SMC AUDITS



SMART CONTRACT SECURITY AUDIT

WKC
WikiCat Coin

PROJECT	WKC / WikiCat Coin ("WIKI CAT")
CHAIN	BNB Chain (BSC, chain id 56)
CONTRACT	`0x6Ec90334d89dBdc89E08A133271be3d104128Edb`
VERSION	Deployed bytecode; Sourcify runtime match (matchId 9867901)
REPORT ID	SMC-WKC-202609-01
DATE	1 September 2026
PREPARED BY	SMC Audits
STATUS	Draft

00 Confidentiality notice

This report is prepared by SMC Audits for the named project and contains confidential information, including identified security findings. It should not be shared publicly until the project has remediated the issues described, and only with SMC Audits' agreement. Publication as part of public verification follows the process agreed in the engagement.

01 Contents

Project overview Section 2
Executive summary Section 3
Scope Section 4
Methodology Section 5
Severity classification Section 6
Findings summary Section 7
Detailed findings Section 8
Remediation status Section 9
Conclusion Section 10
Disclaimer Section 11

02 Project overview

WKC (WikiCat Coin, on-chain name "WIKI CAT") is a deflationary BEP-20 token on BNB Chain. This audit covers the single deployed contract (`CoinToken`) that implements the token: balances, transfers, a per-transfer fee and burn, and a set of administrative controls. The contract is a widely reused token-generator template compiled with an old Solidity toolchain. It is not a proxy and is not upgradeable.

SYSTEM AT A GLANCE

Component	Role
`CoinToken` token contract	Core BEP-20 logic: balances, transfer / transferFrom, allowances
Fee & burn logic (in-contract)	Takes txFee% to a fee address and burns burnFee% on each transfer; both currently 1%
Admin / ownership	owner holds mint, updateFee, pause / unpause, blackListAddress. Ownership is renounced to 0x0000…0001
External integrations	None in-contract. No oracle, no external calls, no ETH handling. Liquidity / DEX pairs out of scope

KEY MECHANISMS TO NOTE

- Fee-on-transfer plus burn. Every transfer from a non-fee address deducts a 1% fee (sent to FeeAddress `0xFA78…2577`) and burns 1% (reducing totalSupply). A recipient therefore receives about 98% of the sent amount.
- Ownership is renounced to a dead address. owner() is `0x0000000000000000000000000000000000000001`, the ecrecover precompile address, for which no private key exists and which cannot originate a transaction. Because transferOwnership forbids address(0), the project renounced by transferring to 0x…0001. Consequence: every onlyOwner function (mint, updateFee, pause, unpause, blackListAddress, transferOwnership) is permanently uncallable. This single fact is the foundation of the token's current security posture.
- No "UI multiplier" mechanism exists on-chain. Project materials describe a balance-scaling "UI multiplier" (a "BEP-8056" idea, a UIMultiplierUpdated event). No such mechanism is present in the verified deployed code or ABI: there is no multiplier, no UIMultiplierUpdated event, and no reflection or rebase logic. balanceOf returns the raw stored balance. This audit is scoped to the code that is actually deployed. "BEP-8056" is not a recognized BNB Chain standard.

03 Executive summary

SMC Audits reviewed the WKC / WikiCat Coin smart contract on BNB Chain at address `0x6Ec90334d89dBdc89E08A133271be3d104128Edb`, using the source verified on Sourcify (a runtime-bytecode match). The review covered the deployed `CoinToken` contract in full, its transfer, fee, burn, allowance, and administrative logic, and cross-checked its live on-chain state.

The review identified 7 findings: 0 critical, 0 high, 0 medium, 2 low, and 5 informational. No issue in the live contract results in loss of user or protocol funds. The most significant observation is structural rather than a code defect: the contract contains administrative powers that would be severe if active, an unbounded mint (arbitrary supply inflation), an updateFee that can set fees to values which brick all transfers, and pause and blacklist switches. All of these are neutralized because ownership is renounced to 0x0000…0001, making every privileged function permanently uncallable. The only live,

code-level findings are a standard ERC-20 approve race condition (Low) and an inconsistency in the blacklist logic together with a permanent-freeze latency (Low).

Overall, the code is a standard, immutable fee-on-transfer and burn token with no live critical exposure. Its integrity is contingent on ownership remaining at the dead address, a condition that is currently true and, given the address used, cannot be reversed.

As of this report, 0 of 7 findings have been formally resolved. The contract is immutable, so the recommendations are advisory: they apply to token holders, integrators, and any future redeployment rather than to the live bytecode.

04 Scope

Item	Detail
Project	WKC / WikiCat Coin ("WIKI CAT")
Chain	BNB Chain (BSC, chain id 56)
Language	Solidity
Contract	`0x6Ec90334d89dBdc89E08A133271be3d104128Edb` (verified source via Sourcify)
Version	Deployed bytecode; Sourcify runtime match (matchId 9867901, verified 2025-11-09). Not a proxy
Compiler	solc 0.4.24+commit.e67f0147; optimizer enabled, runs 200; EVM version byzantium

FILES REVIEWED

File	Lines	Notes
`CoinToken.sol`	308	Entire contract: SafeMath, Ownable, Pausable, StandardToken, PausableToken, CoinToken

OUT OF SCOPE

- Off-chain components, the project website and front end, and any "UI multiplier" behaviour implemented off-chain (it does not exist in the deployed contract).
- Liquidity pools, DEX pairs, routers, and market or economic conditions.
- Deployment process and key management history.
- Full on-chain event history (creation transaction, constructor arguments, Blacklist and OwnershipTransferred logs). These require a paid BSC data-API plan or an archive node and were not available during this review; see the open item in Section 9. Their absence does not affect any Critical, High, or Medium conclusion.

05 Methodology

Every SMC Audits review follows the same process, so the result means the same thing on every engagement.

1. Threat modeling. Understand what the contract does, where value moves, and who holds privilege.
2. Manual review. A line-by-line reading of the code by the audit team. This is the core of the work and where most issues are found.
3. Tooling. Static analysis and security tooling support the manual review, they do not replace it.
4. Proof of concept. Every material finding is confirmed with a working proof of concept rather than asserted.
5. Reporting. Findings are written up with severity, reproduction, and a clear recommendation.
6. Re-audit. Fixes are verified by the team to confirm the issue is closed without introducing a new one.

REVIEW CHECKLIST

Every review covers, at minimum: access control and owner checks; privilege and centralization risk; arithmetic, overflow, and precision; reentrancy and external call safety; token and supply logic (mint and burn); fee, tax, and reflection mechanics; approvals and allowance handling; oracle and price handling; upgradeability and proxy risk; pause, blacklist, and emergency switches; front-running and MEV exposure; and denial of service and gas griefing.

TOOLING USED

Manual review of the Sourcify-verified source; cast (Foundry) for live on-chain state and eth_call reproductions against BNB Chain; Sourcify for source and metadata verification.

06 Severity classification

Each finding is rated on impact and likelihood using one consistent scale.

Severity	Meaning
CRITICAL	Direct loss of funds or full compromise, realistic to exploit
HIGH	Serious impact, exploitable under conditions a project would actually meet
MEDIUM	Meaningful risk, limited by conditions or the size of impact
LOW	Minor risk, hard to exploit or low in impact
INFORMATIONAL	Best practice and code quality notes worth addressing over time

Findings whose underlying capability would be severe if a trust assumption did not hold are rated on their current state, with the conditional severity stated explicitly in the finding.

07 Findings summary

ID	Title	Severity	Status
SMC-L-01	ERC-20 approve race condition	LOW	Open (advisory)
SMC-L-02	Blacklist bypass via transferFrom and permanent-freeze latency	LOW	Open (pending)
SMC-I-01	Privileged functions rely entirely on renounced ownership	INFO	Acknowledged (mitigated)
SMC-I-02	Fragile and imprecise fee formula (100 / fee)	INFO	Open (advisory)
SMC-I-03	Fee-on-transfer breaks naive integrations	INFO	Open (advisory)
SMC-I-04	Legacy compiler (0.4.24) and floating pragma	INFO	Open (advisory)
SMC-I-05	decimals declared as uint256 instead of uint8	INFO	Open (advisory)

Critical	High	Medium	Low	Informational	Total
0	0	0	2	5	7

08 Detailed findings

SMC-L-01 LOW OPEN

ERC-20 approve race condition (SWC-114)

CATEGORY Approvals and allowance handling

LOCATION `approve(address,uint256)`, L189-193 (mitigations increaseApproval / decreaseApproval, L201-216)

DESCRIPTION

approve overwrites the spender's allowance unconditionally, without requiring the current allowance to be zero first. This is the classic ERC-20 approval race: if an owner changes an existing non-zero allowance, a spender who is watching the mempool can spend the old allowance and then, after the new approve lands, spend the new allowance as well.

IMPACT

A spender who already holds a non-zero allowance can, around an allowance change, transfer up to the sum of the old and new allowances instead of just the new value. Impact is bounded by the allowances the owner grants, and exploitation requires a specific victim action plus transaction-ordering advantage, hence Low. The contract already ships the standard mitigation (increaseApproval / decreaseApproval).

PROOF OF CONCEPT

Owner approve(spender, 100), then later approve(spender, 50) to reduce. Spender front-runs the second call with transferFrom(owner, spender, 100) using the old allowance, then after it mines, transferFrom(owner, spender, 50). Result: the spender moved 150 despite the owner intending a 50 cap.

RECOMMENDATION

The live contract is immutable, so no code fix is possible. Advise users and integrators to change allowances via decreaseApproval / increaseApproval, or to set the allowance to 0 before setting a new non-zero value. Any future token version should follow the same pattern.

SMC-L-02 LOW OPEN (PENDING)

Blacklist bypass via transferFrom, and permanent-freeze latency

CATEGORY Access control / emergency switches

LOCATION `transfer` L129, `transferFrom` L162, `_blackList` L220-225, `blackListAddress` L253-255

DESCRIPTION

The blacklist gate is applied inconsistently and, given renounced ownership, is also irreversible. First, a bypass: transfer blocks the call when tokenBlacklist[msg.sender] is true, correctly blocking the blacklisted holder. But transferFrom also checks tokenBlacklist[msg.sender], where msg.sender is the spender (caller), not _from (the token holder). A blacklisted holder who has an outstanding allowance to any non-blacklisted spender can therefore still have their tokens moved via transferFrom, defeating the freeze. Second, a permanent freeze: blackListAddress is onlyOwner, and because ownership is renounced (see SMC-I-01), any address blacklisted before renouncement can never be un-blacklisted; its direct transfer calls are blocked forever.

IMPACT

No protocol-wide fund risk. The realized impact depends entirely on whether any address is currently blacklisted: if none, this is Informational; if some are, those holders are permanently and only partially frozen (direct transfers blocked, transferFrom path open). Rated Low pending confirmation.

PROOF OF CONCEPT

Bypass: with holder V blacklisted and V having previously approved spender S (not blacklisted), S calls transferFrom(V, X, amount) and it succeeds despite V being blacklisted, because line 162 checks S, not V. Permanent freeze: blackListAddress(...) reverts for every caller because owner is 0x...0001, so no blacklist entry can be added or removed post-renouncement.

Confirmation still required (see Section 9): enumerating current blacklist entries needs the Blacklist(address,bool) event history. The tokenBlacklist mapping has no public getter, and full-range eth_getLogs was unavailable on the free BSC data tier during this review. If no addresses were ever blacklisted, downgrade this finding to Informational.

RECOMMENDATION

Immutable, no code fix. Confirm the on-chain blacklist state and disclose any permanently-frozen addresses to holders. Any future version should gate transferFrom on tokenBlacklist[_from] (and, if intended, _to), not on the spender.

SMC-I-01 INFORMATIONAL ACKNOWLEDGED (MITIGATED)

Privileged functions rely entirely on renounced ownership

CATEGORY Privilege and centralization risk

LOCATION `mint` L299-305, `updateFee` L284-288, `pause/unpause` L85-96, `blackListAddress` L253-255, `transferOwnership` L51-55

DESCRIPTION

The contract grants the owner a set of powerful capabilities. In the deployed state these are all inert because owner() is 0x0000000000000000000000000000000000000001, an address with no private key that cannot send a transaction. Every onlyOwner function is therefore permanently uncallable. The capabilities, and what their severity would be if ownership were active, are below.

Function	Capability	Severity if owner were active
`mint`	Mint unlimited new WKC to any account; no cap	CRITICAL arbitrary supply inflation
`updateFee`	Set fees to any value, including values that brick all transfers or redirect fees	HIGH-CRIT token freeze or fee seizure
`pause / unpause`	Halt all transfers indefinitely	HIGH funds trapped while paused
`blackListAddress`	Freeze or unfreeze specific addresses	HIGH targeted censorship

IMPACT

Current impact is none: the powers cannot be exercised. This finding is Informational on the condition that ownership remains renounced. Given the renounce target (0x…0001, unrecoverable), the condition cannot be reversed, which is why the token can be treated as immutable. This is the single most important assumption in the report and should be independently re-verified by any reader.

PROOF OF CONCEPT

```
$ cast call 0x6Ec9…8Edb "owner()(address)"
0x0000000000000000000000000000000000000001

$ cast call 0x6Ec9…8Edb "mint(address,uint256)" 0x…dead 1 --from 0x…dead
execution reverted            # onlyOwner

$ cast call 0x6Ec9…8Edb "updateFee(uint256,uint256,address)" 5 5 0x…dead --from 0x…dead
execution reverted            # onlyOwner

$ cast call 0x6Ec9…8Edb "pause()" --from 0x…dead
execution reverted            # onlyOwner
```

RECOMMENDATION

No action possible or required on the live contract; document the design clearly for holders so the renounce is understood as the load-bearing guarantee. Any future token should replace unbounded mint with a capped, role-gated mint, bound updateFee, and prefer a timelock or multisig over a single owner key rather than relying on renouncement as the safety mechanism.

SMC-I-02 INFORMATIONAL OPEN

Fragile and imprecise fee formula (value / (100 / fee))

CATEGORY Arithmetic and precision / fee mechanics

LOCATION fee math in `transfer` L134-146 and `transferFrom` L168-180; `updateFee` L284-288; constructor L267-278

DESCRIPTION

Fees are computed as tempValue.div(uint256(100 / fee)). The inner 100 / fee is integer division evaluated first, which makes the fee both imprecise and fragile. At the deployed values (txFee = 1, burnFee = 1), 100 / 1 = 100, so the fee is value / 100, a clean 1%. Correct. But for any fee that is not a divisor of 100 the result is skewed: fee = 3 gives 100 / 3 = 33, so the "3%" fee is actually value / 33, about 3.03%. For fee greater than 100, 100 / fee = 0, and div(0) reverts, so every transfer would revert, bricking the token. A fee between 51 and 100 makes the fee at least 100% and causes the subsequent subtraction to underflow-revert. Neither the constructor nor updateFee validates these bounds. This is latent rather than live: the only setter is permanently uncallable (SMC-I-01) and the constructor ran once with safe values.

IMPACT

No live impact, parameters are frozen at safe values. Documented as a design defect that would matter for any redeployment or if ownership were not renounced.

RECOMMENDATION

Advisory for future versions: compute fees as value * feeBps / 10000 (basis points) with explicit bounds, and validate fee parameters in both the constructor and any setter.

SMC-I-03 INFORMATIONAL OPEN

Fee-on-transfer breaks naive integrations

CATEGORY Fee / tax mechanics; integration safety

LOCATION `transfer` L128-154, `transferFrom` L161-186

DESCRIPTION

Because each transfer deducts 1% fee plus 1% burn, the amount credited to the recipient is about 98% of the amount sent. Contracts and services that assume a transferred amount arrives in full, naive DEX routers, CEX deposit accounting, vaults, or bridges, will mis-account, credit the wrong amount, or revert on internal balance checks.

IMPACT

No defect in the token itself; this is an interoperability characteristic that integrators must handle. Mishandling by an integrator can strand or misattribute funds on their side. Informational.

PROOF OF CONCEPT

Send 1,000,000 base units via transfer: the recipient balance increases by 980,000, FeeAddress receives 10,000, and totalSupply decreases by 10,000 (the burn), each accompanied by its own Transfer event.

RECOMMENDATION

Integrators should measure received amounts as balanceAfter minus balanceBefore rather than trusting the sent value, and enable fee-on-transfer swap paths where applicable. No change is possible on the live token.

SMC-I-04 INFORMATIONAL OPEN

Legacy compiler (0.4.24) and floating pragma

CATEGORY Best practice / code quality

LOCATION `L1` (pragma solidity ^0.4.24), SafeMath L3-30

DESCRIPTION

The contract is compiled with solc 0.4.24 (2018-era, Byzantium) and declares a floating pragma ^0.4.24. Old compilers lack modern safety defaults (no built-in overflow checks, safety here depends entirely on the bundled SafeMath), and a floating pragma allows compilation under different patch versions than the one audited. SafeMath uses assert, which on this compiler consumes all remaining gas on failure.

IMPACT

No direct vulnerability was identified from the compiler version, arithmetic is consistently routed through SafeMath. Recorded for provenance and best practice. No fix is possible for the deployed bytecode.

RECOMMENDATION

Any future version should target a current, actively supported compiler (0.8.x or later, which has built-in checked arithmetic) with a fixed pragma.

SMC-I-05 INFORMATIONAL OPEN

decimals declared as uint256 instead of uint8

CATEGORY Standards compliance

LOCATION `L262` (uint public decimals;)

DESCRIPTION

The ERC-20 / BEP-20 convention types decimals as uint8. Here it is declared uint (uint256). The value returned (18) is correct, but strict ABI consumers that decode decimals() as uint8 may misread the return data, and some tooling and interface checks expect the conventional type.

IMPACT

Cosmetic and compatibility only; no security impact. Informational.

RECOMMENDATION

Use uint8 public decimals; in any future version for standards compliance. No change possible on the live contract.

09 Remediation status

The reviewed contract is immutable (not a proxy) and ownership is renounced, so on-chain code changes are not possible. The recommendations are therefore advisory, directed at holders, integrators, and any future redeployment, and no finding can be fixed in place. This table will be updated if the project responds or provides a new in-scope version.

ID	Severity	Status	Note
SMC-L-01	LOW	Open (advisory)	Immutable; mitigations already in-contract (increase / decreaseApproval)
SMC-L-02	LOW	Open (pending)	Awaiting confirmation of on-chain blacklist entries (needs event history)
SMC-I-01	INFO	Acknowledged (mitigated)	Powers neutralized by renounce to 0x...0001; verified live
SMC-I-02	INFO	Open (advisory)	Immutable; latent (setter uncallable)
SMC-I-03	INFO	Open (advisory)	Inherent to design; integrator guidance provided
SMC-I-04	INFO	Open (advisory)	Immutable; applies to future versions
SMC-I-05	INFO	Open (advisory)	Immutable; applies to future versions

Open item for completeness: confirm Blacklist and OwnershipTransferred event history and the creation-transaction constructor arguments. These require a paid BSC data-API plan or an archive node and were not available during this review; they affect only the precise severity of SMC-L-02 and the scope table's constructor detail, not any Critical, High, or Medium conclusion.

10 Conclusion

At the reviewed address, WKC / WikiCat Coin is a standard, immutable fee-on-transfer and burn BEP-20 token. The review found no critical, high, or medium severity issues in the live contract. The powerful administrative capabilities present in the code, unbounded mint, arbitrary fee changes, pause, and blacklist, are all neutralized because ownership has been renounced to an unrecoverable address, which our live checks confirm. The remaining findings are two Low-severity items (a standard approval race and a blacklist-logic inconsistency) and five informational notes, none of which put user or protocol funds at risk in the deployed state.

The token's safety rests on one condition: ownership remaining at the dead address 0x0000…0001. Because of the address chosen, that condition is effectively permanent. Readers should nonetheless independently verify owner() and note that the widely-referenced "UI multiplier / BEP-8056" mechanism is not part of the deployed contract.

> An audit reduces risk. It does not guarantee the absence of every vulnerability. Users and the project should treat this report as one part of a wider security practice.

11 Disclaimer

This report reflects the code reviewed at the specific commit or address stated in the scope, at the time of review. It does not cover changes made after the review, external dependencies outside the stated scope, off-chain components, or economic and market conditions. This report is not financial, investment, or legal advice. SMC Audits provides no warranty, express or implied, and accepts no liability for any loss arising from the use of the audited code. A clean report is not a guarantee of safety.



SMC AUDITS

Security providers specializing in smart contract audit, penetration testing, and application security assessment.

smcaudits.com · SMC DAO