IBS POTS CONTRACT

Audit Report

Mon Mar 23 2026

contact@bitslab.xyz

https://twitter.com/scalebit_

ScaleBit

IBS POTS CONTRACT Audit Report

1 Executive Summary

1.1 Project Information

Description	The project is a dual-token system on BSC featuring configurable buy/sell tax on DEX trades and a deflationary mechanism that periodically burns tokens from the LP pair.
Type	Token
Auditors	jolyon, s3cunda
Timeline	Mon Mar 16 2026 - Mon Mar 23 2026
Languages	Solidity
Platform	Ethereum
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/hengmansolutions/IBS_POTS_CONTRACT
Commits	af5c097643e6519e8a01d42e7c2ff6f1004035e4 cfb40c075fdb158a664aa139db6db5c3209f8da1

1.2 Files in Scope

The following are the SHA1 hashes of the original reviewed files.

ID	File	SHA-1 Hash
CON	constant/constant.sol	d07dfd565ec49f3009b351be4fc1b5c9286ec588
TSY	TokenSystem.sol	3109a91d51c6a7fb35aa44d4ab9c8fd2f05c59f2
TOK	Token.sol	aa3fff9ab081a86c5b98146d4913509324585563
DIS	Distributor.sol	058403dbda1ef0d3d705b44ba75dcc6908d24a72

1.3 Issue Statistic

Item	Count	Fixed	Partially Fixed	Acknowledged
Total	7	6	1	0
Centralization	1	0	1	0
Critical	0	0	0	0
Major	0	0	0	0
Medium	2	2	0	0
Minor	4	4	0	0
Informational	0	0	0	0

1.4 ScaleBit Audit Breakdown

ScaleBit aims to assess repositories for security-related issues, code quality, and compliance with specifications and best practices. Possible issues our team looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Integer overflow/underflow
- Number of rounding errors
- Unchecked External Call
- Unchecked CALL Return Values
- Functionality Checks
- Reentrancy
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic issues
- Gas usage
- Fallback function usage
- tx.origin authentication
- Replay attacks
- Coding style issues

1.5 Methodology

The security team adopted the **"Testing and Automated Analysis"**, **"Code Review"** and **"Formal Verification"** strategy to perform a complete security test on the code in a way that is closest to the real attack. The main entrance and scope of security testing are stated in the conventions in the "Audit Objective", which can expand to contexts beyond the scope according to the actual testing needs. The main types of this security audit include:

(1) Testing and Automated Analysis

Items to check: state consistency / failure rollback / unit testing / value overflows / parameter verification / unhandled errors / boundary checking / coding specifications.

(2) Code Review

The code scope is illustrated in section 1.2.

(3) Audit Process

- Carry out relevant security tests on the testnet or the mainnet;
- If there are any questions during the audit process, communicate with the code owner in time. The code owners should actively cooperate (this might include providing the latest stable source code, relevant deployment scripts or methods, transaction signature scripts, exchange docking schemes, etc.);
- The necessary information during the audit process will be well documented for both the audit team and the code owner in a timely manner.

2 Summary

This report has been commissioned by IBS_POTS_CONTRACT to identify any potential issues and vulnerabilities in the source code of the IBS POTS CONTRACT smart contract, as well as any contract dependencies that were not part of an officially recognized library. In this audit, we have utilized various techniques, including manual code review and static analysis, to identify potential vulnerabilities and security issues.

During the audit, we identified 7 issues of varying severity, listed below.

ID	Title	Severity	Status
DIS-2	`setHolderAddress()` Does Not Synchronize isTaxExempt State	Medium	Fixed
DIS-8	Discussion on `Distributor` Contract	Minor	Fixed
TOK-3	Unnecessary Validation for Unsigned Integer Lower Bound	Minor	Fixed
TOK-4	The `setSellRates()` Emits Incorrect Previous Value in Event	Minor	Fixed
TOK-5	Lack of Two-Step Validation in Governance Transfer	Medium	Fixed
TSY-1	The `dailyBurn()` Lacks Cooldown	Minor	Fixed
TSY-7	Centralization Risks	Centralization	Partially Fixed

3 Participant Process

Here are the relevant actors with their respective abilities within the IBS POTS CONTRACT Smart Contract :

Governance

- `setBuyRates()` — Set the buy-side tax rate.
- `setSellRates()` — Set the sell-side tax rate.
- `setBurnRates()` — Set the LP deflation burn rate.
- `setTreasuryAddress()` — Set the address receiving trade tax.
- `setTaxTreasury()` — Set the address receiving trade tax.
- `setLpTreasury()` — Set the address receiving LP burn proceeds.
- `setHolderAddress()` — Update the holder address.
- `addWhitelist()` — Add an address to the tax-exempt list.
- `excludeTaxBatch()` — Batch update tax-exempt status for addresses.
- `removeWhitelist()` — Remove an address from the tax-exempt list.
- `setLongGovernanceList()` — Register or unregister a buy-side tax trigger address.
- `setShortGovernanceList()` — Register or unregister a sell-side tax trigger address.
- `blacklistBatch()` — Batch set blacklist status for addresses.
- `transferGovernance()` — Transfer governance to a new address.
- `grantTaxer()` — Grant `TAXER_ROLE` to an address.
- `grantBurner()` — Grant `BURNER_ROLE` to an address.
- `revokeTaxer()` — Revoke `TAXER_ROLE` from an address.
- `revokeBurner()` — Revoke `BURNER_ROLE` from an address.

Minter

- `mint()` — Mint tokens to a specified address.

Burner

- `dailyBurn()` — Remove tokens from LP pair: partially burn, partially redirect to LP treasury.

Taxer

- `transferTax()` — Deduct a tax from caller's balance and send to treasury.

User

- `transfer()` — Transfer tokens with automatic buy/sell tax.
- `transferFrom()` — Transfer tokens on behalf of another address with tax.

4 Findings

DIS-2 `setHolderAddress()` Does Not Synchronize isTaxExempt State

Severity: Medium

Status: Fixed

Code Location:

Distributor.sol

Descriptions:

In the constructor, the holder address is explicitly marked as tax-exempt via isTaxExempt[holder] = true. However, when setHolderAddress() updates the holder to a new address, it does not revoke the tax exemption from the old holder nor grant it to the new one.

TokenSystem.sol

```
function setHolderAddress(address holder_) external onlyGovernance {
    address _cur = holder;
    holder = holder_;
    emit HolderChanged(_cur, holder);
}
```

Suggestion:

It is recommended to update setHolderAddress() to atomically synchronize the tax-exempt status.

Resolution:

The team adopted our advice and fixed this issue by revoking the tax exemption from the old holder and granting it to the new one in setHolderAddress(), which can be found at 010ed8d55f027ca3d2bb66f13bf278a1aa325944.

DIS-8 Discussion on `Distributor` Contract

Severity: Minor

Status: Fixed

Code Location:

Distributor.sol;

TokenSystem.sol

Descriptions:

The `Distributor` contract is deployed by `TokenSystem` in its constructor and configured for use, but **no on-chain logic ever interacts with it**. This raises questions about its purpose and whether it represents dead code.

Suggestion:

The project team should clarify the intended role of the Distributor. If it is intended for future use or external integration, this should be documented. If it is leftover from a refactor or no longer needed, consider removing the Distributor deployment and its tax-exempt entry to reduce gas cost and remove unnecessary code.

Resolution:

The team adopted our advice and fixed this issue by remove the `Distributor` contract and relative code, which can be found at 010ed8d55f027ca3d2bb66f13bf278a1aa325944.

TOK-3 Unnecessary Validation for Unsigned Integer Lower Bound

Severity: Minor

Status: Fixed

Code Location:

Token.sol;

TokenSystem.sol

Descriptions:

Multiple functions contain the condition value < 0 where value is of type uint256. Since uint256 is an unsigned integer, it can never be negative, making these checks always evaluate to false. This results in dead code that wastes gas and reduces readability.

Affected functions:

- Token.setBuyRates()
- Token.setSellRates()
- TokenSystem.dailyBurn()
- TokenSystem.setBuyRates()
- TokenSystem.setSellRates()
- TokenSystem.setBurnRates()

Suggestion:

It is recommended to either replace `< 0` with `== 0` if a zero-value check is intended.

Resolution:

The team adopted our advice and fixed this issue by removing the redundant < 0 checks from all affected functions, which can be found at 010ed8d55f027ca3d2bb66f13bf278a1aa325944.

TOK-4 The `setSellRates()` Emits Incorrect Previous Value in Event

Severity: Minor

Status: Fixed

Code Location:

Token.sol

Descriptions:

Token.sol

```
function setSellRates(uint256 _taxRate) external onlyGovernance {
    if (_taxRate < 0 || _taxRate > BPS_100) revert InvalidTaxRate(_taxRate);

    uint256 _cur = longGovernanceRatio;
    shortGovernanceRatio = _taxRate;
    emit SellRateChanged(_cur, shortGovernanceRatio);
}
```

In `setSellRates()`, the cached previous value `_cur` is read from `longGovernanceRatio` (the buy tax rate) instead of `shortGovernanceRatio` (the sell tax rate). As a result, the SellRateChanged event logs the buy rate as the old sell rate.

Suggestion:

It is recommended to cache the correct previous value before updating the state variable.

Resolution:

The team adopted our advice and fixed this issue by correctly reading _cur from shortGovernanceRatio instead of longGovernanceRatio in setSellRates(), which can be found at 010ed8d55f027ca3d2bb66f13bf278a1aa325944.

TOK-5 Lack of Two-Step Validation in Governance Transfer

Severity: Medium

Status: Fixed

Code Location:

Token.sol;

TokenSystem.sol

Descriptions:

The `transferGovernance` function executes the transfer of governance rights in a single transaction. If the target address is incorrect (e.g., due to typos, incorrectly copied addresses, or malicious input), governance rights will be permanently lost, rendering the contract unmanageable.

TokenSystem.sol/Token.sol

```
function transferGovernance(address _governance) external onlyGovernance {
      address _cur = governance;
      governance = _governance;
      emit GovernanceChanged(_cur, governance);
  }
```

Suggestion:

It is recommended to implement a two-step "nominate-confirm" process to mitigate operational risks.

Resolution:

The team adopted our advice and fixed this issue by replacing the custom single-step `transferGovernance` function with OpenZeppelin's `Ownable2Step`. The fix can be found at cfb40c075fdb158a664aa139db6db5c3209f8da1.

TSY-1 The dailyBurn() Lacks Cooldown

Severity: Minor

Status: Fixed

Code Location:

TokenSystem.sol

Descriptions:

The dailyBurn() directly removes tokens from the Uniswap pair , but there is no cooldown on this operation. Despite its name suggesting a once-per-day action, it can be invoked an unlimited number of times within the same day.

TokenSystem.sol

```
function dailyBurn(uint256 burnRate_) external onlyRole(BURNER_ROLE) {
    if (burnRate_ < 0 || burnRate_ > BPS_100)
        revert InvalidBurnRate(burnRate_);

    uint256 balance = this.balanceOf(uniswapPair);
    if (balance == 0) return;

    uint256 burnAmount = (balance * burnRate) / BPS_100;
    if (burnAmount == 0) return;

    uint256 blackAmount = (burnAmount * burnRate_) / BPS_100;
    if (blackAmount > 0) {
        super._update(uniswapPair, address(0), blackAmount);
    }

    uint256 lpAmount = burnAmount - blackAmount;
    if (lpAmount > 0) {
        super._update(uniswapPair, lpTreasury, lpAmount);
    }

    emit DailyBurn(balance, burnRate_, burnAmount, blackAmount, lpAmount);
```

```
        IUniswapV2Pair(uniswapPair).sync();
    }
```

Suggestion:

It is recommended to introduce a cooldown period (e.g., once per 24 hours) to enforce the intended daily invocation frequency of dailyBurn().

Resolution:

The team adopted our advice and fixed this issue by renamed `dailyBurn()` to `governanceBurn()` to remove the misleading "daily" semantics, which can be found at 010ed8d55f027ca3d2bb66f13bf278a1aa325944.

TSY-7 Centralization Risks

Severity: Centralization

Status: Partially Fixed

Code Location:

TokenSystem.sol;

Token.sol

Descriptions:

The contracts exhibit centralization risks across multiple privileged roles and addresses:

- **Single governance address**: The `governance` address in both Token.sol and TokenSystem.sol holds all administrative privileges. It can unilaterally set buy/sell/burn tax rates, change treasury addresses (holder, lpTreasury, taxTreasury), add/remove whitelist and blacklist entries, configure governance lists (longGovernanceList, shortGovernanceList, pairs, isTaxExempt), grant/revoke MINTER_ROLE, TAXER_ROLE, and BURNER_ROLE, and transfer governance to any address.
- **Minter and burner roles**: In Token.sol, MINTER_ROLE (granted to `minter` and `rbs`) allows unlimited token minting. In TokenSystem.sol, BURNER_ROLE can call `dailyBurn` to burn LP tokens. Both roles are controlled solely by governance.
- **No timelock on parameter changes**: Critical changes (tax rates, treasury addresses, blacklist) take effect immediately, leaving no time for users to react or for the community to challenge malicious updates.

Suggestion:

It is recommended that centralized addresses and roles adopt a multi-sig wallet combined with a timelock to mitigate centralization risks.

Resolution:

The team acknowledged this issue and state that:"

Rationale

The protocol implements an algorithmic non-stablecoin model where token minting and burning are governed by the following mechanisms:

1. Automated Execution: Token minting and burning are automatically executed through oracle price feeds and the Range-Bound Stability (RBS) system without manual intervention.
2. Restricted Permissions: The `MINTER_ROLE` and `BURNER_ROL` E are primarily granted to the RBS contract and bond contracts, which operate according to predefined economic parameters.
3. Parameter Management: The governance address manages protocol parameters (tax rates, treasury addresses, etc.), which is standard practice in algorithmic stablecoin protocols, similar to mature projects like Olympus DAO.

Risk Mitigation Measures

• Tax rates are hardcoded with upper limits to prevent being set to 100%

• Multi-signature wallet and timelock mechanisms will be introduced once the protocol reaches stability

• All critical parameter changes will be conducted under community oversight

In addition, the team performed the following on-chain actions to mitigate the centralization risk:

1. **LP Burn**

A substantial amount of IBS/USDT LP tokens was transferred to the dead address, which reduces the project's control over the LP position and mitigates the related centralization risk. tx1 0x11476de0d9dede594ccb01331c040243201d2a091821849ec248fca68cfee84c

2. **Ownership Renounced**

The IBS token contract executed `renounceOwnership()`, which removes the owner's ability to use owner-only administrative functions on that contract. tx2: 0xca7e688a2f966d39cb8274a100d7a5035e8bc99204daaa43d8de300cd668f75f

These actions materially mitigate the originally reported centralization risk.

Conclusion

While the current architecture grants permissions to privileged roles, these permissions primarily support automated economic mechanisms rather than arbitrary manipulation. The team did not introduce additional code modifications in the current version, but it has taken on-chain actions that partially mitigate the centralization risk identified in this finding.

Appendix 1

Issue Level

- **Informational** issues are often recommendations to improve the style of the code or to optimize code that does not affect the overall functionality.
- **Minor** issues are general suggestions relevant to best practices and readability. They don't post any direct risk. Developers are encouraged to fix them.
- **Medium** issues are non-exploitable problems and not security vulnerabilities. They should be fixed unless there is a specific reason not to.
- **Major** issues are security vulnerabilities. They put a portion of users' sensitive information at risk, and often are not directly exploitable. All major issues should be fixed.
- **Critical** issues are directly exploitable security vulnerabilities. They put users' sensitive information at risk. All critical issues should be fixed.

Issue Status

- **Fixed:** The issue has been resolved.
- **Partially Fixed:** The issue has been partially resolved.
- **Acknowledged:** The issue has been acknowledged by the code owner, and the code owner confirms it's as designed, and decides to keep it.

Appendix 2

Disclaimer

This report is based on the scope of materials and documents provided, with a limited review at the time provided. Results may not be complete and do not include all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your own risk. A report does not imply an endorsement of any particular project or team, nor does it guarantee its security. These reports should not be relied upon in any way by any third party, including for the purpose of making any decision to buy or sell products, services, or any other assets. TO THE FULLEST EXTENT PERMITTED BY LAW, WE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, IN CONNECTION WITH THIS REPORT, ITS CONTENT, RELATED SERVICES AND PRODUCTS, AND YOUR USE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NOT INFRINGEMENT.