Smart Contract Audit Report

MEMBO

1. Audit details

Name Token	MEMBO
Contract address	—
Contract URL	https://bscscan.com/address/0x618CddD99BB5246Ae373A5813FB8d9AF55263721#code
Language	Solidity
Platform	bscscan.com
Date	7 February 2025

General Overview

The **MemeBomber (MEMBO)** contract is an ERC20 token with burnable functionality and a fixed supply of10,000,000,000 tokens. It is implemented using OpenZeppelin's security-enhanced libraries, including:

- ERC20Burnable – Allows token holders to burn their own tokens.
- Ownable – Grants ownership control to a specific address.
- SafeERC20 – Ensures secure ERC20 token transfers.

The contract prevents further minting, includes token recovery mechanisms, and implements standard ERC20 features. However, this audit highlights potential security risks, optimizations, and best practices.

1. Security Risks Identified

1.1 Centralized Ownership Risks

- Issue: The entire token supply is minted to the deployer's address.
- Risk: The deployer has complete control over token distribution, making the system vulnerable to misuse or centralization risks.
- Recommendation: Consider implementing amulti-signature wallet orgovernance model to prevent single-point failure or misuse.

1.2 Unrestricted Token Transfers

- Issue: The contract does not impose any restrictions on token transfers.
- Risk: This makes the token vulnerable toflash loan attacks,whale manipulation, andrug pulls (if a large supply is controlled by a single entity).
- Recommendation: If needed, consider implementing:
  - Whitelists/Blacklists to control transfers.
  - Anti-bot mechanisms to prevent front-running attacks at launch.

1.3 Lack of Rate-Limiting or Anti-Bot Measures

- Issue: The contract does not have built-in protection against high-frequency trades.
- Risk: Bots can exploit launch events, leading to unfair token distribution.
- Recommendation: Implement:
  - Time-based restrictions (e.g., cooldowns between transfers).
  - Transaction size limits to prevent massive token dumps.

1.4 Recovery Mechanism Risks

- Issue: The `recoverERC20` function allows the owner to withdraw any ERC20 token mistakenly sent to the contract.
- Risk: If the contract is ever used for staking or locked liquidity, the owner could potentially withdraw critical funds.
- Recommendation: Introduce whitelisted recoverable tokens to prevent unintended fund withdrawals.

1.5 Burning Mechanism Considerations

- Issue: The contract allows anyone to burn their own tokens but does not provide an option to burn on behalf of another user.
- Risk: This may be a limitation in governance or deflationary models where authorized accounts might need to burn tokens.
- Recommendation: Add aburnFrom function with explicit approval logic.

2. Best Practices and Optimizations

2.1 Gas Optimization in `burnTokens`

Observation: The `burnTokens` function includes a redundant `require` statement:

```
require(balanceOf(msg.sender) >= amount, "MEMBO: insufficient
                              balance");
```

Optimization: The ERC20 `_burn` function already checks for sufficient balance, making this check unnecessary.

2.2 Explicit Use of Visibility Modifiers

Observation: Functions like `receive()` and `fallback()` are implicitly public.
Recommendation: Explicitly declare `external` visibility for clarity and security.

2.3 Missing Events for Key Transactions

Observation: The contract emits events for burning and recovery but does not emit an event when tokens are transferred.
Recommendation: Add acustom Transfer event for enhanced tracking.

2.4 Naming Convention Consistency

Observation: Function names use a mix of PascalCase (`TokensBurned`) and camelCase (`burnTokens`).
Recommendation: Standardize function names (e.g., `tokensBurned` instead of `TokensBurned`).

3. Feature Recommendations

3.1 Governance and Upgradeability

Suggestion: Consider integratingAccessControl instead of single-owner `Ownable` to allowrole-based management.

3.2 Emergency Pause Mechanism

Suggestion: Implement a Pausable feature to allow stopping transfers in case of emergency:

```
function transfer(address to, uint256 value) public whenNotPaused
                     returns (bool) { ... }
```

3.3 Timelock for Owner Actions

Suggestion: Implement a timelock for high-risk functions (e.g., token recovery), allowing the community to react before changes take effect

4. Severity Matrix

Centralized Ownership Risks

Severity: High
Likelihood: Medium
Impact: High
Priority: Critical

Unrestricted Token Transfers

Severity: Medium
Likelihood: High
Impact: Medium
Priority: High

Lack of Rate-Limiting/Anti-Bot Measures

Severity: Medium
Likelihood: High
Impact: Medium
Priority: High

Recovery Mechanism Risks

Severity: Low
Likelihood: Medium
Impact: Medium
Priority: Medium

Burning Mechanism Considerations

Severity: Low
Likelihood: Medium
Impact: Low
Priority: Medium

5. Conclusion

The MemeBomber (MEMBO) contract is a well-implemented ERC20 token that leverages OpenZeppelin's security libraries. While it effectively prevents minting beyond the initial supply and includes a recovery mechanism, it has some centralization risks, unrestricted transfers, and lack of anti-bot measures.

By implementing the recommended security improvements, such as governance enhancements, pausing mechanisms, and transaction limits, the contract can enhance its security, fairness, and long-term viability.

soken

SOKEN.IO

WEBSITE: WWW.SOKEN.IO

TELEGRAM: @SOKEN_SUPPORT

X (TWITTER): @SOKEN_TEAM