



EXVUL

SMART CONTRACT **AUDIT REPORT**

BIBO Smart Contract

OCTOBER 2026

Contents

1. EXECUTIVE SUMMARY	3
1.1 Methodology	3
2. FINDINGS OVERVIEW	7
2.1 Project Info And Contract Address	7
2.2 Summary	8
2.3 Key Findings	9
3. DETAILED DESCRIPTION OF FINDINGS	10
3.1 Terminal Zero-Supply Rollover Can Leave Funded Holder Rewards Undistributable . . .	10
3.2 Enforced Valid-Zero Fee-Cap Reports Can Block Fee Collection and Future Emissions . .	12
Informational Findings	13
3.3 Reward Index Arithmetic Has No Explicit Protocol-Lifetime Bound	14
3.4 Creator Rewards Can Only Be Claimed as the Entire Accumulated Balance	15
4. CONCLUSION	16
5. APPENDIX	17
5.1 Basic Coding Assessment	17
5.2 Advanced Code Scrutiny	20
6. DISCLAIMER	22
7. REFERENCES	23
8. About Exvul Security	24

1. EXECUTIVE SUMMARY

ExVul Web3 Security reviewed the modified Solidity implementation in the **BIBO** audit repository. The review covered 34 modified or added logic files across the ve(3,3), Slipstream, and Launchpad packages, together with the security effects of three deleted upstream Solidity files. Twenty-two modified interfaces were reviewed as ABI and integration context. The audited logic delta contained 4,603 gross added and removed source lines.

The assessment focused on BIBO-specific changes at commit 425c581c4849 (the full commit is recorded in Section 2.1). Tests, scripts, dependencies, unchanged upstream logic, deployed state, and live-chain configuration were excluded. The complete UP baseline was unavailable, so UP reuse could not always be separated from BIBO-specific authorship.

The audit identified two Medium-severity findings and two Informational findings. A source-level remediation review performed on 09/10/2026 against `bibo-audit origin/main` commit 7bf33ffd755633ca826bab05042be7f1de6a19c6, whose release manifest identifies Genesis source dd6654232b95af5eb5d2288edafd8897638a4521, verified fixes for both Medium findings. The two Informational hardening observations remain acknowledged.

Independent remediation validation reproduced 701 passing ve(3,3) tests, 475 passing Launchpad tests, and 14 passing targeted Slipstream tests. An additional audit regression confirmed that a real Gauge forwards accrued fees during a valid zero-release distribution without changing emission-release accounting. These FIXED statuses apply to the reviewed source. Existing non-upgradeable deployments require migration or redeployment before the fixes become effective on-chain.

The outcome of the assessment outlined in chapter 3 provides the system's owners a full description of the findings, the associated risk rating for each finding, and detailed remediation guidance.

1.1 Methodology

To standardize the evaluation, we define the following terminology based on OWASP Risk Rating Methodology [10] which is the gold standard in risk assessment using the following risk models:

- **Likelihood:** represents how likely a particular vulnerability is to be uncovered and exploited in the wild.
- **Impact:** measures the technical loss and business damage of a successful attack.
- **Severity:** determine the overall criticality of the risk.

Likelihood can be: High, Medium and Low and impact are categorized into: High, Medium, Low, Informational. Severity is determined by likelihood and impact and can be classified into five categories accordingly: Critical, High, Medium, Low, Informational shown in table 1.1.

		Informational	Low	Medium	High
Likelihood	High	INFO	MEDIUM	HIGH	CRITICAL
	Medium	INFO	LOW	MEDIUM	HIGH
	Low	INFO	LOW	LOW	MEDIUM
		IMPACT			

Table 1.1 Overall Risk Severity

To evaluate the risk, we will be going through a list of items, and each would be labelled with a severity category. The audit was performed with a systematic approach guided by a comprehensive assessment list carefully designed to identify known and impactful security issues. If our tool or analysis does not identify any issue, the contract can be considered safe regarding the assessed item. For any discovered issue, we might further deploy contracts on our private test environment and run tests to confirm the findings. If necessary, we would additionally build a PoC to demonstrate the possibility of exploitation. The concrete list of check items is shown in Table 1.2.

- **Basic Coding Bugs:** We first statically analyze given smart contracts with our proprietary static code analyzer for known coding bugs, and then manually verify (reject or confirm) all the issues found by our tool.
- **Code and business security testing:** We further review business logics, examine system operations, and place DeFi-related aspects under scrutiny to uncover possible pitfalls and/or bugs.
- **Additional Recommendations:** We also provide additional suggestions regarding the coding and development of smart contracts from the perspective of proven programming practices.

Category	Assessment Item
Basic Coding Assessment	• Apply Verification Control • Authorization Access Control • Forged Transfer Vulnerability • Forged Transfer Notification • Numeric Overflow • Transaction Rollback Attack • Transaction Block Stuffing Attack • Soft Fail Attack • Hard Fail Attack • Abnormal Memo • Abnormal Resource Consumption • Secure Random Number

Advanced Source Code Scrutiny	• Asset Security • Cryptography Security • Business Logic Review • Source Code Functional Verification • Account Authorization Control • Sensitive Information Disclosure • Circuit Breaker • Blacklist Control • System API Call Analysis • Contract Deployment Consistency Check • Abnormal Resource Consumption
Additional Recommendations	• Semantic Consistency Checks • Following Other Best Practices

Table 1.2: The Full List of Assessment Items

To better describe each issue we identified, we categorize the findings with Common Weakness Enumeration (CWE-699) [14], which is a community-developed list of software weakness types to better delineate and organize weaknesses around concepts frequently encountered in software development.

2. FINDINGS OVERVIEW

2.1 Project Info And Contract Address

Project Name	Audit Time	Language
BIBO	07/10/2026 - 08/10/2026	Solidity

Repository

<https://github.com/bibodottrade/bibo-audit.git>

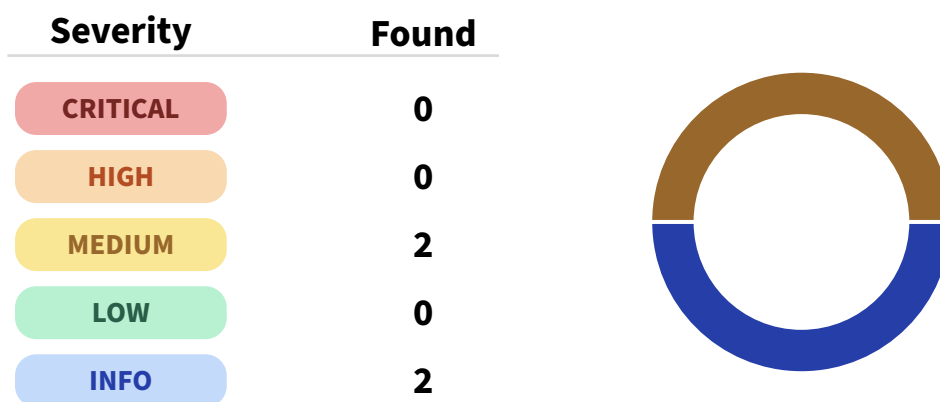
Audit Baseline Commit

425c581c48494fb2052b2e6691412ed633c9e309

Source-Level Remediation Review

09/10/2026 at 7bf33ffd755633ca826bab05042be7f1de6a19c6

2.2 Summary



2.3 Key Findings

Severity	Findings Title	Status
MEDIUM	Terminal Zero-Supply Rollover Can Leave Funded Holder Rewards Undistributable	Fixed
MEDIUM	Enforced Valid-Zero Fee-Cap Reports Can Block Fee Collection and Future Emissions	Fixed
INFO	Reward Index Arithmetic Has No Explicit Protocol-Lifetime Bound	Acknowledge
INFO	Creator Rewards Can Only Be Claimed as the Entire Accumulated Balance	Acknowledge

Table 2.3: Key Audit Findings

3. DETAILED DESCRIPTION OF FINDINGS

3.1 Terminal Zero-Supply Rollover Can Leave Funded Holder Rewards Undistributable

SEVERITY:**MEDIUM****STATUS:****Fixed****PATH:**`contract/launchpad/src/LaunchStakingRewards.sol``contract/launchpad/src/CreatorFeeVault.sol``contract/launchpad/src/LaunchFactory.sol`**DESCRIPTION:**

`LaunchStakingRewards` records rewards that elapse while `_totalSupply == 0` in `rollover`. When a user later stakes, the `updateReward` modifier executes before `_totalSupply` increases, so the empty-supply interval is banked but the banked amount is not restarted. The only reviewed transition that consumes `rollover` is a later positive `notifyRewardAmount` call from the immutable `rewardsDistribution` Vault.

```
modifier updateReward(address account) {
    uint256 applicable = lastTimeRewardApplicable();
    if (_totalSupply == 0) rollover += (applicable - lastUpdateTime) * rewardRate;
    rewardPerTokenStored = rewardPerToken();
    lastUpdateTime = applicable;
    if (account != address(0)) {
        rewards[account] = earned(account);
        userRewardPerTokenPaid[account] = rewardPerTokenStored;
    }
    _;
}

function notifyRewardAmount(uint256 reward) external onlyRewardsDistribution
    updateReward(address(0)) {
    if (reward == 0) revert ZeroAmount();
    if (_totalSupply == 0) revert NothingStaked();
    uint256 total = reward + rollover;
    rollover = 0;
    // The new stream is calculated from total.
}
```

The reachable state path is:

1. `CreatorFeeVault.forwardHoldersLeg` funds a positive stream while stake exists.

2. All users withdraw during the stream, leaving `_totalSupply == 0`.
3. A later checkpoint adds the empty interval's funded reward to `rollover`.
4. Eligible stake returns, but staking only checkpoints the banked amount and does not restart it.
5. The launch is already in a terminal funding state, so no later positive Vault notification is guaranteed.

The source supports two terminal funding states. A `POOL_LIVE` launch with creator fee tax permanently set to zero has closed curve trading and no creator-fee liability from its canonical pool. A `RESCUED` launch has swept and closed the curve without creating a canonical pool. In either state, once an already credited holders budget is exhausted, there may be no authenticated source for another positive notification.

Existing rollover tests do not cover this terminal sequence:

- `test_RollsOverRewardThatElapsesWithNothingStaked` mints one additional unit and calls `notifyRewardAmount(1)` before expecting the rollover to clear.
- `test_NotifyWhileEmptyIsRefusedAndTheTailJoinsTheNextOne` relies on a later positive notification after stake returns.

IMPACT:

Funded quote tokens remain held by `LaunchStakingRewards` and are represented by `rollover`, but eligible holders may have no protocol-reachable transition that distributes them. Stake principal and previously checkpointed account rewards remain claimable, and the liability remains token-backed. The issue is a persistent holder-reward liveness failure under terminal funding conditions, not insolvency or unauthorized extraction.

RECOMMENDATIONS:

Provide a narrowly authenticated rollover settlement or restart transition that does not require new fee income. The transition should checkpoint global and account liabilities before restarting the stream, define which stake cohort receives rewards accumulated during the empty interval, and prevent repeated calls from extending or duplicating the same liability.

Add regression tests covering a terminal launch with no further funding, stake returning after an empty-supply interval, repeated settlement attempts, rate truncation, and conservation between reward-token balance, accrued account rewards, active stream remainder, and `rollover`.

3.2 Enforced Valid-Zero Fee-Cap Reports Can Block Fee Collection and Future Emissions

SEVERITY:**MEDIUM****STATUS:****Fixed****PATH:**`contract/ve33/contracts/Voter.sol``contract/ve33/contracts/gauge-caps/GaugeCapController.sol``contract/ve33/contracts/gauges/Gauge.sol``contract/slipstream/contracts/gauge/CLGauge.sol`**DESCRIPTION:**

In Enforced mode, a valid cap report with `denominatorWeth == 0` can produce zero allowed emission. `Voter._distributeEnforced` records the nominal amount as withheld and returns when `release == 0` before calling `_notifyGaugeReward`.

```
emit GaugeCapAccounted(_epoch, _gauge, _nominal, requestedCumulative, release, withheld,
    quote.status);

if (release == 0) return;
if (!governanceCapActive && reportUsable) {
    bool marked = _markGaugeCapReportUsed(_gauge, _epoch);
    if (!marked) revert GaugeCapControllerMarkFailed();
}
_notifyGaugeReward(_gauge, release);
```

Both production Gauge implementations perform their automatic pool-fee checkpoint only inside the positive Voter notification path. Their manual notification variants deliberately omit fee collection.

```
function notifyRewardAmount(uint256 amount) external nonReentrant {
    address sender = _msgSender();
    if (sender != voter) revert NotVoter();
    if (amount == 0) revert ZeroAmount();
    _claimFees();
    _notifyRewardAmount(sender, amount);
}

function notifyRewardWithoutClaim(uint256 amount) external nonReentrant {
    address sender = _msgSender();
    if (sender != IVotingEscrow(ve).team()) revert NotTeam();
    if (amount == 0) revert ZeroAmount();
    _notifyRewardAmount(sender, amount);
}
```

The documented cap denominator is derived from fees already forwarded to the prior epoch's:

`FeesVotingReward.tokenRewardsPerEpoch`

Therefore, when a valid zero report makes `release` zero, the current pool fees are not automatically harvested and forwarded. The next honest report can still observe a zero prior-forwarded-fee denominator and repeat the same outcome.

The condition requires Enforced mode, a valid zero-denominator report, current positive pool fees and vote weight, no qualifying legacy or other positive notification, and no operator recovery. Component tests confirm that a zero denominator yields zero allowance, positive Gauge notifications harvest fees, and both `notifyRewardWithoutClaim` variants skip fee collection. Existing tests do not connect these behaviors across epochs into the feedback loop.

IMPACT:

Trading fees can remain backed in the pool or Gauge instead of reaching fee voters, while the affected Gauge's emission allocation remains withheld and can later be burned by epoch settlement. The condition is recoverable and does not redirect funds or cause irreversible principal loss, but normal fee distribution and emissions can remain stalled until a positive notification or authorized policy action interrupts the loop.

RECOMMENDATIONS:

Decouple the fixed-destination fee checkpoint from positive emission release. Execute it before the zero-release return, or expose a permissionless checkpoint whose pool, tokens, and reward receiver are all derived from immutable or Voter-registered state rather than caller input.

The fix should preserve the following invariants: fee rewards equal actual received tokens; repeated checkpoints cannot double-count fees; a fee-only checkpoint does not change emission reserve or released-emission accounting; and classic Gauge and CLGauge follow equivalent behavior. Add a cross-epoch regression test that begins with a valid zero report and current pool fees, performs zero-release distribution, builds the next report from forwarded fees, and verifies the cycle no longer persists.

Informational Findings

3.3 Reward Index Arithmetic Has No Explicit Protocol-Lifetime Bound

SEVERITY:

INFO

STATUS:

Acknowledge

PATH:`contract/launchpad/src/LaunchStakingRewards.sol`**DESCRIPTION:**

`rewardPerToken` multiplies elapsed time by `rewardRate` before passing the value to `Math.mulDiv`, and the Q128-scaled index accumulates over the lifetime of the staking contract. `notifyRewardAmount` limits the current rate against the contract's token balance, but the source does not state or enforce a protocol-wide bound on cumulative funding, supported reward-token supply, or contract lifetime.

```
function rewardPerToken() public view returns (uint256) {
    if (_totalSupply == 0) {
        return rewardPerTokenStored;
    }
    return rewardPerTokenStored
        + Math.mulDiv((lastTimeRewardApplicable() - lastUpdateTime) * rewardRate, Q128,
            _totalSupply);
}
```

Representative tests for six-decimal and eighteen-decimal reward tokens at the maximum launch supply pass. No admitted quote-token profile, supported budget, or plausible lifetime approached a harmful `uint256` bound. This is an unverified hardening observation rather than a confirmed vulnerability.

IMPACT:

No concrete failure is established for the supported configurations. If future token-admission or funding policy permits values outside the assumed range, checked arithmetic could revert during reward-index updates and interrupt staking, withdrawal, or reward claims until the state or implementation changes.

RECOMMENDATIONS:

Document and enforce maximum reward-token supply, per-stream and cumulative funding, and supported lifetime. Add property tests proving that `elapsed * rewardRate` and the cumulative Q128 index remain within `uint256` under those limits. Otherwise, restructure the index so every intermediate value has an explicit checked bound.

3.4 Creator Rewards Can Only Be Claimed as the Entire Accumulated Balance

SEVERITY:

INFO

STATUS:

Acknowledge

PATH:`contract/launchpad/src/CreatorFeeVault.sol``contract/launchpad/src/ProtocolFeeEscrow.sol`**DESCRIPTION:**

`CreatorFeeVault.claimCreator` always transfers the full `creatorClaimable` balance and does not allow the recipient to request a smaller amount. The analogous shared `ProtocolFeeEscrow` supports partial claims for compatibility with tokens that enforce a maximum amount per transfer.

```
function claimCreator() external nonReentrant returns (uint256 amount) {
    if (msg.sender != currentRecipient) revert NotRecipient();
    amount = creatorClaimable;
    if (amount == 0) revert ZeroAmount();
    creatorClaimable = 0;
    accountedBalance -= amount;
    quote.safeTransfer(msg.sender, amount);
    emit CreatorLegClaimed(msg.sender, amount);
}
```

The quote-token admission checks require exact-transfer and non-rebasing behavior, but no admitted quote token had a relevant per-transfer ceiling or a reachable creator balance above one. This is an unverified token-compatibility observation rather than a confirmed liveness defect.

IMPACT:

No affected admitted token or reachable locked balance was established. If a future admitted quote token rejects transfers above a fixed ceiling and `creatorClaimable` exceeds it, the creator could not claim through the current all-or-nothing function.

RECOMMENDATIONS:

Reject quote tokens with transfer ceilings during admission, or add a bounded partial-claim function. It should retain `nonReentrant`, validate the requested amount, reduce `creatorClaimable` and `accountedBalance` before transfer, and preserve the Vault's balance-to-liability invariant.

4. CONCLUSION

In this audit, we thoroughly analyzed **BIBO** smart contract implementation. The problems found are described and explained in detail in Section 3. The problems found in the audit have been communicated to the project leader. We therefore consider the audit result to be **PASSED**.

To improve this report, we greatly appreciate any constructive feedbacks or suggestions, on our methodology, audit findings, or potential gaps in scope/coverage.

5. APPENDIX

5.1 Basic Coding Assessment

5.1.1 Apply Verification Control

Description	The security of apply verification
Result	Not found
Severity	CRITICAL

5.1.2 Authorization Access Control

Description	Permission checks for external integral functions
Result	Not found
Severity	CRITICAL

5.1.3 Forged Transfer Vulnerability

Description	Assess whether there is a forged transfer notification vulnerability in the contract
Result	Not found
Severity	CRITICAL

5.1.4 Transaction Rollback Attack

Description	Assess whether there is transaction rollback attack vulnerability in the contract
Result	Not found
Severity	CRITICAL

5.1.5 Transaction Block Stuffing Attack

Description	Assess whether there is transaction blocking attack vulnerability
Result	Not found
Severity	CRITICAL

5.1.6 Soft Fail Attack Assessment

Description	Assess whether there is soft fail attack vulnerability
Result	Not found
Severity	CRITICAL

5.1.7 Hard Fail Attack Assessment

Description	Examine for hard fail attack vulnerability
Result	Not found
Severity	CRITICAL

5.1.8 Abnormal Memo Assessment

Description	Assess whether there is abnormal memo vulnerability in the contract
Result	Not found
Severity	CRITICAL

5.1.9 Abnormal Resource Consumption

Description	Examine whether abnormal resource consumption in contract processing
Result	Not found
Severity	CRITICAL

5.1.10 Random Number Security

Description	Examine whether the code uses insecure random number
Result	Not found
Severity	CRITICAL

5.2 Advanced Code Scrutiny

5.2.1 Cryptography Security

Description	Examine for weakness in cryptograph implementation
Result	Not found
Severity	HIGH

5.2.2 Account Permission Control

Description	Examine permission control issue in the contract
Result	Not found
Severity	MEDIUM

5.2.3 Malicious Code Behavior

Description	Examine whether sensitive behavior present in the code
Result	Not found
Severity	MEDIUM

5.2.4 Sensitive Information Disclosure

Description	Examine whether sensitive information disclosure issue present in the code
Result	Not found
Severity	MEDIUM

5.2.5 System API

Description	Examine whether system API application issue present in the code
Result	Not found
Severity	LOW

6. DISCLAIMER

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without ExVul'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 considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts ExVul to perform a security assessment. This report 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 in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing 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. ExVul's position is that each company and individual are responsible for their own due diligence and continuous security. ExVul'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.

7. REFERENCES

- [1] MITRE. CWE-191: Integer Underflow (Wrap or Wraparound). <https://cwe.mitre.org/data/definitions/191.html>.
 - [2] MITRE. CWE-197: Numeric Truncation Error. <https://cwe.mitre.org/data/definitions/197.html>.
 - [3] MITRE. CWE-400: Uncontrolled Resource Consumption. <https://cwe.mitre.org/data/definitions/400.html>.
 - [4] MITRE. CWE-440: Expected Behavior Violation. <https://cwe.mitre.org/data/definitions/440.html>.
 - [5] MITRE. CWE-684: Protection Mechanism Failure. <https://cwe.mitre.org/data/definitions/693.html>.
 - [6] MITRE. CWE CATEGORY: 7PK - Security Features. <https://cwe.mitre.org/data/definitions/254.html>.
 - [7] MITRE. CWE CATEGORY: Behavioral Problems. <https://cwe.mitre.org/data/definitions/438.html>.
 - [8] MITRE. CWE CATEGORY: Numeric Errors. <https://cwe.mitre.org/data/definitions/189.html>.
 - [9] MITRE. CWE CATEGORY: Resource Management Errors. <https://cwe.mitre.org/data/definitions/399.html>.
 - [10] OWASP. Risk Rating Methodology. https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology
-

8. About Exvul Security

Premier Security for the Web3 Ecosystem

ExVul is a premier Web3 security firm committed to forging a secure and trustworthy decentralized ecosystem. Our elite team consists of security veterans from world-leading technology and blockchain security firms, including Huawei, YBB Captical, Qihoo 360, Amber, ByteDance, MoveBit, and PeckShield. Team member Nolan is ranked as a top-40 whitehat on Immunefi and is the platform's sole All-Star in the APAC region.

Our expertise covers the full spectrum of Web3 security. We conduct **meticulous smart contract audits**, having fortified thousands of projects on chains like Evm, Solana, Aptos, Sui etc. Our **Blockchain Protocol Audits** secure the core infrastructure of L1/L2 by uncovering deep-seated vulnerabilities. We also offer **comprehensive wallet audits** to protect user assets and provide **proactive web3 pentest**, enabling partners to neutralize threats before they strike.

Trusted by industry leaders, ExVul is the security partner for **OKX, Bitget, Cobo, Infini, Stacks, Aptos, Sui, CoreDAO, Sei** etc.

Contact

 **Website**
www.exvul.com

 **Twitter**
[@EXVULSEC](https://twitter.com/EXVULSEC)

 **Email**
contact@exvul.com

 **Github**
github.com/EXVUL-Sec