SMC AUDIT

SMART CONTRACT
SECURITY AUDIT



AUDIT REPORT

Defi Tiger (DTG) Security Review

PROJECT	Defi Tiger (DTG)
CHAIN	BNB Smart Chain (chain id 56)
CONTRACT	0xb1957BDbA889686EbdE631DF970ecE6A7571A1B6
VERSION	Verified on-chain bytecode · Sourcify exact match (matchId 48607014) · solc 0.8.2 · not a proxy
REPORT ID	SMC-DTG-202610-01
DATE	22 October 2026
PREPARED BY	SMC Audits
STATUS	Final

0 0

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.

SMC AUDITS

01

01 Contents

02	Executive summary
03	Project overview
04	Scope
05	Methodology
06	Severity classification
07	Findings summary
08	Detailed findings
09	Remediation status
10	Conclusion
11	Disclaimer

SMC AUDITS

02

02 Executive summary

SMC Audits reviewed the Defi Tiger (DTG) token contract on BNB Smart Chain at its deployed, verified address 0xb1957BDbA889686EbdE631DF970ecE6A7571A1B6. DTG is an RFI / reflection token (the SafeMoon family): a fixed-supply BEP-20 that, on each taxed transfer, reflects a share to all holders, burns a share, and routes a share to a charity address, with per-account reflection accounting and an owner-managed excluded-account set.

The review identified 8 findings: 0 critical, 0 high, 0 medium, 4 low, and 4 informational. The token is free of the dangerous permissionless classes. There is no arbitrary mint (supply is fixed at construction), no proxy or upgrade path, no exploitable reentrancy (the transfer path performs only internal accounting and holds no inline DEX swap), no blacklist, no pausing, and no path by which an unprivileged attacker can steal another holder's balance or mint value. We verified the reflection math conserves value across every transfer path and deliver, and confirmed on a chain fork that DTG is not a honeypot: a buy followed by a full sell both succeed.

The most significant code defect is that the fee setter updateFee caps each of the three fees below 100% individually but does not cap their sum, so a total fee above 100% would underflow the transfer math and permanently revert every transfer. On this deployment that lever is neutralized: ownership is renounced (owner() == address(0)), so updateFee and all other privileged functions can never be called again. We report it as Low, with the explicit note that it would be High on any live-owner redeployment of this template. The remaining findings are a permanent, now-immutable 9% fee including a fixed charity stream, two code-level accounting imprecisions that do not permit theft, and code-quality items.

0	0	0	4	4	8
CRITICAL	HIGH	MEDIUM	LOW	INFO	TOTAL

As of this report, 0 of 8 findings have been resolved; the report is delivered for the record. The overall risk of the deployed, renounced contract is low.

03

03 Project overview

Defi Tiger (DTG) is a deflationary RFI / reflection BEP-20 token on BNB Smart Chain. This audit covers the deployed token contract GoldKeyToken.sol that implements the token, its transfer and fee logic, and its reflection-based supply accounting.

System at a glance

COMPONENT	ROLE
Token contract (GoldKeyToken)	BEP-20 balances, transfers, and RFI reflection accounting.
Fee / reflection module	9% on taxed transfers: 3% reflected to holders, 3% burned, 3% to charity.
Admin / ownership	Owner renounced (owner() == address(0)), all privileged setters frozen.
External integrations	PancakeSwap V2 DTG/WBNB pair (out of scope); no oracle, bridge, or inline swap.

Key mechanisms to note

The fee is a permanent 9% split 3/3/3 (reflection / burn / charity). Reflection raises every non-excluded holder's balance through a falling rate rather than by transfers. An owner-set excluded-account set is frozen by renouncement. There is no mint, no pause, and no blacklist.

04

04 Scope

ITEM	DETAIL
Project	Defi Tiger (DTG)
Chain	BNB Smart Chain (chain id 56)
Language	Solidity
Contract / address	0xb1957BDbA889686EbdE631DF970ecE6A7571A1B6, verified source via Sourcify (matchId 48607014).
Compiler	solc 0.8.2, optimizer enabled (200 runs). Not a proxy.

Addresses

COMPONENT	ADDRESS
DTG token	0xb1957BDbA889686EbdE631DF970ecE6A7571A1B6
Charity FeeAddress	0x05021000da36d43fB81D24B1F788eaed8B2B10bF
PancakeSwap V2 pair (DTG/WBNB)	0xD2e4a524d1a932adbC70fb41F2bEC05884d5f6C2
Owner	0x0000000000000000000000000000000000000000 (renounced)

SMC AUDITS

Files reviewed

FILE	LINES	NOTES
GoldKeyToken.sol	~828	Full token source; Context, IBEP20, SafeMath, Address, Ownable, and the RFI token.

Out of scope

Off-chain components, the PancakeSwap contracts, the charity recipient's own behaviour, and market / economic conditions. The review covers the deployed token contract only.

05

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, the core of the work.
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

Access control, signer and owner checks; approvals and allowance handling; 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; oracle and price handling; upgradeability and proxy risk; pause, blacklist, and emergency switches; front-running and MEV exposure; denial of service and gas griefing.

06

06 Severity classification

SEVERITY	MEANING

SMC AUDITS

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
INFO	Best practice and code quality notes worth addressing over time

07

07 Findings summary

ID	TITLE	SEVERITY	STATUS
SMC-L-01	updateFee does not cap the total fee; a sum above 100% would brick all transfers, neutralized by renouncement	LOW	Open
SMC-L-02	Permanent, immutable 9% transfer fee including a 3% charity stream to a fixed address	LOW	Open
SMC-L-03	Dead duplicate _tOwned[FeeAddress] write; latent corruption only if the charity account were excluded	LOW	Open
SMC-L-04	Reflection rate recomputed mid-transfer causes dust-level drift on excluded-account transfers	LOW	Open
SMC-I-01	Redundant SafeMath on Solidity 0.8.x	INFO	Open
SMC-I-02	Unbounded loops over the excluded-account set	INFO	Open
SMC-I-03	Missing events on the (now-dead) privileged functions	INFO	Open
SMC-I-04	Code hygiene: floating pragma, redundant owner getter, no zero-address check on charity, unused helper	INFO	Open

0	0	0	4	4	8
CRITICAL	HIGH	MEDIUM	LOW	INFO	TOTAL

SMC AUDITS

08

08 Detailed findings

PoC note: material findings are confirmed with fork-based proofs of concept against the live bytecode. No live mainnet transaction was made. Live state verified: owner() == address(0) (renounced), Sourcify exact match.

SMC AUDITS

SMC-L-01 LOW Open

updateFee does not cap the total fee; a sum above 100% would brick all transfers

CATEGORY

Centralization / denial of service (neutralized)

LOCATION

GoldKeyToken.sol: updateFee(uint256,uint256,uint256) → getTTransferAmount(...)

DESCRIPTION

updateFee validates each fee with require(_txFee < 100 && _burnFee < 100 && _charityFee < 100), each fee independently, up to 99%. It does not bound the sum. Setting the three fees so their total exceeds 100% (for example 34 / 34 / 34 = 102%) makes getTTransferAmount = tAmount - tFee - tBurn - tCharity underflow and revert, which permanently reverts every taxed transfer. Before renouncement this was a one-way freeze / honeypot lever for the owner.

IMPACT

On this deployment the impact is neutralized: ownership is renounced (owner() == address(0)), so updateFee can never be called again and the fees are frozen at 3 / 3 / 3. Reported as Low for the record; on any redeployment of this reused template that keeps a live owner, it is a High-severity honeypot lever.

PROOF OF CONCEPT

```
function test_H1_updateFee_can_brick_all_transfers_fresh_deploy() public {
    GoldKeyToken t = new GoldKeyToken("Test","T",9,1_000_000,3,3,3, feeAddr,
address(this), svc);
    t.transfer(bob, 1_000 * 1e9);         // baseline transfer works
    t.updateFee(34, 34, 34);              // each < 100 passes; sum = 102%
    vm.expectRevert();
    t.transfer(bob, 1_000 * 1e9);         // every transfer now reverts -> bricked
}
function test_H1_neutralized_on_live_instance() public {
    assertEq(ILiveFee(DTG).owner(), address(0));   // renounced
    vm.prank(makeAddr("anyone")); vm.expectRevert();
    ILiveFee(DTG).updateFee(34, 34, 34);           // cannot be called -> lever
disabled
}
```

RECOMMENDATION

For any reuse of this template, replace the per-fee check with a total-fee cap, e.g. require(_txFee + _burnFee + _charityFee < 100), and prefer a conservative maximum well below 100%.

SMC-L-02 LOW Open

Permanent, immutable 9% transfer fee including a 3% charity stream to a fixed address

CATEGORY

Economic / design

LOCATION

GoldKeyToken.sol: _getTBasics(...), _sendToCharity(...)

DESCRIPTION

Every taxed transfer loses 9%, 3% reflected to holders, 3% burned, 3% sent to the fixed charity FeeAddress. Because ownership is renounced, the fee rate (updateFee) and the charity recipient (setAsCharityAccount) are frozen forever. A holder moving or selling tokens always pays 9%, and 3% of every transfer is an unstoppable stream to 0x05021000. The charity address is a 23-byte account consistent with an EIP-7702 delegated externally-owned account.

IMPACT

No loss beyond the disclosed fee, but integrators and holders should understand the fee and the charity stream are permanent and cannot be changed or disabled. Round-trip trading cost is roughly two 9% fees plus pool slippage.

RECOMMENDATION

None required for safety; document the permanent fee and charity stream for users and integrators.

SMC-L-03 LOW Open

Dead duplicate _tOwned[FeeAddress] write; latent corruption only if the charity account were excluded

CATEGORY

Accounting / code quality

LOCATION

GoldKeyToken.sol: _sendToCharity(...)

DESCRIPTION

_sendToCharity credits the charity to both _rOwned[FeeAddress] and _tOwned[FeeAddress] unconditionally. Because FeeAddress is not excluded, balanceOf reads _rOwned and _getCurrentSupply iterates only the excluded set, so the _tOwned[FeeAddress] write is never read, pure gas waste and an ever-growing unused value. It would only become a real balance corruption if FeeAddress were ever excluded, which is impossible now (ownership renounced).

IMPACT

None on this deployment (inert). Noted as a latent defect and gas waste.

RECOMMENDATION

Write _tOwned[FeeAddress] only inside the excluded branch, or drop the _tOwned write.

SMC AUDITS

SMC-L-04 LOW Open

Reflection rate recomputed mid-transfer causes dust-level drift on excluded-account transfers

CATEGORY

Accounting precision

LOCATION

GoldKeyToken.sol: _sendToCharity(...) (re-reads _getRate() after an excluded balance mutation)

DESCRIPTION

_getValues computes the fee reflections at one rate, but _sendToCharity recomputes the charity reflection at a freshly read rate after the excluded-account balances in the same transfer have changed _getCurrentSupply's inputs. On transfers that touch an excluded account the two rates can differ by the floor-division remainder, so the sum of reflected balances and _rTotal can drift by at most dust per such transfer. For the dominant non-excluded path and for deliver, the two rates are equal and there is no drift. The drift is never credited to an attacker and does not permit theft.

IMPACT

Dust-level imprecision on excluded-account transfers only. No exploit.

RECOMMENDATION

Pass the already-computed rate into _sendToCharity instead of re-reading _getRate().

SMC-I-01 INFO Open

Redundant SafeMath on Solidity 0.8.x

CATEGORY

Best practice / code quality

DESCRIPTION

The contract uses SafeMath throughout on Solidity 0.8.x, whose arithmetic is already checked. The library is redundant and wastes gas.

RECOMMENDATION

Use native checked arithmetic.

SMC AUDITS

SMC-I-02 INFO Open

Unbounded loops over the excluded-account set

CATEGORY

Denial of service and gas griefing

DESCRIPTION

_getCurrentSupply and includeAccount iterate over the _excluded array. A large excluded set would make these costly or revert on gas. Frozen now (ownership renounced), so inert; relevant for any template reuse.

RECOMMENDATION

Bound the excluded set or avoid unbounded iteration in future versions.

SMC-I-03 INFO Open

Missing events on the (now-dead) privileged functions

CATEGORY

Best practice / code quality

DESCRIPTION

updateFee, excludeAccount, includeAccount, and setAsCharityAccount emit no events, which weakens off-chain monitoring. These functions are now permanently disabled; relevant for template reuse.

RECOMMENDATION

Emit events on every privileged state change in future versions.

SMC-I-04 INFO Open

Code hygiene

CATEGORY

Best practice / code quality

DESCRIPTION

Floating pragma ^0.8.2 (pin the compiler), a redundant public _owner alongside owner(), no zero-address check on FeeAddress in the constructor, and an unused private _getTaxFee(). None affect the deployed contract's security.

RECOMMENDATION

Pin the pragma, remove the redundant getter and unused helper, and add a zero-address check in future versions.

SMC AUDITS

09

09 Remediation status

After the project addresses the findings, SMC Audits re-audits the changes and records the outcome here. The deployed contract is immutable in effect: ownership is renounced, so the privileged setters are frozen and the fee, exclusions and charity address cannot change. The findings are delivered for the record and for any future redeployment of this template.

ID	SEVERITY	STATUS	VERIFIED FIX
SMC-L-01	LOW	Open	Neutralized on this deployment by renouncement. Cap the total fee on any template reuse.
SMC-L-02	LOW	Open	Pending project remediation. Document the permanent fee and charity stream.
SMC-L-03	LOW	Open	Pending project remediation.
SMC-L-04	LOW	Open	Pending project remediation.
SMC-I-01..I-04	INFO	Open	Pending project remediation.

SMC AUDITS

10

10 Conclusion

The deployed Defi Tiger (DTG) token is low risk. It is free of the dangerous permissionless classes: no arbitrary mint (supply is fixed), no proxy or upgrade path, no exploitable reentrancy (the transfer path holds no inline DEX swap and makes no external calls), no oracle or flash-loan surface, no blacklist or pause, and no path by which an unprivileged attacker can steal holder balances or mint value.

The RFI reflection math conserves value across every transfer path and deliver, and the token is not a honeypot, a buy followed by a full sell both succeed on-chain. The one meaningful code defect is that updateFee caps each fee below 100% but not their sum, which would let a live owner set a total fee above 100% and permanently brick all transfers. On this deployment that lever is neutralized because ownership is renounced, so fees, exclusions, and the charity address are frozen as deployed. We recommend, for any reuse of this template, capping the total fee, removing the redundant _tOwned charity write, passing the computed rate into the charity step, and adding events and a total-fee bound.

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

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 · CONFIDENTIAL · SMCAUDITS.COM

SMC AUDITS