Apply	Acknowledged	Partially Fixed	Fixed
🔴 High	4					
🟠 Medium	3					
🟡 Low	10					
🟢 Informational	0					

Assessment Results

Score Results

Review	Score
Global Score	**50/100**
Assure KYC	https://projects.assuredefi.com/project/drvn-labo
Audit Score	45/100

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 DRVN Labo 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 adDRVN Labo 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 adDRVN Labo, 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 serDRVN Labos 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 serDRVN Labos, 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 serDRVN Labos may access, and depend upon, multiple layers of third parties.