Cyberscope

A TAC Security Company

Audit Report

NOVA AI

December 2025

Network BSC

Address 0x2Fc1493bcc38DA9921Ac749899826289a146A1Fb

Audited by © cyberscope

Analysis

Critical Medium Minor / Informative Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Passed
●	OTUT	Transfers User's Tokens	Passed
●	ELFM	Exceeds Fees Limit	Passed
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Passed

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
	ROF	Redundant Ownership Functionality	Unresolved
	PSU	Potential Subtraction Underflow	Unresolved
	L02	State Variables could be Declared Constant	Unresolved
	L19	Stable Compiler Version	Unresolved

Table of Contents

Risk Classification

The criticality of findings in Cyberscope's smart contract audits is determined by evaluating multiple variables. The two primary variables are:

1. **Likelihood of Exploitation**: This considers how easily an attack can be executed, including the economic feasibility for an attacker.
2. **Impact of Exploitation**: This assesses the potential consequences of an attack, particularly in terms of the loss of funds or disruption to the contract's functionality.

Based on these variables, findings are categorized into the following severity levels:

1. **Critical**: Indicates a vulnerability that is both highly likely to be exploited and can result in significant fund loss or severe disruption. Immediate action is required to address these issues.
2. **Medium**: Refers to vulnerabilities that are either less likely to be exploited or would have a moderate impact if exploited. These issues should be addressed in due course to ensure overall contract security.
3. **Minor**: Involves vulnerabilities that are unlikely to be exploited and would have a minor impact. These findings should still be considered for resolution to maintain best practices in security.
4. **Informative**: Points out potential improvements or informational notes that do not pose an immediate risk. Addressing these can enhance the overall quality and robustness of the contract.

Severity	Likelihood / Impact of Exploitation
Critical	Highly Likely / High Impact
Medium	Less Likely / High Impact or Highly Likely/ Lower Impact
Minor / Informative	Unlikely / Low to no Impact

Review

Contract Name	Nova
Compiler Version	v0.8.27+commit.40a35a09
Optimization	200 runs
Explorer	https://bscscan.com/address/0x2fc1493bcc38da9921ac749899826289a146a1fb
Address	0x2fc1493bcc38da9921ac749899826289a146a1fb
Network	BSC
Symbol	NOVA
Decimals	18
Total Supply	210.000

Audit Updates

Initial Audit	08 Dec 2025

Source Files

Filename	SHA256
Nova.sol	120d8ee77930bc81f14daf457fced9df6317145b489d96d24a62b68081b1909d

Findings Breakdown

Critical	0
Medium	0
Minor / Informative	4

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	0	0	0
Medium	0	0	0	0
Minor / Informative	4	0	0	0

ROF - Redundant Ownership Functionality

Criticality	Minor / Informative
Location	Nova.sol#L30
Status	Unresolved

Description

The contract defines a `tokenOwner` variable, which is typically used in token contracts to restrict access to administrative or configuration functions. However, this contract does not include any owner-specific or privileged functionality that relies on this variable. As a result, tokenOwner is never used in practice and serves no functional purpose, making it redundant.

```
Shell
address private tokenOwner;
```

Recommendation

The team is advised to remove the ownable functionality from the contract to enhance code optimization and readability.

PSU - Potential Subtraction Underflow

Criticality	Minor / Informative
Location	Nova.sol#L66
Status	Unresolved

Description

The contract subtracts two values, the second value may be greater than the first value. As a result, the subtraction may underflow and cause the execution to revert.

Specifically the `transferFrom` function calls the `grantApproval` with the `value` argument equal to `spendingAllowances[sender][msg.sender]` minus the transferred value. The function does not ensure that the allowances of the spender are equal to or greater than the transferred value. This may result in the following two outcomes:

- In earlier versions of Solidity, the allowance will underflow and wrap around to a very large number, effectively granting the spender an unlimited allowance. As a result, the spender may be able to transfer significantly more tokens than initially approved, potentially draining the owner's balance.
- In Solidity >= 0.8.0, arithmetic underflow triggers a panic and reverts the transaction. In this case, any call to `transferFrom` where `spendingAllowances[sender][msg.sender] < value` will revert, but not because of an explicit and intentional allowance check. This breaks the expected ERC-20 semantics and may cause unexpected reverts for integrators relying on standard `transferFrom` behavior.

```
Shell
pragma solidity >0.4.0 <= 0.9.0;
...
function transferFrom(address sender, address
receiver, uint256 value) external override returns
```

```
(bool) {
    executeTransfer(sender, receiver, value);
    grantApproval(sender, msg.sender,
spendingAllowances[sender][msg.sender] - value);
    return true;
}
```

Recommendation

The team is advised to properly handle the code to avoid underflow subtractions and ensure the reliability and safety of the contract. The contract should ensure that the first value is always greater than the second value. It should add a sanity check in the setters of the variable or not allow executing the corresponding section if the condition is violated.

L02 - State Variables could be Declared Constant

Criticality	Minor / Informative
Location	Nova.sol#L26,27,28
Status	Unresolved

Description

State variables can be declared as constant using the constant keyword. This means that the value of the state variable cannot be changed after it has been set. Additionally, the constant variables decrease gas consumption of the corresponding transaction.

```shell
string public name = "Nova"
string public symbol = "NOVA"

uint8 public decimals = 18
```

Recommendation

Constant state variables can be useful when the contract wants to ensure that the value of a state variable cannot be changed by any function in the contract. This can be useful for storing values that are important to the contract's behavior, such as the contract's address or the maximum number of times a certain function can be called. The team is advised to add the constant keyword to state variables that never change.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	Nova.sol#L6
Status	Unresolved

Description

The `^` symbol indicates that any version of Solidity that is compatible with the specified version (i.e., any version that is a higher minor or patch version) can be used to compile the contract. The version lock is a mechanism that allows the author to specify a minimum version of the Solidity compiler that must be used to compile the contract code. This is useful because it ensures that the contract will be compiled using a version of the compiler that is known to be compatible with the code.

```
Shell
pragma solidity >0.4.0 <= 0.9.0;
```

Recommendation

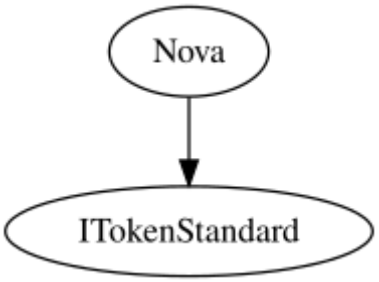
The team is advised to lock the pragma to ensure the stability of the codebase. The locked pragma version ensures that the contract will not be deployed with an unexpected version. An unexpected version may produce vulnerabilities and undiscovered bugs. The compiler should be configured to the lowest version that provides all the required functionality for the codebase. As a result, the project will be compiled in a well-tested LTS (Long Term Support) environment.

Functions Analysis

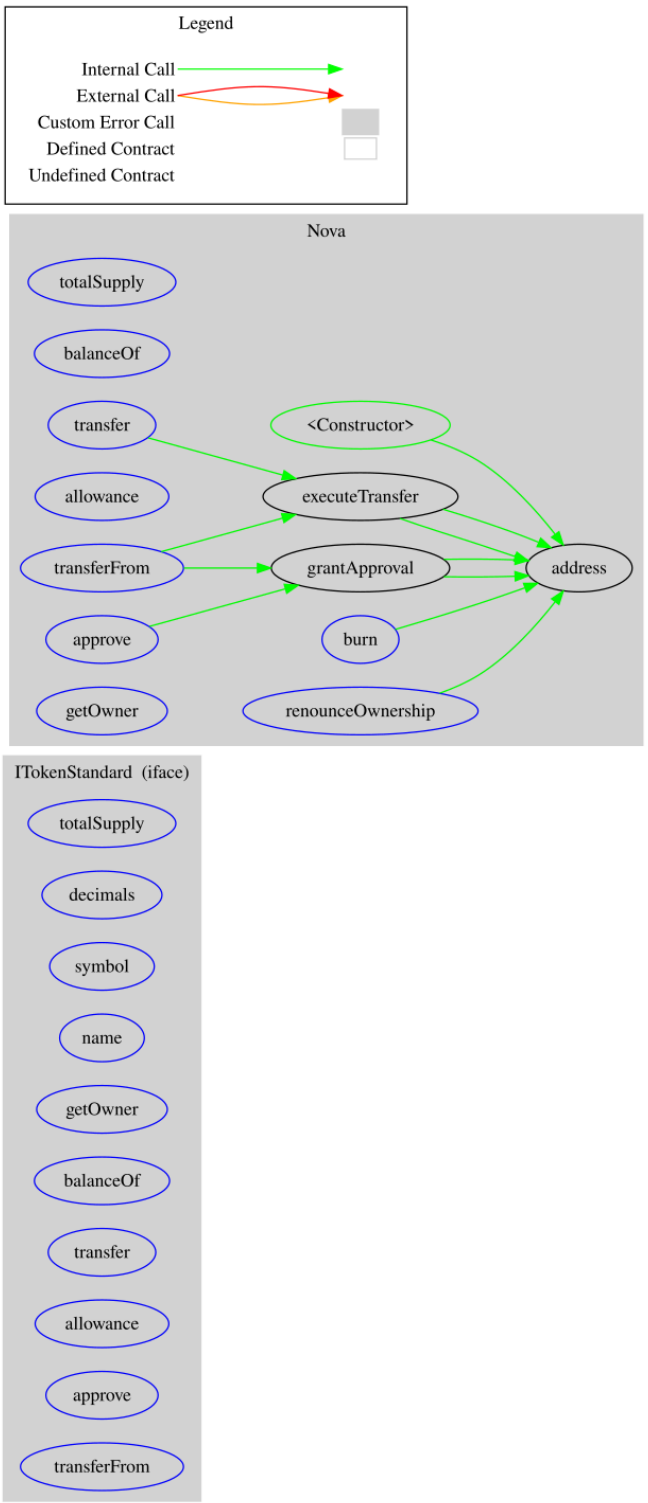
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
ITokenStandard	Interface			
	totalSupply	External		-
	decimals	External		-
	symbol	External		-
	name	External		-
	getOwner	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
Nova	Implementation	ITokenStandard		
		Public	✓	-
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-

	transferFrom	External	✓	-
	executeTransfer	Private	✓	
	grantApproval	Private	✓	
	getOwner	External		-
	burn	External	✓	-
	renounceOwnership	External	✓	-

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Custom Error Call
Defined Contract
Undefined Contract
Nova
totalSupply
balanceOf
transfer
allowance
transferFrom
approve
getOwner
<Constructor>
executeTransfer
grantApproval
burn
renounceOwnership
address
ITokenStandard (iface)
totalSupply
decimals
symbol
name
getOwner
balanceOf
transfer
allowance
approve
transferFrom

Summary

NOVA AI contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. NOVA AI is an interesting project that has a friendly and growing community. The Smart Contract analysis reported no compiler error or critical issues.

Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Cyberscope's prior written consent. This report is not nor should be considered an "endorsement" or "disapproval" of any particular project or team. This report is not nor should be regarded as an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Cyberscope to perform a security assessment. This document does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors' business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk Cyberscope's position is that each company and individual are responsible for their own due diligence and continuous security Cyberscope's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. You agree that your access and/or use including but not limited to any services reports and materials will be at your sole risk on an as-is where-is and as-available basis Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

About Cyberscope

Cyberscope is a TAC blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.

A **TAC Security** Company

The Cyberscope team

cyberscope.io