soken

Smart Contract Audit Report: Snakes Token (Blue Snakes)

soken

Token Name	Blue Snakes
Contract address	0x4345a1056E614E61C877eC1CB90510Afc8753123
Contract URL	https://bscscan.com/token/0x4345a1056E614E61C877eC1CB90510Afc8753123
Language	Solidity
Platform	BSC
Date	9 May 2025

1. Summary

The `SnakesToken` contract is an ERC20 token with the following features:

- Burnable tokens
- Permit (EIP-2612) for gasless approvals
- Votes (governance support)
- Ownable (admin control)
- Account freezing (owner can freeze/unfreeze accounts)

The contract uses OpenZeppelin libraries, which are industry-standard and well-audited.

2. Findings

2.1. Critical Severity

No critical issues found.

2.2. High Severity

No high-severity issues found.

2.3. Medium Severity

2.3.1. Centralization Risk: Owner Can Freeze/Unfreeze Accounts

• The freezeAccount and unfreezeAccount functions are gated by the onlyOwner modifier, which refers to a single address.

• According to audit, this address is a Gnosis Safe multisignature wallet, meaning multiple signers are required to perform sensitive actions such as freezing.

• However, this address has not been publicly verified as a Safe or annotated on BscScan.

• Recommendation: While the use of a multisig wallet helps mitigate centralization risk, we recommend the owner publicly verify the Safe to enhance transparency and user trust. At minimum, the project should clearly disclose this ownership structure in documentation or on its website.

2.4. Low Severity

2.4.1. Pausable Not Implemented

• The contract imports ERC20Pausable but does not use it. If pausing is intended, implement pause/unpause functions. If not, remove the unused import.

• **Recommendation:** Remove unused imports to reduce bytecode size and potential confusion.

2.4.2. Initial Mint to msg.sender

• The constructor mints the entire supply to msg.sender, not initialOwner. This may be intentional, but could be surprising if the deployer is not the intended owner.

• **Recommendation:** Consider minting to initialOwner for clarity and security.

2.5. Informational

2.5.1. NatSpec Comments

• The contract lacks NatSpec documentation for functions and parameters.

• **Recommendation:** Add NatSpec comments for better code clarity and automated documentation.

2.5.2. Event Emission on State Change

• The contract emits events for freezing/unfreezing accounts, which is good practice.

2.5.3. Solidity Version

• Uses a recent Solidity version (0.8.27), which is good for security.

3. Best Practice Suggestions

• Upgradeability: If future upgrades are planned, consider implementing a proxy pattern (e.g., UUPS or Transparent Proxy).
• Testing: Ensure comprehensive unit and integration tests, especially for freeze/unfreeze logic and multisig interaction (e.g., simulate multi-party confirmations).
• Access Control Transparency: Although ownership is managed via a multisignature wallet, consider publicly verifying the Gnosis Safe address or at minimum disclosing signer and threshold configuration in official documentation to improve transparency and user trust.
• Code Hygiene: Remove unused imports such as ERC20Pausable if pause functionality is not intended, to reduce contract size and improve clarity.

4. Vulnerability Checklist

Vulnerability	Status
Reentrancy	Not Found
Integer Overflow	Not Found (>=0.8.0)
Access Control	Gated by onlyOwner; ownership is a multisig wallet (not publicly verified)
Denial of Service	Not found — no unbounded loops or gas griefing patterns
Gas Optimization	No major issues
Unused Imports	ERC20 Pausable unused
Upgradability	Not implemented
Event Emission	Proper for freeze/unfreeze
Owner Can Freeze Accounts	True – mitigated by multisig, but lacks public verification

5. Conclusion

The SnakesToken contract is well-structured and leverages OpenZeppelin's secure libraries. The main risk identified is the centralization of the freeze function, which is gated by an onlyOwner modifier. According to the audit, this role is assigned to a Gnosis Safe multisig wallet, which improves governance security. However, the ownership structure has not been publicly verified. Addressing this transparency gap and resolving low-severity issues will further strengthen trust and project credibility.

6. OpenZeppelin Contracts (Dependencies)

Summary:

All OpenZeppelin contracts used (Ownable, Votes, ERC20, etc.) are unmodified and imported as dependencies. These contracts are widely used, well-audited, and considered secure by the blockchain community.

Findings:

• No vulnerabilities found in OpenZeppelin contracts themselves.

• Integration Risks:

• If custom logic overrides parent functions incorrectly, it could introduce vulnerabilities. In SnakesToken, the only override is _update, which is handled correctly with additional frozen account checks.

Best Practices:

• Always use the latest stable version of OpenZeppelin contracts.

• Avoid modifying OpenZeppelin code directly; extend via inheritance as done here.

7. General Observations

• Single Custom Contract: Only SnakesToken contains custom logic.
• No Proxy/Upgradability: No upgradability pattern is used.
• No External Calls: No external contract calls in custom logic, reducing reentrancy risk.
• No Ether Handling: No payable functions or Ether transfers, reducing attack surface.

8. Final Recommendations

• Centralization Risk: Owner can freeze/unfreeze accounts. Consider DAO/multisig for production.
• Remove Unused Imports: ERC20Pausable is imported but not used.
• Minting Logic: Consider minting to initialOwner instead of msg.sender for clarity.
• Documentation: Add NatSpec comments for all public/external functions.
• Testing: Ensure all custom logic (especially freezing) is thoroughly tested.

9. Conclusion

The SnakesToken contract is well-structured and built on OpenZeppelin's secure and audited libraries. No critical or high-severity vulnerabilities were identified. The main point of attention is the centralized power of the freeze/unfreeze function, which is protected by an onlyOwner modifier.

According to the audit, this owner address is a Gnosis Safe multisignature wallet, which adds a layer of governance protection. However, this multisig setup has not been publicly verified. To strengthen trust and transparency for token holders and third parties, we recommend publicly disclosing or verifying the Safe address and its signer configuration.

Low-severity issues, such as unused imports and clarity in initial minting logic, were noted and can be addressed to further improve code quality. Overall, the contract meets common security standards for ERC20 implementations, and no logic-breaking flaws were found.

10. DISCLAIMER

By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree with these terms, please stop reading the report and delete any copies you have downloaded and/or printed.

This report is provided for informational purposes only and does not constitute investment advice. No one may rely on this report or its contents, and Soken and its affiliates accept no responsibility to you or any third party. Soken makes no guarantees or representations regarding the accuracy, completeness, or reliability of this report.

This report is provided “as is,” without any express or implied warranties, conditions, or representations, including (but not limited to) warranties implied by law regarding quality, fitness for a particular purpose, or the use of reasonable care and diligence.

To the extent permitted by law, Soken disclaims all liability and shall not be responsible for any losses or damages of any kind (including but not limited to direct, indirect, special, punitive, incidental, or economic losses) that may arise for you or third parties as a result of using or interpreting the information in this report.

Security analysis is conducted solely on smart contracts. The audit does not include a review of platform operations, applications, business logic, or user interface. Product source code has not been reviewed.

DYOR (Do Your Own Research) – Before making any decisions related to this project, it is recommended to conduct your own analysis.

soken

x.com/soken_team

t.me/soken_support

website: soken.io

info@soken.io