Ore Smart Contract Security Audit Report

Date: 2026-11-01 Chain: BNB Smart Chain (BSC)
Version: v2 (remediation status added, see Section 10)
Baseline: branch BSC @ acd533c + uncommitted working tree
Compiler: solidity ^0.8.20 (floating pragma)
Method: manual line-by-line review + business-logic and economic-model analysis; the re-review also ran the Foundry test suite (forked BSC testnet)

1. Scope

File	Lines	SHA-256 (first 16)
MiningToken.sol	413	1de6f2aacb5b942d
MiningPool.sol	258	08fbf65686e6c0ff
OreIEO.sol	588	6e51c20fac848c68
OreBidBook.sol	307	0c495757c882669c
QuizContract.sol	488	a00e6dedaf62492b
libraries/OreGenesisCurve.sol	159	e4004587651f0b93
libraries/AgentIdLib.sol	110	9d5b9777b1c00d42
Total	2323 lines	

Out of scope: the backend quiz-judging / reward service, the frontend, deployment scripts, and external contracts (PancakeSwap, the Binance Agent ID registry).

2. Summary

Severity	Count	Fixed	Notes
Critical	0	-	
High	1	1	owner minting power
Medium	4	1	plus 1 partially fixed
Low	6	0	
Info	7	0	plus 1 partially fixed

No vulnerability was found that lets an external attacker directly steal user funds. The main initial risk was over-centralised control: a single owner key could mint the remaining ~19.95M ORE after launch, and reward amounts had no on-chain bound. Both (H-01, M-04) are fixed and tested.

Initial rating: Medium risk
Re-review: Low-to-medium risk (remaining items are mostly admin parameters and MEV)

3. Architecture

MiningToken	ORE token, hard cap 21,000,000; pre-launch transfer lock; DEX buy/sell tax (1%, max 5%)
OreIEO	linear curve sale P→5P, hard cap 80 BNB, ≤ 0.03 BNB per Agent ID; graduates into a pool whose LP goes to 0xdEaD
OreBidBook	pre-launch bid-only order book (2% fee on each side)
MiningPool	staking gate (≥150 ORE, 24h lock); no rewards
QuizContract	quiz mining: submit answers / record rewards / mint on claim

Supply: IEO 5% = 1,050,000 (772,000 sold + LP + rest burned)
Mining 95% = 19,950,000 (minted by QuizContract)

4. High

[H-01] Owner could mint up to the cap after launch ✅ Fixed
Where: MiningToken.sol:118-124, 193-197, 331-337 (pre-fix line numbers)
Issue: onlyMinter let the owner itself, and any address added via setMinter, call ownerMint once tradingOpen, bounded only by maxSupply. No on-chain release schedule tied to MiningPool.periodRewards existed.
Impact: a leaked or malicious owner key could mint the remaining ~19.95M ORE in one go and dump it.
Recommendation:
- remove the owner from onlyMinter, keep QuizContract;
- enforce per-period emission limits on-chain;
- renounce or move ownership to a multisig + timelock.
Fix: owner removed from onlyMinter; new lockMinters() freezes the minter list forever; QuizContract now caps rewards per period (see M-04).

5. Medium

[M-01] Unbounded reward arrays: gas DoS on claim
Where: QuizContract.sol:311-345, 349-433
Issue: every win pushes a RewardEntry (2 storage slots); claims iterate, compact or delete the whole array. At one round per 600s a user can collect up to 4,320 entries a month; left unclaimed long enough, a claim exceeds the block gas limit and those rewards become unclaimable.
Recommendation: aggregate per user (unlocked total plus a few daily lock buckets) or add claim(maxEntries).

[M-02] Fee swaps use amountOutMin = 0 (sandwichable)
Where: MiningToken.sol:299-322
       QuizContract.sol:181-221
Issue: both swaps accept any output and swapTax() /
  swapFees() are permissionless. QuizContract.swapFees
  swaps the whole balance, not one threshold, so an
  attacker can trigger and sandwich it when it is large.
Recommendation: make swapFees onlyOwner or cap it at the
  threshold; derive minOut from reserves (e.g. 95%); or
  submit privately.

[M-03] Owner can change user-facing params without bounds
                                    ◐ Partially fixed
Where: QuizContract.sol:157-160, 223-230
       MiningToken.sol:285-296, 218-222
Issue:
  - setSubmitFee has no cap: mining scripts with unlimited
    approval to Quiz could be charged any amount;
  - setMiningPool(0) disabled the stake gate;
    setMiningToken could switch the minted token;
  - setAmmPair can mark any address (e.g. an exchange
    deposit address) as a pair, taxing it up to 5%;
  - setPreLaunchSpender can whitelist any contract to move
    ORE before launch (still only to wallets, no pool).
Recommendation: cap submitFee; make miningToken immutable;
  restrict setAmmPair to Pancake factory pairs; timelock
  and emit events for key parameter changes.
Fix: miningToken is immutable and setMiningToken removed;
  setMiningPool can be set only once and never to zero.
  The submitFee cap and setAmmPair restriction are open.

[M-04] Reward amounts set off-chain with no on-chain bound
                                    ✅ Fixed
Where: QuizContract.sol:311-345, MiningPool.sol:236-257
Issue: addClaimable* accepted any amount, unchecked against
  periodRewards or remainingMintable; periodRewards was
  advisory only. A backend bug or leaked key could
  over-issue, and claims past the cap revert entirely.
Recommendation: track allocations per
  MiningPool.currentPeriod() and reject anything above
  the period's periodRewards.
Fix: new emissionCap() = sum of periodRewards for every
  period started so far (unused budget carries over);
  winner and referral rewards both count towards
  totalAllocated and revert above the cap; the cap is 0
  before mining starts.

6. Low

[L-01] Pre-launch skim guard can be bypassed via sync
Where: OreIEO.sol:336-346
Issue: before launch anyone can send WBNB to the pair and
  call sync(), leaving reserves at (0, X); launch then sees
  non-zero reserves, skips skim, and the donated WBNB joins
  the pool, opening above the intended 20/3·P. The donor
  loses X and can hardly profit, but the code comment is
  inaccurate.
Recommendation: read the existing WBNB reserve X and size
  the ORE side from (lpBnb + X) so the opening price is
  always 20/3·P.

[L-02] Stake lock applies only to the first stake
Where: MiningPool.sol:143-146
Issue: top-ups do not reset stakeTime, so after 24h a
  top-up can be withdrawn at once. Limited effect on the
  eligibility gate.
Recommendation: reset stakeTime by weighted average if a
  strict lock is wanted.

[L-03] Claims fail entirely once the cap is reached
Where: QuizContract.sol:364, 384-388
Issue: ownerMint above maxSupply reverts the whole claim,
  so nothing at all can be claimed; the 50% early-claim
  penalty is newly minted to marketingWallet and also uses
  up mining supply.
Recommendation: clamp to remainingMintable and simply not
  mint the penalty (equivalent to burning it).

[L-04] answerLetter is not validated
Where: QuizContract.sol:270-276
Issue: strings of any length are accepted, bloating event
  logs and backend parsing.
Recommendation: use a uint8 option or restrict to letters.

[L-05] tx.origin gating excludes smart-contract wallets
Where: OreIEO.sol:253, OreBidBook.sol:122/172,
       MiningToken.sol:233/376-387
Issue: multisig / AA wallets cannot join the IEO, place
  bids or transfer before launch. EIP-7702 accounts can
  batch calls despite tx.origin, but remain bound by the
  Agent ID and 0.03 BNB limits.
Recommendation: state this clearly in docs and the UI.

[L-06] Answers are public on-chain and could be copied
(rated High in the initial review)
Where: QuizContract.sol:270-307
Issue: submitAnswer logs the answer letter in plain text,
readable from the mempool or mined events during a round.
Re-review: downgraded to Low because
- only Agent / Worker wallets registered with the backend
can win;
- the backend's GET /api/quiz/question already gives each
staked Agent an AI answer (answer_letter), so copying
on-chain answers adds at most a "majority vote" edge.
Recommendation: if the server stops supplying answers,
switch to per-round ephemeral-key encrypted submissions
(still one tx) with the private key published after.

7. Informational

[I-01] transferOwnership in MiningPool / QuizContract emits
no event; several Quiz setters emit none either.
◐ QuizContract now emits OwnershipTransferred /
MarketingWalletSet / MiningPoolSet; MiningPool unchanged.
[I-02] No two-step ownership transfer; a typo is permanent.
[I-03] Floating pragma ^0.8.20; pin an exact version.
[I-04] MiningPool ignores transfer/transferFrom return
values (safe today: ORE returns true or reverts).
[I-05] Dead code: MiningPool.miningEnded, marketingWallet;
Quiz.finalizeRound does not block later addClaimable.
[I-06] Adding / removing liquidity is taxed as a sell / buy;
traders must use the SupportingFeeOnTransferTokens calls.
[I-07] OreBidBook lets wallets without an Agent ID buy ORE
before launch (by design; document it); if feeRecipient
rejects BNB, fees stay in the contract.

8. Security strengths

✅ Tight pre-launch lock: wallet → registered wallet only,
signed by the sender (from == tx.origin); the pair has
code and cannot receive, so nobody (owner included) can
create a pool before the IEO.
✅ Only the launcher can mint before launch; other minters
need tradingOpen, so the 5% IEO share stays reserved.
✅ The IEO mints exactly GENESIS_SUPPLY; the rest goes to
0xdEaD and the LP is locked forever.
✅ 48h after the sale anyone can launch, regardless of
pause or registry state, so the token can never be
locked forever; an empty sale can still open trading.
✅ AgentIdLib calls the external registry with fixed gas,
bounded return size and manual ABI checks, defeating
return bombs and malformed data.
✅ The price curve uses a 512-bit mulDiv and always rounds
in the protocol's favour (cost ≤ budget is proven);
caps are booked per Agent ID and per wallet.
✅ All IEO / BidBook money paths are nonReentrant + CEI;
failed BNB refunds revert; the project fund is pulled.
✅ BidBook escrow is booked cumulatively and reaches exactly
zero when a bid fills: escrowTotal + feesAccrued ==
balance holds.
✅ Tax is hard-capped at 5%, and a failed tax swap never
blocks a user's sell.

9. Remediation priority

Priority	ID	Status / planned change
P0	H-01	✅ Fixed
P1	M-04	✅ Fixed
P1	M-01	aggregated storage / batched claim
P1	M-02 M-03	minOut / param caps (M-03 partial)
P2	L-01 ~ L-06	small changes (L-06 kept as is)
P3	I-01 ~ I-07	code hygiene (I-01 partial)

Before mainnet: move every owner to a multisig (e.g. Safe
2/3) + 24h timelock; add invariant tests (supply ≤ cap,
the BidBook balance identity).

10. Remediation tracking (v2 re-review)

Re-reviewed file	Lines	SHA-256 (first 16)
MiningToken.sol	424	68462ed53bfc79a6
QuizContract.sol	531	6206d8f92d67d2ed

MiningToken
- onlyMinter: launcher, or (isMinter && tradingOpen);
the owner can never mint
- lockMinters(): freezes the minter list (irreversible)
QuizContract
- emissionCap() / totalAllocated: on-chain reward cap
- miningToken immutable; setMiningToken removed
- setMiningPool settable once, never to zero
Operations
- make lock-minters; deploy-all locks the minter list
right after set-mining-pool (SKIP_LOCK_MINTERS=1 skips)
- set-mining-pool no longer accepts the zero address

Tests (Foundry, forked BSC testnet, solc 0.8.34)
70 / 70 passing, including 4 new ones:
- test_owner_cannotMint_evenAfterLaunch
- test_lockMinters_freezesMinterList
- test_quiz_setMiningPool_onlyOnce
- test_quiz_emissionCap_boundsAllRewards

Note: these fixes take effect only after MiningToken and
QuizContract are redeployed; once the minter list is locked,
replacing QuizContract requires redeploying the token too.

11. Disclaimer

This report reflects a manual review of the source versions listed above. It is not investment advice and does not guarantee the absence of other vulnerabilities. Backend services, key management and external contracts (PancakeSwap, the Agent ID registry) are out of scope. Any code change requires a fresh assessment.