AuditLens

SMART CONTRACT SECURITY AUDIT

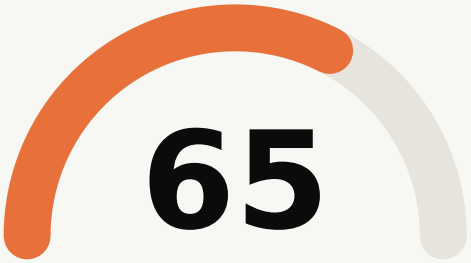
Smart Contract Security Audit

Hybrid static & AI-assisted analysis

DAN20Token_0001

Chain: danny **Compiler:** v0.8.29+commit.ab55807c

Address: `0x6F22D0eaB6DdAC3d36B9F3043c3524FD4B7936a7`



High Risk

24 detailed findings · 3 severity ≥ high

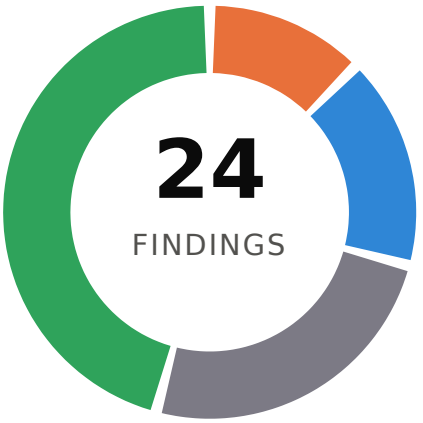
High **3** Low **4** Informational **6** Gas **11**

EXECUTIVE SUMMARY

Executive Summary

Static analysis of DAN20Token_0001 produced 24 finding(s): 3 high, 4 low, 6 informational, 11 gas. Review the high-severity items first. Enable AI analysis for deeper business-logic and economic-attack coverage.

Critical	0
High	3
Medium	0
Low	4
Informational	6
Gas	11



High **3** Low **4**
Informational **6** Gas **11**

Scope & Methodology

Contract	**DAN20Token_0001**
Chain	danny
Address	`0x6F22D0eaB6DdAC3d36B9F3043c3524FD4B7936a7`
Compiler	v0.8.29+commit.ab55807c
Files analyzed	1

ANALYSIS ENGINES

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

Contract Overview

698	3	47
Lines of code	Contracts	Functions

4	8	1
External calls	State variables	Files analyzed

Contracts

CONTRACT	KIND	INHERITS
Address	library	—
Initializable	abstract	—
IERC20	interface	—
IERC20Metadata	interface	IERC20
ERC20	contract	Initializable, IERC20Metadata
DAN20Token_0001	contract	Initializable, ERC20

FUNCTION SURFACE

External	**10**
Public	**13**
Internal	**23**
Private	**1**

Payable: 0

View / Pure: 20

Externally exposed: 23

STANDARDS COVERAGE

Standards Coverage

DAN20Token_0001 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	**2 flagged**
✓	SC02	Price Oracle Manipulation	No issues
✓	SC03	Logic Errors	No issues
!	SC04	Lack of Input Validation	**1 flagged**
✓	SC05	Reentrancy Attacks	No issues
✓	SC06	Unchecked External Calls	No issues
✓	SC07	Flash Loan Attacks	No issues
!	SC08	Integer Overflow & Underflow	**5 flagged**
✓	SC09	Insecure Randomness	No issues
✓	SC10	Denial of Service	No issues

SWC Registry

7/12 clear

✓	SWC-107	Reentrancy	No issues
!	SWC-105	Unprotected access / privileged funcs	**2 flagged**
✓	SWC-106	Unprotected selfdestruct	No issues
✓	SWC-115	Authorization via tx.origin	No issues
!	SWC-101	Integer overflow / underflow	**5 flagged**
✓	SWC-104	Unchecked call return value	No issues
!	SWC-112	Delegatecall to untrusted callee	**1 flagged**
✓	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	**3 flagged**

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

Findings Summary

#	SEVERITY	FINDING	STANDARDS	LOCATION
01	HIGH	**delegatecall usage** Unsafe Delegatecall · medium confidence	SWC-112	DAN20Token_0001.sol:192
02	HIGH	**Unprotected privileged function `burn`** Access Control · medium confidence	SWC-105 · SC01	DAN20Token_0001.sol:688
03	HIGH	**Unprotected privileged function `burnFrom`** Access Control · medium confidence	SWC-105 · SC01	DAN20Token_0001.sol:692
04	LOW	**Parameter `_name` shadows a state variable** Shadowing · medium confidence	SWC-119	DAN20Token_0001.sol:676
05	LOW	**Parameter `_symbol` shadows a state variable** Shadowing · medium confidence	SWC-119	DAN20Token_0001.sol:676
06	LOW	**Parameter `_decimals` shadows a state variable** Shadowing · medium confidence	SWC-119	DAN20Token_0001.sol:676
07	LOW	**No zero-address check in `initialize`** Input Validation · low confidence	SC04	DAN20Token_0001.sol:676
08	INFORMATIONAL	**Floating pragma** Best Practice · high confidence	SWC-103	DAN20Token_0001.sol:665
09	INFORMATIONAL	**unchecked{} block** Arithmetic · low confidence	SWC-101 · SC08	DAN20Token_0001.sol:599
10	INFORMATIONAL	**unchecked{} block** Arithmetic · low confidence	SWC-101 · SC08	DAN20Token_0001.sol:611
11	INFORMATIONAL	**unchecked{} block** Arithmetic · low confidence	SWC-101 · SC08	DAN20Token_0001.sol:623
12	INFORMATIONAL	**unchecked{} block** Arithmetic · low confidence	SWC-101 · SC08	DAN20Token_0001.sol:633
13	INFORMATIONAL	**unchecked{} block** Arithmetic · low confidence	SWC-101 · SC08	DAN20Token_0001.sol:652

Gas Optimizations

#	SEVERITY	FINDING	STANDARDS	LOCATION

01	GAS	**Custom errors could replace 18 require strings** Gas Optimization · high confidence	—	`DAN20Token_0001.sol:72`
02	GAS	**`name` can be external** Gas Optimization · low confidence	—	`DAN20Token_0001.sol:561`
03	GAS	**`symbol` can be external** Gas Optimization · low confidence	—	`DAN20Token_0001.sol:562`
04	GAS	**`decimals` can be external** Gas Optimization · low confidence	—	`DAN20Token_0001.sol:563`
05	GAS	**`totalSupply` can be external** Gas Optimization · low confidence	—	`DAN20Token_0001.sol:564`
06	GAS	**`balanceOf` can be external** Gas Optimization · low confidence	—	`DAN20Token_0001.sol:567`
07	GAS	**`transfer` can be external** Gas Optimization · low confidence	—	`DAN20Token_0001.sol:571`
08	GAS	**`approve` can be external** Gas Optimization · low confidence	—	`DAN20Token_0001.sol:580`
09	GAS	**`transferFrom` can be external** Gas Optimization · low confidence	—	`DAN20Token_0001.sol:585`
10	GAS	**`increaseAllowance` can be external** Gas Optimization · low confidence	—	`DAN20Token_0001.sol:591`
11	GAS	**`decreaseAllowance` can be external** Gas Optimization · low confidence	—	`DAN20Token_0001.sol:596`

DETAILED FINDINGS

Detailed Findings

01 delegatecall usage

HIGH SWC-112

SEVERITY
High

CONFIDENCE
medium

ATTACK PATH
Point to attacker code →
Unsafe Delegatecall → **Funds at risk**

LOCATION
DAN20Token_0001.sol
L192 / ~698

SWC-112 SWC

Detected by: **STATIC**

```
    ) internal returns (bytes memory) {
        (bool success, bytes memory returndata) = target.delegatecall(data);
        return verifyCallResultFromTarget(target, success, returndata, errorMessage);
```

Description. `delegatecall` executes external code in this contract's storage context (target: `target`).

Impact. If the target address is user-controlled, an attacker can execute arbitrary code against this contract's storage, including overwriting the owner or self-destructing it.

Recommendation. Ensure the delegatecall target is a trusted, immutable address; validate against an allowlist and never derive it from untrusted input.

02 Unprotected privileged function `burn`

HIGH SWC-105

SEVERITY
High

CONFIDENCE
medium

ATTACK PATH
Unauthorized caller → Access Control → **Funds at risk**

LOCATION
DAN20Token_0001.sol
L688-690 / ~698

SWC-105 SWC **SC01** OWASP · Access Control

Detected by: **STATIC**

```
function burn(uint256 amount) public {
    _burn(msg.sender, amount);
}
```

Description. `DAN20Token_0001.burn` is externally callable and changes state but has no access-control modifier or visible `msg.sender` check, despite a privileged-sounding name.

Impact. Any account may invoke this privileged operation (e.g. minting, withdrawing, pausing), potentially draining or bricking the contract.

Recommendation. Restrict the function with an access-control modifier such as `onlyOwner` or OpenZeppelin `AccessControl` roles.

03 Unprotected privileged function `burnFrom`

HIGH SWC-105

SEVERITY
High

CONFIDENCE
medium

ATTACK PATH
Unauthorized caller → Access Control → **Funds at risk**

LOCATION
DAN20Token_0001.sol
L692-695 / ~698

SWC-105 SWC **SC01** OWASP · Access Control

Detected by: **STATIC**

```
function burnFrom(address account, uint256 amount) public {
    _spendAllowance(account, msg.sender, amount);
    _burn(account, amount);
}
```

Description. `DAN20Token_0001.burnFrom` is externally callable and changes state but has no access-control modifier or visible `msg.sender` check, despite a privileged-sounding name.

Impact. Any account may invoke this privileged operation (e.g. minting, withdrawing, pausing), potentially draining or bricking the contract.

Recommendation. Restrict the function with an access-control modifier such as `onlyOwner` or OpenZeppelin `AccessControl` roles.

04 Parameter `_name` shadows a state variable

LOW SWC-119

SEVERITY
Low

CONFIDENCE
medium

ATTACK PATH
Malicious input → Shadowing → **Deviation**

LOCATION
DAN20Token_0001.sol
L676-686 / ~698

SWC-119 SWC

Detected by: **STATIC**

```
function initialize(
    address _owner,
    string memory _name,
    string memory _symbol,
    uint8 _decimals,
    uint256 _initialSupply
) external initializer {
    uint256 _supply = _initialSupply * 10 ** _decimals;
    ERC20.init(_name, _symbol, _decimals);
    _mint(_owner, _supply);
}
```

Description. Parameter/return `_name` in `initialize` shadows the state variable `_name`.

Impact. Reads/writes may target the local variable instead of the intended state variable, causing silent logic bugs.

Recommendation. Rename the local variable or state variable to remove the shadowing.

05 Parameter `_symbol` shadows a state variable

LOW SWC-119

SEVERITY
Low

CONFIDENCE
medium

ATTACK PATH
Malicious input → Shadowing → **Deviation**

LOCATION
DAN20Token_0001.sol
L676-686 / ~698

SWC-119 SWC

Detected by: **STATIC**

```
function initialize(
    address _owner,
    string memory _name,
    string memory _symbol,
    uint8 _decimals,
    uint256 _initialSupply
) external initializer {
    uint256 _supply = _initialSupply * 10 ** _decimals;
    ERC20.init(_name, _symbol, _decimals);
    _mint(_owner, _supply);
}
```

Description. Parameter/return `_symbol` in `initialize` shadows the state variable `_symbol`.

Impact. Reads/writes may target the local variable instead of the intended state variable, causing silent logic bugs.

Recommendation. Rename the local variable or state variable to remove the shadowing.

06 Parameter `_decimals` shadows a state variable

LOW SWC-119

SEVERITY
Low

CONFIDENCE
medium

ATTACK PATH
Malicious input → Shadowing → **Deviation**

LOCATION
DAN20Token_0001.sol
L676-686 / ~698

SWC-119 SWC

Detected by: **STATIC**

```
function initialize(
    address _owner,
    string memory _name,
    string memory _symbol,
    uint8 _decimals,
    uint256 _initialSupply
) external initializer {
    uint256 _supply = _initialSupply * 10 ** _decimals;
    ERC20.init(_name, _symbol, _decimals);
    _mint(_owner, _supply);
}
```

Description. Parameter/return `_decimals` in `initialize` shadows the state variable `_decimals`.

Impact. Reads/writes may target the local variable instead of the intended state variable, causing silent logic bugs.

Recommendation. Rename the local variable or state variable to remove the shadowing.

07 No zero-address check in `initialize`

LOW

SEVERITY
Low

CONFIDENCE
low

ATTACK PATH
Malicious input → Input Validation →
Deviation

LOCATION
DAN20Token_0001.sol
L676-686 / ~698

SC04 OWASP · Input Validation

Detected by: STATIC

```
function initialize(
    address _owner,
    string memory _name,
    string memory _symbol,
    uint8 _decimals,
    uint256 _initialSupply
) external initializer {
    uint256 _supply = _initialSupply * 10 ** _decimals;
    ERC20.init(_name, _symbol, _decimals);
    _mint(_owner, _supply);
}
```

Description. `initialize` takes address parameter(s) (_owner) 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.

08 Floating pragma

INFORMATIONAL SWC-103

SEVERITY
Informational

CONFIDENCE
high

ATTACK PATH
Malicious input → Best Practice →
Deviation

LOCATION
DAN20Token_0001.sol
L665 / ~698

SWC-103 SWC

Detected by: STATIC

```
// Original license: SPDX_License_Identifier: MIT
pragma solidity ^0.8.19;
```

Description. The pragma `^0.8.1` 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;`.

09 unchecked{} block

INFORMATIONAL SWC-101

SEVERITY
Informational

CONFIDENCE
low

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

LOCATION
DAN20Token_0001.sol
L599 / ~698

SWC-101 SWC **SC08** OWASP · Overflow

Detected by: **STATIC**

```
        require(currentAllowance >= subtractedValue, "ERC20: decreased allowance below zero");
        unchecked { _approve(msg.sender, spender, currentAllowance - subtractedValue); }
        return true;
```

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 unchecked{} block

INFORMATIONAL SWC-101

SEVERITY
Informational

CONFIDENCE
low

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

LOCATION
DAN20Token_0001.sol
L611-614 / ~698

SWC-101 SWC **SC08** OWASP · Overflow

Detected by: **STATIC**

```
        require(fromBalance >= amount, "ERC20: transfer amount exceeds balance");
        unchecked {
            _balances[from] = fromBalance - amount;
            _balances[to]  += amount;
        }
        emit Transfer(from, to, amount);
```

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.

11 unchecked{} block

INFORMATIONAL SWC-101

SEVERITY
Informational

CONFIDENCE
low

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

LOCATION
DAN20Token_0001.sol
L623 / ~698

SWC-101 SWC **SC08** OWASP · Overflow

Detected by: **STATIC**

```
        _totalSupply += amount;
        unchecked { _balances[account] += amount; }
        emit Transfer(address(0), account, amount);
```

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.

12 unchecked{} block

INFORMATIONAL SWC-101

SEVERITY
Informational

CONFIDENCE
low

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

LOCATION
DAN20Token_0001.sol
L633-636 / ~698

SWC-101 SWC **SC08** OWASP · Overflow

Detected by: **STATIC**

```
        require(accountBalance >= amount, "ERC20: burn amount exceeds balance");
        unchecked {
            _balances[account] = accountBalance - amount;
            _totalSupply      -= amount;
        }
        emit Transfer(account, address(0), amount);
```

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.

13 unchecked{} block

INFORMATIONAL SWC-101

SEVERITY
Informational

CONFIDENCE
low

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

LOCATION
DAN20Token_0001.sol
L652 / ~698

SWC-101 SWC **SC08** OWASP · Overflow

Detected by: **STATIC**

```
        require(currentAllowance >= amount, "ERC20: insufficient allowance");
        unchecked { _approve(owner, spender, currentAllowance - amount); }
    }
```

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.

Gas Optimizations

01 Custom errors could replace 18 require strings

GAS

SEVERITY
Gas

CONFIDENCE
high

ATTACK PATH
Normal call → Gas Optimization →
Higher gas cost

LOCATION
DAN20Token_0001.sol
L72 / ~698

Detected by: STATIC

```
    function sendValue(address payable recipient, uint256 amount) internal {
        require(address(this).balance >= amount, "Address: insufficient balance");
```

Description. 18 `require` statements use string messages. Custom errors (Solidity ≥0.8.4) are cheaper to deploy and revert.

Impact. String revert reasons increase bytecode size and revert gas cost.

Recommendation. Replace string requires with `error Foo(); if (!cond) revert Foo();`.

02 `name` can be external

GAS

SEVERITY
Gas

CONFIDENCE
low

ATTACK PATH
Normal call → Gas Optimization →
Higher gas cost

LOCATION
DAN20Token_0001.sol
L561 / ~698

Detected by: STATIC

```
    function name()        public view virtual override returns (string memory) { return _name; }
    function symbol()      public view virtual override returns (string memory) { return _symbol; }
```

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.

03 `symbol` can be external

GAS

SEVERITY

Gas

CONFIDENCE

low

ATTACK PATH

Normal call → Gas Optimization →

Higher gas cost

LOCATION

DAN20Token_0001.sol

L562 / ~698

Detected by: STATIC

```
function name()        public view virtual override returns (string memory) { return _name; }
function symbol()      public view virtual override returns (string memory) { return _symbol; }
function decimals()    public view virtual override returns (uint8)         { return _decimals; }
```

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.

04 `decimals` can be external

GAS

SEVERITY

Gas

CONFIDENCE

low

ATTACK PATH

Normal call → Gas Optimization →

Higher gas cost

LOCATION

DAN20Token_0001.sol

L563 / ~698

Detected by: STATIC

```
function symbol()      public view virtual override returns (string memory) { return _symbol; }
function decimals()    public view virtual override returns (uint8)         { return _decimals; }
function totalSupply() public view virtual override returns (uint256)       { return _totalSupply; }
```

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.

05 `totalSupply` can be external

GAS

SEVERITY
Gas

CONFIDENCE
low

ATTACK PATH
Normal call → Gas Optimization →
Higher gas cost

LOCATION
DAN20Token_0001.sol
L564 / ~698

Detected by: STATIC

```
    function decimals()    public view virtual override returns (uint8)        { return _decimals; }
    function totalSupply() public view virtual override returns (uint256)      { return _totalSupply; }
```

Description. `totalSupply` 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
DAN20Token_0001.sol
L567-569 / ~698

Detected by: STATIC

```
    function balanceOf(address account) public view virtual override 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

DAN20Token_0001.sol

L571-574 / ~698

Detected by: STATIC

```
function transfer(address to, uint256 amount) public virtual override returns (bool) {
    _transfer(msg.sender, to, amount);
    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

DAN20Token_0001.sol

L580-583 / ~698

Detected by: STATIC

```
function approve(address spender, uint256 amount) public virtual override returns (bool) {
    _approve(msg.sender, spender, amount);
    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
DAN20Token_0001.sol
L585-589 / ~698

Detected by: STATIC

```
function transferFrom(address from, address to, uint256 amount) public virtual override returns (bool)
    _spendAllowance(from, msg.sender, amount);
    _transfer(from, to, amount);
    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 `increaseAllowance` can be external

GAS

SEVERITY
Gas

CONFIDENCE
low

ATTACK PATH
Normal call → Gas Optimization →
Higher gas cost

LOCATION
DAN20Token_0001.sol
L591-594 / ~698

Detected by: STATIC

```
function increaseAllowance(address spender, uint256 addedValue) public virtual returns (bool) {
    _approve(msg.sender, spender, allowance(msg.sender, spender) + addedValue);
    return true;
}
```

Description. `increaseAllowance` 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.

11 `decreaseAllowance` can be external

GAS

SEVERITY

Gas

CONFIDENCE

low

ATTACK PATH

Normal call → Gas Optimization →

Higher gas cost

LOCATION

DAN20Token_0001.sol

L596-601 / ~698

Detected by: STATIC

```
function decreaseAllowance(address spender, uint256 subtractedValue) public virtual returns (bool) {
    uint256 currentAllowance = allowance(msg.sender, spender);
    require(currentAllowance >= subtractedValue, "ERC20: decreased allowance below zero");
    unchecked { _approve(msg.sender, spender, currentAllowance - subtractedValue); }
    return true;
}
```

Description. `decreaseAllowance` 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.