VIDMA

Drop customer logo

SMART CONTRACT AUDIT

Project: **Cherut Coin, LLC — XRU Token**

Date: **June 2026**

TABLE OF CONTENTS

SUMMARY

Vidma is pleased to present this audit report outlining our assessment of code, smart contracts, and other important audit insights and suggestions for management, developers, and users.

The system is a deliberately minimal, fixed-supply token distribution stack for the ArenX platform: *CherutCoin* is an ERC20Burnable token that mints its entire 1,500,000,000 XRU supply to the deployer at construction with no mint path, no owner, and no admin functions, targeting BSC. Distribution is split between direct transfers at TGE (Strategic 100M, Exchange 75M, UMG 200M = 375M) and vested allocations handled by *ArenaVestingDeployer*, a stateless, ownerless factory that — given the token and a future TGE timestamp — deploys seven audited OpenZeppelin *VestingWallet* instances (one per pool: Founders, Team, Operations, Marketing, Reserve, Community, Staking) totaling 1.125B, with all beneficiaries, amounts, and schedules hardcoded as constants.

Locks are modeled as a delayed linear start = TGE + lock (no cliff), and a single permissionless *fund()* pulls the exact per-pool amounts from the caller via SafeERC20, after which it is economically impossible to call again. Each *VestingWallet* is owned by its pool's 2-of-3 multisig beneficiary, who claims via permissionless *release()*; the primitive has no clawback, allows beneficiary / ownership transfer, and vests any donated tokens on the same schedule.

During the audit process, the Vidma team identified 2 issues, both of low severity — one resolved and one marked not valid. A detailed summary and the current state are displayed in the table below.

Severity of the issue	Total found	Resolved	Not valid
Critical	0 issues	0 issues	0 issues
High	0 issues	0 issues	0 issues
Medium	0 issues	0 issues	0 issues
Low	2 issues	1 issue	1 issue
Informational	0 issues	0 issues	0 issues
Total	**2 issues**	**1 issue**	**1 issue**

After evaluating the findings in this report and the final state after fixes, the Vidma auditors can state that the contracts are fully operational and secure. Under the given circumstances, we set the following risk level:

High Confidence

To set the codebase quality mark, our auditors evaluate the initial commit submitted for the scope of the audit and the last commit with the fixes. This approach helps us adequately and sequentially evaluate the quality of the code. Code style, optimization of the contracts, the number of issues, and the risk level of the issues are all taken into consideration. The Vidma team has developed a transparent evaluation codebase quality system, presented below.

Please note that the points are deducted out of 100 for each and every issue on the list of findings (according to the current status of the issue). Issues marked as "not valid" are not subject to point deduction.

Severity of the issue	Resolved	Unresolved
Critical	7	10
High	5	7
Medium	2	5
Low	0.5	1
Informational	0.1	0.5

Evaluating the initial commit and the last commit with the fixes, the Vidma audit team set the codebase quality mark and final score shown on the next page.

Codebase quality: 99.60

Evaluating the **initial commit** and the **last commit with the fixes**, the Vidma audit team set the following **codebase quality** mark.

Score

Based on the **overall result of the audit** and the state of the final reviewed commit, the Vidma audit team grants the following **score**:

In addition to manual check and static analysis, the auditing team has conducted a number of integrated autotests to ensure the given codebase has an adequate performance and security level. The test results and coverage can be found in the accompanying section of this audit report.

Please be aware that this audit does not certify the definitive reliability and security level of the contract. This document describes all vulnerabilities, typos, performance issues, and security issues found by the Vidma audit team. If the code is still under development, we highly recommend running one more audit once the code is finalized.

SCOPE OF WORK

Cherut Coin, LLC is the issuer of the XRU token (CherutCoin) — a fixed-supply ERC20 distribution stack deployed on BNB Smart Chain, comprising the token itself and a stateless vesting deployer that allocates the supply across seven vesting pools.

Within the scope of this audit, two independent auditors thoroughly investigated the given codebase and analyzed the overall security and performance of the smart contracts. The outcome is disclosed in this document.

The scope of work for the given audit consists of the following contracts:

- `CherutCoin.sol`
- `ArenaVestingDeployer.sol`

The source code was taken from the following source:

`https://github.com/Arena-Arenx/arenax-token.smartcontracts`

Initial commit submitted for the audit:

`4475950c0ebe470e663f4ab10568f4355f9ea143`

Last commit reviewed by the auditing team:

`d0493ca501b9010ab2e38ddb185733a4cc10fe22`

WORKFLOW OF THE AUDITING PROCESS

The Vidma audit team uses sophisticated, well-developed techniques to ensure contracts are free of vulnerabilities and security risks. The overall workflow consists of the following phases:

Phase 1: The research phase

Research

After kick-off, our security team researches the contract's logic and the expected behavior of the audited contracts.

Documentation reading

Auditors deep-dive into the project's documentation to map the codebase's behavior patterns and potential testing scenarios.

The outcome

Auditing strategies and methods are set, and the team is ready to begin the first audit part.

Phase 2: Manual part of the audit

Manual check

The team reads through the code to find security issues, typos, or discrepancies with the contract's logic. The initial commit stated in the agreement is considered.

Static analysis check

Static analysis tools surface any vulnerabilities missed during the manual check.

The outcome

An interim report with the list of issues.

Phase 3: Testing part of the audit

Integration tests

Auditors run integration tests with the Hardhat framework; coverage and results appear in the accompanying section of this report.

The outcome

A second interim report with new issues found during the testing part.

STRUCTURE AND ORGANIZATION OF THE FINDINGS

For simplicity in reviewing the findings in this report, Vidma auditors classify the findings according to the severity level of the issues (from most critical to least critical).

All issues are marked as “Resolved” or “Unresolved”, depending on whether they have been fixed by Cherut Coin, LLC. Issues with a “Not Valid” status remain on the list of findings but are not eligible for score-point deduction.

The latest commit with the fixes reviewed by the auditors is indicated in the “Scope of Work” section of the report.

The Vidma team always provides a detailed description of the issues and recommendations on how to fix them.

Classification of found issues is graded according to the levels of severity described below:

Critical

The issue affects the contract in such a way that funds may be lost or allocated incorrectly, or could result in a significant loss.

Example: underflow / overflow, precision errors, locked funds.

High

The issue significantly affects the contract's ability to compile or operate. These are potential security or operational issues.

Example: compilation errors, pausing/unpausing of functionality, randomness, recursion, logic that can consume all gas in a block, missing limits for locking period or cooldown, arithmetic errors that can cause underflow.

Medium

The issue slightly impacts the contract's ability to operate by slightly hindering its intended behavior.

Example: absence of emergency withdrawal of funds, using assert for parameter sanitization.

Low

The issue doesn't contain operational or security risks, but is more related to optimization of the codebase.

Example: unused variables, inappropriate function visibility (public instead of external), useless imports, misuse of constant/immutable, absent event indexing, absent events for important state changes, missing getters, using a string as a key instead of a hash.

Informational

Every point that increases onboarding time and code reading, as well as issues that have no impact on the contract's ability to operate.

Example: code style, NatSpec, typos, license, refactoring, naming conventions, layout order, function order, lack of documentation.

Missing zero-address validation for token address

Low | ML – 01 | Resolved

The constructor of the *ArenaVestingDeployer* contract doesn't validate the *_token* address. If *_token* is accidentally passed as *address(0)*, all vesting wallets will still be deployed, but *fund()* will be permanently unusable.

Recommendation:

Add the necessary validation for the token's address in the constructor.

No events emitted on deployment or funding

Low | ML – 02 | Not valid

The *ArenaVestingDeployer* contract emits no events of its own. The constructor deploys seven *VestingWallet* instances but logs nothing tying each wallet address to its pool identity, amount, and schedule. *fund()* performs the seven distributing transfers but emits no funding marker. The gap is at the semantic layer — it makes the system less transparent and harder to monitor.

Recommendation:

Emit events when vesting wallets are created and when funding is completed.

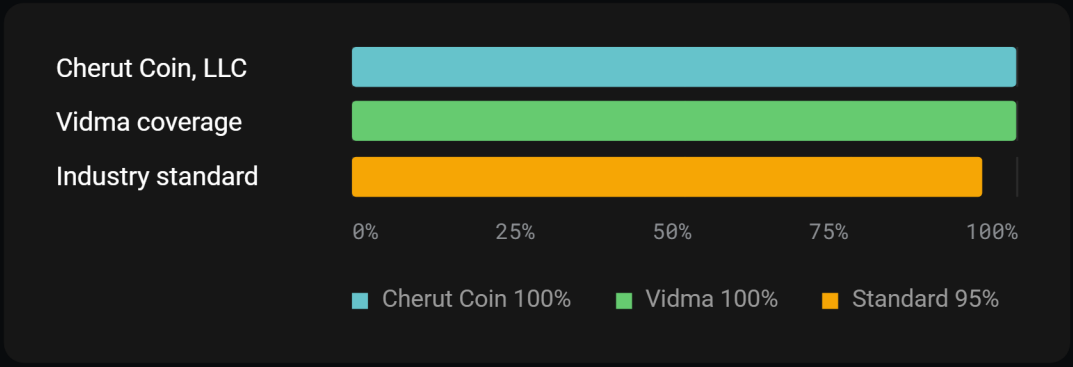
Comment from the development team

`Constructor (deployment):` the seven wallet addresses are stored in public immutable state variables (*FOUNDERS_WALLET*, *TEAM_WALLET*, etc.), so anyone can read the full deployment configuration on-chain. The transaction itself is permanently in the block history — no event adds meaningful information.

`fund():` *SafeERC20.safeTransferFrom* emits a standard ERC-20 *Transfer* event for each of the seven transfers — those are already indexed by every block explorer. A redundant *Funded()* event would carry no additional data.

TEST RESULTS

To verify the security and performance of the contracts, a number of integration tests were carried out using the Hardhat testing framework. In this section, we provide both the tests written by ArenX and the tests written by Vidma auditors.



It is important to note that Vidma auditors do not modify, edit, or add tests to the existing tests provided in the Cherut Coin, LLC repository. We write totally separate tests with a code coverage of a minimum of 95% to meet industry standards.

Tests written by Cherut Coin, LLC

The tests provided here were taken from the following commit:

`github.com/Arena-Arenx/arenax-token.smartcontracts · d0493ca5`

Test coverage

File	% Stmts	% Branch	% Funcs	% Lines
contracts\	100	100	100	100
ArenaVestingDeployer.sol	100	100	100	100
CherutCoin.sol	100	100	100	100
All files	**100**	**100**	**100**	**100**

Test results — Cherut Coin, LLC

ArenaVestingDeployer

Deployment
- ✓ deploys a VestingWallet for every pool
- ✓ each VestingWallet has the correct beneficiary
- ✓ each VestingWallet has the correct start and duration
- ✓ rejects zero token address
- ✓ rejects TGE in the past
- ✓ XRU_TOKEN is set to the token address

fund()
- ✓ reverts InsufficientAllowance when allowance is too low
- ✓ distributes exact allocation to each VestingWallet
- ✓ total across all wallets equals TOTAL_VESTED
- ✓ deployer holds no tokens after fund()
- ✓ reverts InsufficientAllowance on second call

VestingWallet — release (Marketing, no lock, 24mo vest)
- ✓ nothing releasable before TGE
- ✓ ~50% releasable at 12 months
- ✓ 100% releasable after vesting ends
- ✓ release() transfers tokens to beneficiary
- ✓ released() tracks cumulative amount

VestingWallet — release (Founders, 12mo lock + 24mo vest)
- ✓ nothing releasable during lock period
- ✓ ~50% releasable 12 months after lock end
- ✓ 100% releasable after lock + vesting

VestingWallet — Reserve (6mo lock + 9mo vest)
- ✓ nothing releasable during lock
- ✓ 100% releasable after lock + vesting

all pools — full release after vesting ends
- ✓ founders receives full allocation
- ✓ team receives full allocation
- ✓ operations receives full allocation
- ✓ marketing receives full allocation
- ✓ reserve receives full allocation
- ✓ community receives full allocation
- ✓ staking receives full allocation

CherutCoin

Metadata
- ✓ name is Cherut Coin
- ✓ symbol is XRU
- ✓ decimals is 18

Initial supply
- ✓ total supply is 1,500,000,000 XRU
- ✓ entire supply minted to deployer
- ✓ no mint function exists after deployment

Transfer
- ✓ reduces sender balance and increases receiver balance
- ✓ reverts when transferring to zero address

Burn
- ✓ reduces total supply
- ✓ reduces burner balance
- ✓ burnFrom reduces allowance and total supply
- ✓ reverts when burning more than balance

Direct transfers
- ✓ Strategic pool received 100M directly
- ✓ Exchange pool received 75M directly
- ✓ UMG received 200M directly

VestingWallets
- ✓ Founders — 375M, 12mo lock, 24mo vesting
- ✓ Team & Advisors — 180M, 12mo lock, 24mo vesting
- ✓ Operations & Development — 125M, 3mo lock, 18mo vesting
- ✓ Marketing & Growth — 90M, no lock, 24mo vesting
- ✓ Reserve — 85M, 6mo lock, 9mo vesting
- ✓ Community Incentives — 150M, no lock, 15mo vesting
- ✓ Staking Rewards — 120M, no lock, 12mo vesting
- ✓ deployer holds no tokens

51 passing (425ms)

Tests written by Vidma auditors

Test coverage

File	% Stmts	% Branch	% Funcs	% Lines
contracts\	100	100	100	100
ArenaVestingDeployer.sol	100	100	100	100
CherutCoin.sol	100	100	100	100
All files	**100**	**100**	**100**	**100**

Test results

```
ArenaVestingDeployer
    Setup Phase Test Cases
      ✓ should receive the token's address correctly
      ✓ should expose TOTAL_VESTED as 1,125,000,000 XRU
      ✓ should deploy a VestingWallet with a non-zero address for
      every pool
      ✓ should set the correct beneficiary (owner) for every
      VestingWallet
      ✓ should set the correct start, duration and end for every
      VestingWallet
      ✓ should not hold or move any tokens during deployment
      ✓ should revert if the provided token address is zero
      ✓ should revert if the provided TGE is in the past
      ✓ should revert if the provided TGE equals the current block
      timestamp
```

ArenaVestingDeployer (continued)

Funding Phase Test Cases

- ✓ should distribute the exact allocation to each VestingWallet
- ✓ should pull exactly TOTAL_VESTED from the funder and hold no tokens itself
- ✓ should account for the full 1,125,000,000 XRU across all VestingWallets
- ✓ should keep the excess allowance untouched when the funder over-approves
- ✓ should revert if the allowance is below TOTAL_VESTED
- ✓ should revert when fund() is called a second time
- ✓ should revert if the allowance is sufficient but the caller's balance is not

Vesting Release Phase Test Cases

- ✓ should have nothing releasable before the lock period expires
- ✓ should have exactly zero releasable at the start instant
- ✓ should release tokens linearly at 25%, 50% and 75% of the vesting period
- ✓ should cap the releasable amount at the full allocation after the end
- ✓ should transfer vested tokens to the beneficiary correctly
- ✓ should allow anyone to trigger a release with tokens going to the beneficiary
- ✓ should accumulate multiple partial releases up to the full allocation
- ✓ should vest tokens donated to the wallet on the same schedule
- ✓ should redirect releases to the new owner after an ownership transfer
- ✓ should also vest native ETH sent to the wallet to the beneficiary
- ✓ should release the full allocation to every pool after lock and vesting end

CherutCoin

Setup Phase Test Cases

- ✓ should receive the token's name correctly
- ✓ should receive the token's symbol correctly
- ✓ should receive the token's decimals correctly
- ✓ should receive the token's total supply correctly
- ✓ should receive the owner's balance correctly

Approval Phase Test Cases

- ✓ should set the allowance from the owner to the spender correctly
- ✓ should override previous allowance correctly
- ✓ should reset the allowance correctly when set to zero
- ✓ should revert when the spender is zero address

Transfer Phase Test Cases

- ✓ should transfer tokens between accounts correctly
- ✓ should revert when the recipient is zero address
- ✓ should revert when the transfer's amount exceeds the sender's balance
- ✓ should transfer tokens via transferFrom() within allowance
- ✓ should not reduce infinite allowance while transferring
- ✓ should revert transfer via transferFrom() without allowance
- ✓ should revert transfer via transferFrom() when amount exceeds allowance
- ✓ should revert transfer via transferFrom() when recipient is zero address
- ✓ should revert transfer via transferFrom() when sender is zero address

Burnable Phase Test Cases

- ✓ should burn tokens correctly
- ✓ should revert when burning more than balance
- ✓ should burn tokens from another account correctly via burnFrom()
- ✓ should revert burn tokens via burnFrom() without allowance
- ✓ should revert when the amount exceeds the allowance while using burnFrom()
- ✓ should revert when the amount exceeds target balance while using burnFrom()
- ✓ should revert burn tokens via burnFrom() when target is zero address

Deployment E2E Test Cases

Step 1 — Contract Deployment

✓ should deploy CherutCoin with correct metadata and mint entire supply to deployer

✓ should deploy ArenaVestingDeployer with correct initial state

✓ should deploy a VestingWallet per pool with correct beneficiary, schedule and zero balance

Step 2 — Direct Pool Transfers (no vesting)

✓ should transfer 100,000,000 XRU directly to the Strategic pool

✓ should transfer 75,000,000 XRU directly to the Exchange Liquidity pool

✓ should transfer 200,000,000 XRU directly to the Universal Music Group pool

✓ should leave the deployer holding exactly TOTAL_VESTED after the direct transfers

Step 3 — Vesting Wallet Funding

✓ should approve the ArenaVestingDeployer for 1,125,000,000 XRU

✓ should distribute the exact allocation to every VestingWallet via fund()

✓ should leave no tokens in the deployer or the owner after funding

Step 4 — Token Accounting

✓ should account for all 1,500,000,000 XRU across direct pools and vesting wallets

Step 5 — Vesting Release Lifecycle

✓ should have nothing releasable for any pool before TGE

✓ should release the full allocation to each pool beneficiary after lock and vesting end

✓ should have fully drained every VestingWallet after release

65 passing (879ms)

We are delighted to have had the chance to work with the Cherut Coin, LLC team and to contribute to your company's success by reviewing and certifying the security of your smart contracts.

The statements made in this document should be interpreted neither as investment nor as legal advice, nor should its authors be held accountable for decisions made based on this document.

Vidma is a security audit company helping crypto companies ensure their code and products operate safely and as intended, enabling founders to sleep soundly at night. We specialize in auditing DeFi protocols, layer-one protocols, and marketplace solutions. Our team consists of experienced and internationally trained specialists.

Website: **vidma.io**

VIDMA