EVM
SMART AUDIT

SMART CONTRACT SECURITY AUDIT

Smart Contract Security Audit

Hybrid static & AI-assisted analysis

AmantaraToken

Chain: bsc-testnet **Compiler:** v0.8.28+commit.7893614a

Address: `0x3e881D03bdA2Ff2D32f2Dd74828ee1F0Bbb82E1D`

Medium Risk

22 detailed findings · 0 severity ≥ high

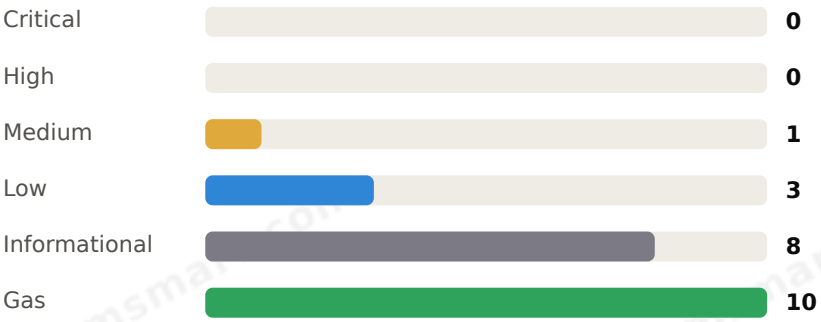
Medium **1** Low **3** Informational **8**

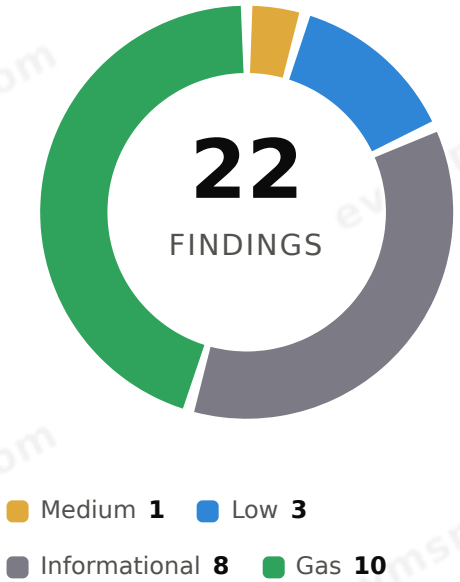
Gas **10**

EXECUTIVE SUMMARY

Executive Summary

Static analysis of AmantaraToken produced 22 finding(s): 1 medium, 3 low, 8 informational, 10 gas. Review the high-severity items first. Enable AI analysis for deeper business-logic and economic-attack coverage.





Scope & Methodology

Contract	**AmantaraToken**
Chain	bsc-testnet
Address	`0x3e881D03bdA2Ff2D32f2Dd74828ee1F0Bbb82E1D`
Compiler	v0.8.28+commit.7893614a
Files analyzed	8

ANALYSIS ENGINES

- Built-in detectors — Enabled
- Slither — Not run
- AI deep analysis — Not run

Contract Overview

986	5	45
Lines of code	Contracts	Functions

0	10	8
External calls	State variables	Files analyzed

Contracts

CONTRACT	KIND	INHERITS
Ownable	abstract	Context
IERC20Errors	interface	—
IERC721Errors	interface	—
IERC1155Errors	interface	—
ERC20	abstract	Context, IERC20, IERC20Metadata, IERC20Errors
ERC20Capped	abstract	ERC20
IERC20Metadata	interface	IERC20
IERC20	interface	—
Context	abstract	—
AmantaraToken	contract	ERC20, ERC20Capped, Ownable

FUNCTION SURFACE

External	**13**
Public	**18**
Internal	**14**
Private	**0**

Payable: 0 | View / Pure: 18

Externally exposed: 31

STANDARDS COVERAGE

Standards Coverage

AmantaraToken was reviewed against the following industry checklists; flagged items map to findings in this report.

OWASP Smart Contract Top 10 (2025)

7/10 clear

!	SC01	Access Control Vulnerabilities	**1 flagged**
✓	SC02	Price Oracle Manipulation	No issues
✓	SC03	Logic Errors	No issues
!	SC04	Lack of Input Validation	**3 flagged**
✓	SC05	Reentrancy Attacks	No issues
✓	SC06	Unchecked External Calls	No issues
✓	SC07	Flash Loan Attacks	No issues
!	SC08	Integer Overflow & Underflow	**4 flagged**
✓	SC09	Insecure Randomness	No issues
✓	SC10	Denial of Service	No issues

SWC Registry

10/12 clear

✓	SWC-107	Reentrancy	No issues
✓	SWC-105	Unprotected access / privileged funcs	No issues
✓	SWC-106	Unprotected selfdestruct	No issues
✓	SWC-115	Authorization via tx.origin	No issues
!	SWC-101	Integer overflow / underflow	**4 flagged**
✓	SWC-104	Unchecked call return value	No issues
✓	SWC-112	Delegatecall to untrusted callee	No issues
✓	SWC-120	Weak randomness	No issues
✓	SWC-116	Block-values as time proxy	No issues
✓	SWC-128	DoS by gas / unbounded ops	No issues
!	SWC-103	Floating pragma	**1 flagged**
✓	SWC-119	Variable shadowing	No issues

DETECTORS EXECUTED · 20

Potential reentrancy (state write after external call) · tx.origin used for authorization · delegatecall to a potentially user-controlled address · Return value of low-level call not checked · selfdestruct callable without access control · Weak randomness from block/tx properties · block.timestamp dependence · Privileged function without access control · Owner holds powerful privileges (centralization) · Floating pragma · Arithmetic without overflow protection · unchecked arithmetic block · Missing zero-address validation · Sensitive state change without event · Shadowed variable declarations · Loop over dynamic data (gas / DoS) · Array length read in loop condition · Post-increment in loop · Use custom errors instead of require strings · Public function never called internally

Findings Summary

#	SEVERITY	FINDING	STANDARDS	LOCATION
01	MEDIUM	**High owner privilege / centralization risk** Centralization · medium confidence	SC01	`AmantaraToken.sol:121`
02	LOW	**No zero-address check in `_transferOwnership`** Input Validation · low confidence	SC04	`Ownable.sol:95`
03	LOW	**No zero-address check in `constructor`** Input Validation · low confidence	SC04	`AmantaraToken.sol:103`
04	LOW	**No zero-address check in `_update`** Input Validation · low confidence	SC04	`AmantaraToken.sol:212`
05	INFORMATIONAL	**Floating pragma** Best Practice · high confidence	SWC-103	`Ownable.sol:4`
06	INFORMATIONAL	**unchecked{} block** Arithmetic · low confidence	SWC-101 · SC08	`ERC20.sol:185`
07	INFORMATIONAL	**unchecked{} block** Arithmetic · low confidence	SWC-101 · SC08	`ERC20.sol:192`
08	INFORMATIONAL	**unchecked{} block** Arithmetic · low confidence	SWC-101 · SC08	`ERC20.sol:197`
09	INFORMATIONAL	**unchecked{} block** Arithmetic · low confidence	SWC-101 · SC08	`ERC20.sol:300`
10	INFORMATIONAL	**No event emitted in `transferOwnership`** Best Practice · low confidence	—	`Ownable.sol:84`
11	INFORMATIONAL	**No event emitted in `_update`** Best Practice · low confidence	—	`ERC20Capped.sol:43`
12	INFORMATIONAL	**No event emitted in `_update`** Best Practice · low confidence	—	`AmantaraToken.sol:212`

Gas Optimizations

#	SEVERITY	FINDING	STANDARDS	LOCATION
01	GAS	**`renounceOwnership` can be external** Gas Optimization · low confidence	—	`Ownable.sol:76`
02	GAS	**`transferOwnership` can be external** Gas Optimization · low confidence	—	`Ownable.sol:84`

03	GAS	**`name` can be external** Gas Optimization · low confidence	–	ERC20.sol:52
04	GAS	**`symbol` can be external** Gas Optimization · low confidence	–	ERC20.sol:60
05	GAS	**`decimals` can be external** Gas Optimization · low confidence	–	ERC20.sol:77
06	GAS	**`balanceOf` can be external** Gas Optimization · low confidence	–	ERC20.sol:87
07	GAS	**`transfer` can be external** Gas Optimization · low confidence	–	ERC20.sol:99
08	GAS	**`approve` can be external** Gas Optimization · low confidence	–	ERC20.sol:120
09	GAS	**`transferFrom` can be external** Gas Optimization · low confidence	–	ERC20.sol:142
10	GAS	**`transferFrom` can be external** Gas Optimization · low confidence	–	AmantaraToken.sol:195

DETAILED FINDINGS

Detailed Findings

01 High owner privilege / centralization risk

MEDIUM

SEVERITY
Medium

CONFIDENCE
medium

ATTACK PATH
Unauthorized caller → Centralization →
State corruption

LOCATION
AmantaraToken.sol
L121-132 / ~225

SC01 OWASP · Access Control

Detected by: STATIC

```
 */
function mint(address to, uint256 amount) external onlyOwner {
    if (mintingFinished) {
        revert MintingIsFinished();
    }
    if (to == address(0)) {
        revert ZeroAddress();
    }
    if (amount == 0) {
        revert ZeroAmount();
    }
    _mint(to, amount);
}
```

Description. The owner/admin can perform several powerful actions: AmantaraToken.mint, AmantaraToken.finishMinting, AmantaraToken.setBlacklisted, AmantaraToken.finishBlacklisting.

Impact. A compromised or malicious owner key can mint unlimited tokens, freeze transfers, blacklist holders, or alter fees — a rug-pull vector.

Recommendation. Transfer privileged roles to a timelock + multisig, renounce unnecessary powers, and document the trust model for users.

02 No zero-address check in `_transferOwnership` LOW

SEVERITY: Low | CONFIDENCE: low | ATTACK PATH: Malicious input → Input Validation → Deviation | LOCATION: Ownable.sol L95-99 / ~101

SC04 OWASP · Input Validation

Detected by: STATIC

```
     */
    function _transferOwnership(address newOwner) internal virtual {
        address oldOwner = _owner;
        _owner = newOwner;
        emit OwnershipTransferred(oldOwner, newOwner);
    }
}
```

Description. `_transferOwnership` takes address parameter(s) (newOwner) and appears to assign them without validating against `address(0)`.

Impact. Setting a critical address to the zero address can permanently disable functionality or lock funds.

Recommendation. Add `require(addr != address(0), "zero address");` before assigning address parameters.

03 No zero-address check in `constructor` LOW

SEVERITY: Low | CONFIDENCE: low | ATTACK PATH: Malicious input → Input Validation → Deviation | LOCATION: AmantaraToken.sol L103-111 / ~225

SC04 OWASP · Input Validation

Detected by: STATIC

```
     */
    constructor(address initialOwner, uint256 cap_, uint256 initialSupply)
        ERC20("AMANTARA", "AUSD")
        ERC20Capped(cap_)
        Ownable(initialOwner)
    {
        if (initialSupply != 0) {
            _mint(initialOwner, initialSupply);
        }
    }
```

Description. `constructor` takes address parameter(s) (initialOwner) and appears to assign them without validating against `address(0)`.

Impact. Setting a critical address to the zero address can permanently disable functionality or lock funds.

Recommendation. Add `require(addr != address(0), "zero address");` before assigning address parameters.

04 No zero-address check in `_update`

LOW

SEVERITY
Low

CONFIDENCE
low

ATTACK PATH
Malicious input → Input Validation → **Deviation**

LOCATION
AmantaraToken.sol
L212-223 / ~225

SC04 OWASP · Input Validation

Detected by: STATIC

```
     */
    function _update(address from, address to, uint256 value) internal override(ERC20, ERC20Capped) {
        if (!blacklistingFinished) {
            if (isBlacklisted[from]) {
                revert AccountIsBlacklisted(from);
            }
            if (isBlacklisted[to]) {
                revert AccountIsBlacklisted(to);
            }
        }

        super._update(from, to, value);
    }
}
```

Description. `_update` takes address parameter(s) (from, to) and appears to assign them without validating against `address(0)`.

Impact. Setting a critical address to the zero address can permanently disable functionality or lock funds.

Recommendation. Add `require(addr != address(0), "zero address");` before assigning address parameters.

05 Floating pragma

INFORMATIONAL SWC-103

SEVERITY
Informational

CONFIDENCE
high

ATTACK PATH
Malicious input → Best Practice → **Deviation**

LOCATION
Ownable.sol
L4 / ~101

SWC-103 SWC

Detected by: STATIC

```
pragma solidity ^0.8.20;
```

Description. The pragma `^0.8.20` is not locked to a specific compiler version.

Impact. Contracts may be compiled with a different, possibly buggy, compiler version than the one tested.

Recommendation. Lock the pragma to a specific version, e.g. `pragma solidity 0.8.24;`.

06 unchecked{} block

INFORMATIONAL SWC-101

SEVERITY
Informational

CONFIDENCE
low

ATTACK PATH
Crafted amount → Arithmetic → **Deviation**

LOCATION
ERC20.sol
L185-188 / ~306

SWC-101 SWC **SC08** OWASP · Overflow

Detected by: STATIC

```
        }
        unchecked {
            // Overflow not possible: value <= fromBalance <= totalSupply.
            _balances[from] = fromBalance - value;
        }
    }
```

Description. An `unchecked` block disables overflow/underflow protection for the enclosed arithmetic.

Impact. If bounds are not otherwise guaranteed, arithmetic here may silently wrap around.

Recommendation. Confirm operands cannot overflow within the unchecked block; document the invariant.

07 unchecked{} block

INFORMATIONAL SWC-101

SEVERITY
Informational

CONFIDENCE
low

ATTACK PATH
Crafted amount → Arithmetic → **Deviation**

LOCATION
ERC20.sol
L192-195 / ~306

SWC-101 SWC **SC08** OWASP · Overflow

Detected by: STATIC

```
        if (to == address(0)) {
            unchecked {
                // Overflow not possible: value <= totalSupply or value <= fromBalance <= totalSupply.
                _totalSupply -= value;
            }
        } else {
```

Description. An `unchecked` block disables overflow/underflow protection for the enclosed arithmetic.

Impact. If bounds are not otherwise guaranteed, arithmetic here may silently wrap around.

Recommendation. Confirm operands cannot overflow within the unchecked block; document the invariant.

08 unchecked{} block

INFORMATIONAL SWC-101

SEVERITY
Informational

CONFIDENCE
low

ATTACK PATH
Crafted amount → Arithmetic → **Deviation**

LOCATION
ERC20.sol
L197-200 / ~306

SWC-101 SWC **SC08** OWASP · Overflow

Detected by: **STATIC**

```
        } else {
            unchecked {
                // Overflow not possible: balance + value is at most totalSupply, which we know fits into a
                _balances[to] += value;
            }
        }
```

Description. An `unchecked` block disables overflow/underflow protection for the enclosed arithmetic.

Impact. If bounds are not otherwise guaranteed, arithmetic here may silently wrap around.

Recommendation. Confirm operands cannot overflow within the unchecked block; document the invariant.

09 unchecked{} block

INFORMATIONAL SWC-101

SEVERITY
Informational

CONFIDENCE
low

ATTACK PATH
Crafted amount → Arithmetic → **Deviation**

LOCATION
ERC20.sol
L300-302 / ~306

SWC-101 SWC **SC08** OWASP · Overflow

Detected by: **STATIC**

```
        }
        unchecked {
            _approve(owner, spender, currentAllowance - value, false);
        }
    }
```

Description. An `unchecked` block disables overflow/underflow protection for the enclosed arithmetic.

Impact. If bounds are not otherwise guaranteed, arithmetic here may silently wrap around.

Recommendation. Confirm operands cannot overflow within the unchecked block; document the invariant.

10 No event emitted in `transferOwnership`

INFORMATIONAL

SEVERITY: Informational

CONFIDENCE: low

ATTACK PATH: Malicious input → Best Practice → **Deviation**

LOCATION: Ownable.sol L84-89 / ~101

Detected by: STATIC

```
     */
    function transferOwnership(address newOwner) public virtual onlyOwner {
        if (newOwner == address(0)) {
            revert OwnableInvalidOwner(address(0));
        }
        _transferOwnership(newOwner);
    }
```

Description. `transferOwnership` changes sensitive configuration/state but emits no event.

Impact. Off-chain monitors and users cannot track critical changes, hindering transparency and incident response.

Recommendation. Emit an event capturing the old and new values for every sensitive state change.

11 No event emitted in `_update`

INFORMATIONAL

SEVERITY: Informational

CONFIDENCE: low

ATTACK PATH: Malicious input → Best Practice → **Deviation**

LOCATION: ERC20Capped.sol L43-53 / ~55

Detected by: STATIC

```
    /// @inheritdoc ERC20
    function _update(address from, address to, uint256 value) internal virtual override {
        super._update(from, to, value);

        if (from == address(0)) {
            uint256 maxSupply = cap();
            uint256 supply = totalSupply();
            if (supply > maxSupply) {
                revert ERC20ExceededCap(supply, maxSupply);
            }
        }
    }
}
```

Description. `_update` changes sensitive configuration/state but emits no event.

Impact. Off-chain monitors and users cannot track critical changes, hindering transparency and incident response.

Recommendation. Emit an event capturing the old and new values for every sensitive state change.

12 No event emitted in `_update`

INFORMATIONAL

SEVERITY
Informational

CONFIDENCE
low

ATTACK PATH
Malicious input → Best Practice →
Deviation

LOCATION
AmantaraToken.sol
L212-223 / ~225

Detected by: STATIC

```
     */
    function _update(address from, address to, uint256 value) internal override(ERC20, ERC20Capped) {
        if (!blacklistingFinished) {
            if (isBlacklisted[from]) {
                revert AccountIsBlacklisted(from);
            }
            if (isBlacklisted[to]) {
                revert AccountIsBlacklisted(to);
            }
        }

        super._update(from, to, value);
    }
}
```

Description. `_update` changes sensitive configuration/state but emits no event.

Impact. Off-chain monitors and users cannot track critical changes, hindering transparency and incident response.

Recommendation. Emit an event capturing the old and new values for every sensitive state change.

Gas Optimizations

01 `renounceOwnership` can be external

GAS

SEVERITY: Gas

CONFIDENCE: low

ATTACK PATH: Normal call → Gas Optimization → **Higher gas cost**

LOCATION: Ownable.sol L76-78 / ~101

Detected by: STATIC

```
 */
function renounceOwnership() public virtual onlyOwner {
    _transferOwnership(address(0));
}
```

Description. `renounceOwnership` is `public` but appears to never be called internally.

Impact. Public functions copy arguments to memory; external is cheaper for calldata-only use.

Recommendation. Mark it `external` if it is not called from within the contract.

02 `transferOwnership` can be external

GAS

SEVERITY: Gas

CONFIDENCE: low

ATTACK PATH: Normal call → Gas Optimization → **Higher gas cost**

LOCATION: Ownable.sol L84-89 / ~101

Detected by: STATIC

```
 */
function transferOwnership(address newOwner) public virtual onlyOwner {
    if (newOwner == address(0)) {
        revert OwnableInvalidOwner(address(0));
    }
    _transferOwnership(newOwner);
}
```

Description. `transferOwnership` is `public` but appears to never be called internally.

Impact. Public functions copy arguments to memory; external is cheaper for calldata-only use.

Recommendation. Mark it `external` if it is not called from within the contract.

03 `name` can be external

GAS

SEVERITY
Gas

CONFIDENCE
low

ATTACK PATH
Normal call → Gas Optimization →
Higher gas cost

LOCATION
ERC20.sol
L52-54 / ~306

Detected by: STATIC

```
 */
function name() public view virtual returns (string memory) {
    return _name;
}
```

Description. `name` is `public` but appears to never be called internally.

Impact. Public functions copy arguments to memory; external is cheaper for calldata-only use.

Recommendation. Mark it `external` if it is not called from within the contract.

04 `symbol` can be external

GAS

SEVERITY
Gas

CONFIDENCE
low

ATTACK PATH
Normal call → Gas Optimization →
Higher gas cost

LOCATION
ERC20.sol
L60-62 / ~306

Detected by: STATIC

```
 */
function symbol() public view virtual returns (string memory) {
    return _symbol;
}
```

Description. `symbol` is `public` but appears to never be called internally.

Impact. Public functions copy arguments to memory; external is cheaper for calldata-only use.

Recommendation. Mark it `external` if it is not called from within the contract.

05 `decimals` can be external

GAS

SEVERITY
Gas

CONFIDENCE
low

ATTACK PATH
Normal call → Gas Optimization →
Higher gas cost

LOCATION
ERC20.sol
L77-79 / ~306

Detected by: STATIC

```
 */
function decimals() public view virtual returns (uint8) {
    return 18;
}
```

Description. `decimals` is `public` but appears to never be called internally.

Impact. Public functions copy arguments to memory; external is cheaper for calldata-only use.

Recommendation. Mark it `external` if it is not called from within the contract.

06 `balanceOf` can be external

GAS

SEVERITY
Gas

CONFIDENCE
low

ATTACK PATH
Normal call → Gas Optimization →
Higher gas cost

LOCATION
ERC20.sol
L87-89 / ~306

Detected by: STATIC

```
/// @inheritdoc IERC20
function balanceOf(address account) public view virtual returns (uint256) {
    return _balances[account];
}
```

Description. `balanceOf` is `public` but appears to never be called internally.

Impact. Public functions copy arguments to memory; external is cheaper for calldata-only use.

Recommendation. Mark it `external` if it is not called from within the contract.

07 `transfer` can be external

GAS

SEVERITY

Gas

CONFIDENCE

low

ATTACK PATH

Normal call → Gas Optimization →

Higher gas cost

LOCATION

ERC20.sol

L99-103 / ~306

Detected by: STATIC

```
     */
    function transfer(address to, uint256 value) public virtual returns (bool) {
        address owner = _msgSender();
        _transfer(owner, to, value);
        return true;
    }
```

Description. `transfer` is `public` but appears to never be called internally.

Impact. Public functions copy arguments to memory; external is cheaper for calldata-only use.

Recommendation. Mark it `external` if it is not called from within the contract.

08 `approve` can be external

GAS

SEVERITY

Gas

CONFIDENCE

low

ATTACK PATH

Normal call → Gas Optimization →

Higher gas cost

LOCATION

ERC20.sol

L120-124 / ~306

Detected by: STATIC

```
     */
    function approve(address spender, uint256 value) public virtual returns (bool) {
        address owner = _msgSender();
        _approve(owner, spender, value);
        return true;
    }
```

Description. `approve` is `public` but appears to never be called internally.

Impact. Public functions copy arguments to memory; external is cheaper for calldata-only use.

Recommendation. Mark it `external` if it is not called from within the contract.

09 `transferFrom` can be external

GAS

SEVERITY
Gas

CONFIDENCE
low

ATTACK PATH
Normal call → Gas Optimization →
Higher gas cost

LOCATION
ERC20.sol
L142-147 / ~306

Detected by: STATIC

```
 */
function transferFrom(address from, address to, uint256 value) public virtual returns (bool) {
    address spender = _msgSender();
    _spendAllowance(from, spender, value);
    _transfer(from, to, value);
    return true;
}
```

Description. `transferFrom` is `public` but appears to never be called internally.

Impact. Public functions copy arguments to memory; external is cheaper for calldata-only use.

Recommendation. Mark it `external` if it is not called from within the contract.

10 `transferFrom` can be external

GAS

SEVERITY
Gas

CONFIDENCE
low

ATTACK PATH
Normal call → Gas Optimization →
Higher gas cost

LOCATION
AmantaraToken.sol
L195-200 / ~225

Detected by: STATIC

```
 */
function transferFrom(address from, address to, uint256 value) public override returns (bool success) {
    if (!blacklistingFinished && isBlacklisted[_msgSender()]) {
        revert AccountIsBlacklisted(_msgSender());
    }
    return super.transferFrom(from, to, value);
}
```

Description. `transferFrom` is `public` but appears to never be called internally.

Impact. Public functions copy arguments to memory; external is cheaper for calldata-only use.

Recommendation. Mark it `external` if it is not called from within the contract.

Appendix

Severity Classification

SEVERITY	DESCRIPTION
CRITICAL	Directly exploitable; leads to loss of funds or renders the contract unusable.
HIGH	Serious vulnerability that can be exploited under realistic conditions.
MEDIUM	Exploitable under specific conditions or with limited impact.
LOW	Minor issue or deviation from best practice with low impact.
INFORMATIONAL	Observation or note; no direct security impact.
GAS	Opportunity to reduce gas consumption.

Disclaimer. This report is provided for informational purposes only and does not constitute a warranty regarding the security of the audited code. Automated and AI-assisted analysis cannot guarantee the absence of vulnerabilities. A manual review by qualified auditors is recommended before deploying to production.