Cyberscope

Audit Report

AAMTOKEN

April 2025

Network BSC

Address 0x96bbb7878005a279ab8d191e503f211eff0f2330

Audited by © cyberscope

Analysis

Critical Medium Minor / Informative Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Unresolved
●	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
●	NCEA	Non-Compatible ERC20 Allowance	Unresolved
●	CO	Code Optimization	Unresolved
●	CCR	Contract Centralization Risk	Unresolved
●	MC	Missing Check	Unresolved
●	MEM	Missing Error Messages	Unresolved
●	MRF	Missing Renounce Functionality	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L06	Missing Events Access Control	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	AAMTOKEN
Compiler Version	v0.8.28+commit.7893614a
Optimization	200 runs
Explorer	https://bscscan.com/address/0x96bbb7878005a279ab8d191e503f211eff0f2330
Address	0x96bbb7878005a279ab8d191e503f211eff0f2330
Network	BSC
Symbol	AMTN
Decimals	18
Total Supply	3.000.000.000
Badge Eligibility	Must Fix Criticals

Audit Updates

Initial Audit	14 Apr 2025

Source Files

Filename	SHA256
contracts/AAMToken.sol	347561e5fabea6909e2c719a96c136b3bf78de11cf92f58e680908908ce803dc

Findings Breakdown

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	1	0	0	0
Medium	0	0	0	0
Minor / Informative	8	0	0	0

ST - Stops Transactions

Criticality	Critical
Location	contracts/AAMToken.sol#L50,70,116,138,201
Status	Unresolved

Description

The contract owner has the authority to stop the sales for all users. The owner may take advantage of it by setting `_paused` to true. As a result, the contract may operate as a honeypot.

```
modifier whenNotPaused() {
    require(!_paused, "Paused");
    _;
}

function pauseContract() external onlyOwner {
    _paused = true;
    emit Paused(msg.sender);
}

function batchTransfer(
    address[] calldata recipients,
    uint256[] calldata amounts
) external whenNotPaused returns (bool)

function transferFrom(
    address sender,
    address recipient,
    uint256 amount
) external override whenNotPaused returns (bool)

function transfer(
    address recipient,
    uint256 amount
) external override whenNotPaused returns (bool)
```

Recommendation

The team should carefully manage the private keys of the owner’s account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions.

Temporary Solutions:

These measurements do not decrease the severity of the finding

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-signature wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.

Permanent Solution:

- Renouncing the ownership, which will eliminate the threats but it is non-reversible.

NCEA - Non-Compatible ERC20 Allowance

Criticality	Minor / Informative
Location	contracts/AAMToken.sol#L112,120
Status	Unresolved

Description

The `transferFrom` method has a conditional statement that checks if the `msg.sender` is the actual `sender` of the tokens. If it is, the allowance is not being updated. While this is technically correct, it is possible that other decentralized platforms update their own local state according to the standards specifications. In that case the functionality may not work as intended as the external platform's state will be different from the actual one.

```
function transferFrom( address sender, address recipient,
uint256 amount) external override whenNotPaused returns (bool)
{
    //...
    if (msg.sender != sender) {
        uint256 allowed = _allowances[sender][msg.sender];
        require(allowed >= amount, "Allowance exceeded");
        unchecked {
            _allowances[sender][msg.sender] = allowed - amount;
        }
    }
    //...
}
```

Recommendation

The team is advised to update the allowance regardless of who the sender is. This will ensure that the contract is compatible across all platforms that utilize ERC20 standards.

CO - Code Optimization

Criticality	Minor / Informative
Location	contracts/AAMToken.sol#L138
Status	Unresolved

Description

There are code segments that could be optimized. A segment may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer operations.

Specifically, `batchTransfer` uses the length of `recipients` and `amounts` to perform two `for` loops. Since the length of the arrays must be equal, a local variable could be used to store the length of one of the two arrays and use that for the rest of the function.

```
function batchTransfer(
    address[] calldata recipients,
    uint256[] calldata amounts
) external whenNotPaused returns (bool) {
    require(recipients.length == amounts.length, "Mismatched
arrays");
    uint256 totalAmount = 0;
    for (uint256 i = 0; i < amounts.length; i++) {
        totalAmount += amounts[i];
    }
    require(_balances[msg.sender] >= totalAmount, "Insufficient
balance");
    _balances[msg.sender] -= totalAmount;
    for (uint256 i = 0; i < recipients.length; i++) {
        _balances[recipients[i]] += amounts[i];
        emit Transfer(msg.sender, recipients[i], amounts[i]);
    }
    return true;
}
```

Recommendation

The team is advised to take these segments into consideration and rewrite them so the runtime will be more performant. That way it will improve the efficiency and performance of the source code and reduce the cost of executing it.

CCR - Contract Centralization Risk

Criticality	Minor / Informative
Location	contracts/AAMToken.sol#L196,201,206
Status	Unresolved

Description

The contract's functionality and behavior are heavily dependent on external parameters or configurations. While external configuration can offer flexibility, it also poses several centralization risks that warrant attention. Centralization risks arising from the dependence on external configuration include Single Point of Control, Vulnerability to Attacks, Operational Delays, Trust Dependencies, and Decentralization Erosion.

```
function transferOwnership(address newOwner) external onlyOwner
function pauseContract() external onlyOwner
function unpauseContract() external onlyOwner
```

Recommendation

To address this finding and mitigate centralization risks, it is recommended to evaluate the feasibility of migrating critical configurations and functionality into the contract's codebase itself. This approach would reduce external dependencies and enhance the contract's self-sufficiency. It is essential to carefully weigh the trade-offs between external configuration flexibility and the risks associated with centralization.

MC - Missing Check

Criticality	Minor / Informative
Location	contracts/AAMToken.sol#L138,161
Status	Unresolved

Description

The contract is processing variables that have not been properly sanitized and checked that they form the proper shape. These variables may produce vulnerability issues.

Specifically, a check is missing in `batchTransfer` to ensure that recipients are not address zero.

```
function batchTransfer(
    address[] calldata recipients,
    uint256[] calldata amounts
) external whenNotPaused returns (bool)
```

In the `_transfer` method a check is missing to ensure that the amount to be sent is equal or smaller than the balance of the sender.

```
function _transfer(
    address sender,
    address recipient,
    uint256 amount
) internal {
    require(
        sender != address(0) && recipient != address(0),
        "Invalid address"
    );

    _balances[sender] -= amount;
    _balances[recipient] += amount;

    emit Transfer(sender, recipient, amount);
}
```

Recommendation

The team is advised to properly check the variables according to the required specifications.

MEM - Missing Error Messages

Criticality	Minor / Informative
Location	contracts/AAMToken.sol#L161
Status	Unresolved

Description

The contract is missing error messages. Specifically, there are no error messages to accurately reflect the problem, making it difficult to identify and fix the issue. As a result, the users will not be able to find the root cause of the error.

In the `_transfer` function a check and error message is missing to ensure that the balance of the sender is greater than or equal to the amount to be transferred.

```
function _transfer(
    address sender,
    address recipient,
    uint256 amount
) internal {
    require(
        sender != address(0) && recipient != address(0),
        "Invalid address"
    );
    _balances[sender] -= amount;
    _balances[recipient] += amount;
    emit Transfer(sender, recipient, amount);
}
```

Recommendation

The team is suggested to provide a descriptive message to the errors. This message can be used to provide additional context about the error that occurred or to explain why the contract execution was halted. This can be useful for debugging and for providing more information to users that interact with the contract.

MRF - Missing Renounce Functionality

Criticality	Minor / Informative
Location	contracts/AAMToken.sol
Status	Unresolved

Description

When initialized the contract sets the `msg.sender` as the contract's `owner`. However the contract does not implement any functionality to renounce the ownership of the contract.

```
address public owner;

modifier onlyOwner() {
    require(msg.sender == owner, "Only owner");
    _;
}
```

Recommendation

While it is technically possible to transfer ownership to a contract wallet address that has zero functionality, simulating renouncing the ownership, it is recommended to add the necessary functionality for transferring it to address zero.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	contracts/AAMToken.sol#L30,31,32
Status	Unresolved

Description

The Solidity style guide is a set of guidelines for writing clean and consistent Solidity code. Adhering to a style guide can help improve the readability and maintainability of the Solidity code, making it easier for others to understand and work with.

The followings are a few key points from the Solidity style guide:

1. Use camelCase for function and variable names, with the first letter in lowercase (e.g., myVariable, updateCounter).
2. Use PascalCase for contract, struct, and enum names, with the first letter in uppercase (e.g., MyContract, UserStruct, ErrorEnum).
3. Use uppercase for constant variables and enums (e.g., MAX_VALUE, ERROR_CODE).
4. Use indentation to improve readability and structure.
5. Use spaces between operators and after commas.
6. Use comments to explain the purpose and behavior of the code.
7. Keep lines short (around 120 characters) to improve readability.

```
string private constant _name = "AAMTOKEN"
string private constant _symbol = "AMTN"
uint8 private constant _decimals = 18
```

Recommendation

By following the Solidity naming convention guidelines, the codebase increased the readability, maintainability, and makes it easier to work with.
Find more information on the Solidity documentation
https://docs.soliditylang.org/en/stable/style-guide.html#naming-conventions.

L06 - Missing Events Access Control

Criticality	Minor / Informative
Location	contracts/AAMToken.sol#L198
Status	Unresolved

Description

Events are a way to record and log information about changes or actions that occur within a contract. They are often used to notify external parties or clients about events that have occurred within the contract, such as the transfer of tokens or the completion of a task. There are functions that have no event emitted, so it is difficult to track off-chain changes.

```
owner = newOwner
```

Recommendation

To avoid this issue, it's important to carefully design and implement the events in a contract, and to ensure that all required events are included. It's also a good idea to test the contract to ensure that all events are being properly triggered and logged.

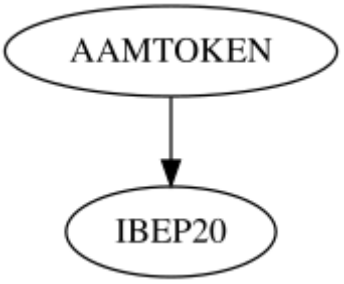
By including all required events in the contract and thoroughly testing the contract's functionality, the contract ensures that it performs as intended and does not have any missing events that could cause issues.

Functions Analysis

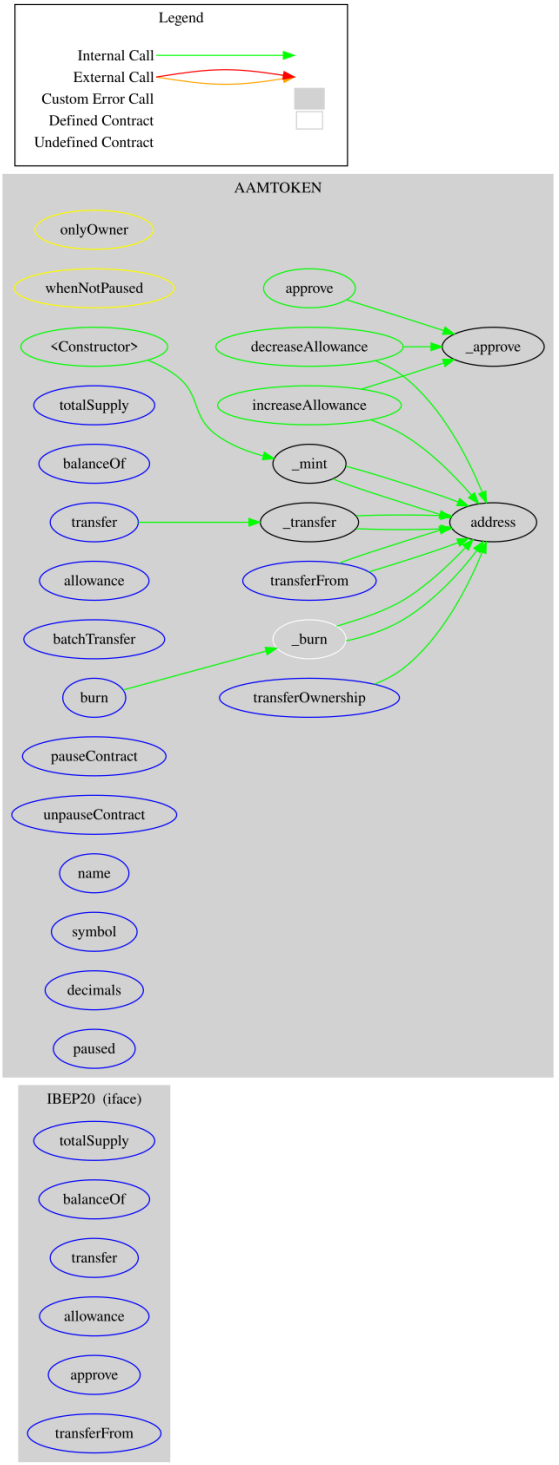
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
AAMTOKEN	Implementation	IBEP20		
		Public	✓	-
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	whenNotPaused
	allowance	External		-
	approve	Public	✓	-
	_approve	Internal	✓	
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	transferFrom	External	✓	whenNotPaused
	batchTransfer	External	✓	whenNotPaused
	_transfer	Internal	✓	
	_mint	Internal	✓	
	_burn	Internal	✓	
	burn	External	✓	whenNotPaused
	transferOwnership	External	✓	onlyOwner
	pauseContract	External	✓	onlyOwner
	unpauseContract	External	✓	onlyOwner

	name	External		-
	symbol	External		-
	decimals	External		-
	paused	External		-

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Custom Error Call
Defined Contract
Undefined Contract
AAMTOKEN
onlyOwner
whenNotPaused
<Constructor>
totalSupply
balanceOf
transfer
allowance
batchTransfer
burn
pauseContract
unpauseContract
name
symbol
decimals
paused
approve
decreaseAllowance
increaseAllowance
_mint
_transfer
transferFrom
_burn
transferOwnership
_approve
address
IBEP20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom

Summary

AAMTOKEN contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. There are some functions that can be abused by the owner like stop transactions. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract or renouncing ownership will eliminate all the contract threats.

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 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.

The Cyberscope team

cyberscope.io