KODE

Security Assessment

HealingSpaces

Verified on 28/08/2025

✓ SECURITY VERIFIED

v.0.0.1

Audit Report for HSPC Token Contract

Date:	August 28, 2025
Contract Name:	HSPC (HealingSpaces)
Website:	*www.healingspaces.io*
Contract Address:	0x9fa39ab07776c4f995f0e96707d6ec25350c5b67
Blockchain:	Binance Smart Chain
Solidity Version:	0.8.30

Status: ✅ PASSED!

Key Findings

Critical	Medium	Low
0	0	1

Security Score 5/5

● Centralization Risk HSPC-001

80% supply controlled by owner via reserve unlock mechanism poses centralization risk.

✓ Acknowledged

Audit Scope

The security assessment was focused on identifying vulnerabilities in the smart contract code that could potentially lead to:

- Fund loss or token theft
- Unauthorized minting of tokens
- Front running attacks
- Logic errors in reserve calculations
- Reentrancy vulnerabilities
- Access control weaknesses
- Owner privilege abuse

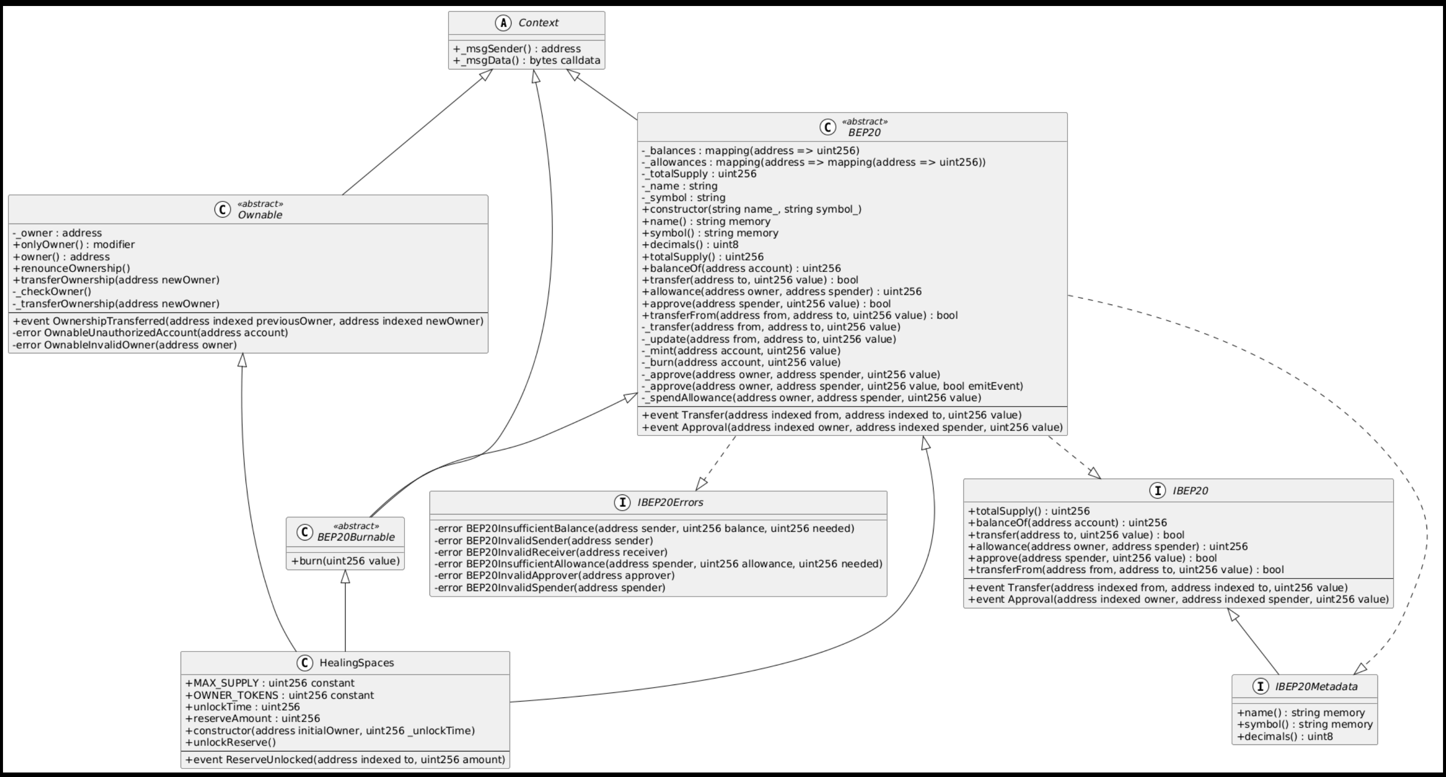
Audit Findings

✓ **No Critical Issues**
No critical security vulnerabilities found

✓ **No Medium Risk Issues**
No medium severity issues found

! **1 Low Risk Issue**
Centralization risk from owner control

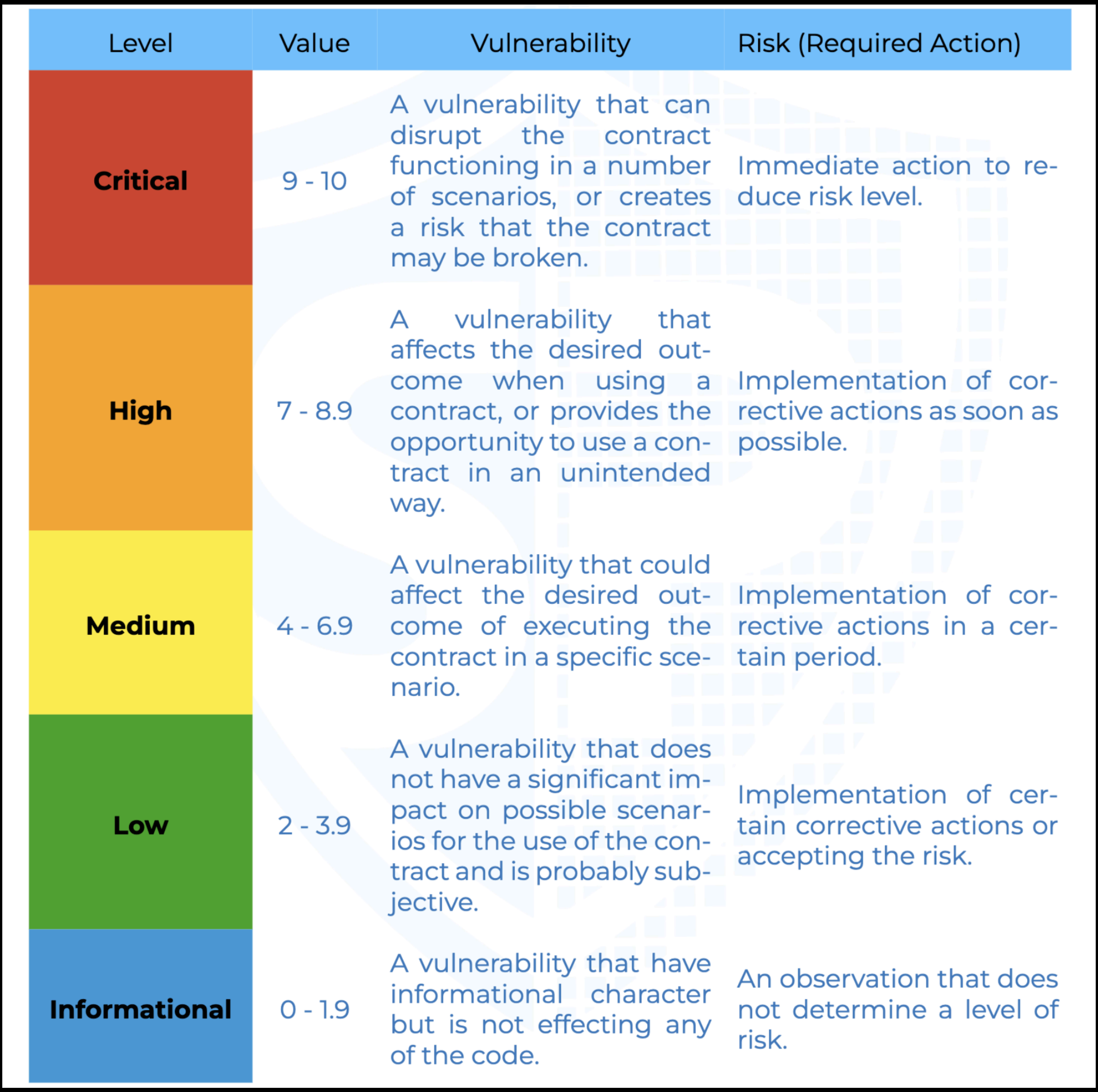
The contract shows clean Solidity code with proper BEP-20 compliance and security measures.



A Context
+_msgSender() : address
+_msgData() : bytes calldata
C «abstract» Ownable
-_owner : address
+onlyOwner() : modifier
+owner() : address
+renounceOwnership()
+transferOwnership(address newOwner)
-_checkOwner()
-_transferOwnership(address newOwner)
+event OwnershipTransferred(address indexed previousOwner, address indexed newOwner)
-error OwnableUnauthorizedAccount(address account)
-error OwnableInvalidOwner(address owner)
C «abstract» BEP20
-_balances : mapping(address => uint256)
-_allowances : mapping(address => mapping(address => uint256))
-_totalSupply : uint256
-_name : string
-_symbol : string
+constructor(string name_, string symbol_)
+name() : string memory
+symbol() : string memory
+decimals() : uint8
+totalSupply() : uint256
+balanceOf(address account) : uint256
+transfer(address to, uint256 value) : bool
+allowance(address owner, address spender) : uint256
+approve(address spender, uint256 value) : bool
+transferFrom(address from, address to, uint256 value) : bool
-_transfer(address from, address to, uint256 value)
-_update(address from, address to, uint256 value)
-_mint(address account, uint256 value)
-_burn(address account, uint256 value)
-_approve(address owner, address spender, uint256 value)
-_approve(address owner, address spender, uint256 value, bool emitEvent)
-_spendAllowance(address owner, address spender, uint256 value)
+event Transfer(address indexed from, address indexed to, uint256 value)
+event Approval(address indexed owner, address indexed spender, uint256 value)
C «abstract» BEP20Burnable
+burn(uint256 value)
I IBEP20Errors
-error BEP20InsufficientBalance(address sender, uint256 balance, uint256 needed)
-error BEP20InvalidSender(address sender)
-error BEP20InvalidReceiver(address receiver)
-error BEP20InsufficientAllowance(address spender, uint256 allowance, uint256 needed)
-error BEP20InvalidApprover(address approver)
-error BEP20InvalidSpender(address spender)
I IBEP20
+totalSupply() : uint256
+balanceOf(address account) : uint256
+transfer(address to, uint256 value) : bool
+allowance(address owner, address spender) : uint256
+approve(address spender, uint256 value) : bool
+transferFrom(address from, address to, uint256 value) : bool
+event Transfer(address indexed from, address indexed to, uint256 value)
+event Approval(address indexed owner, address indexed spender, uint256 value)
C HealingSpaces
+MAX_SUPPLY : uint256 constant
+OWNER_TOKENS : uint256 constant
+unlockTime : uint256
+reserveAmount : uint256
+constructor(address initialOwner, uint256 _unlockTime)
+unlockReserve()
+event ReserveUnlocked(address indexed to, uint256 amount)
I IBEP20Metadata
+name() : string memory
+symbol() : string memory
+decimals() : uint8

1. Introduction

This audit evaluates the HealingSpaces (HSPC) token contract, a BEP-20 compliant token with ownership features, time locked reserves, and burnable functionality. The contract is built upon OpenZeppelin inspired BEP20, Ownable, and Burnable implementations, leveraging Solidity 0.8.30's built in safety checks. The purpose of this report is to assess the correctness and security of the contract, verify its compliance with the BEP-20 standard, identify potential vulnerabilities, and suggest improvements to enhance its security, usability, and governance.



Level	Value	Vulnerability	Risk (Required Action)
Critical	9 - 10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
High	7 - 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon as possible.
Medium	4 - 6.9	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.	Implementation of corrective actions in a certain period.
Low	2 - 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	0 - 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk.

2. Contract Overview

The HSPC contract leverages OpenZeppelin's token libraries. Below are the key components:

2.1. Inheritance and Dependencies

- **BEP20**: Implements BEP-20 standard token functionality (balances, transfers, approvals).
- **BEP20Burnable:** Allows holders to burn their own tokens.
- **Ownable**: Provides access control for sensitive functions (onlyOwner).
- **Solidity Version**: 0.8.30, benefiting from overflow checks and gas optimizations.

2.2. Token Specifications

- **Name**: "HealingSpaces"
- **Symbol**: "HSPC"
- **Decimals**: 18 (standard for BEP-20 tokens, allowing precise fractional amounts).
- **Maximum Supply:** 10,000,000 HSPC (10M)
- **Initial Supply**: 2,000,000 HSPC tokens (2 Million), minted to the deployer's address during deployment.
- **Reserve Allocation:** 8,000,000 HSPC (8 Million) locked until unlockTime.

2.3. Ownership Features

- **Initial Owner**: Deployment address provided in constructor.
- **Transferable**: Ownership can be transferred to another address via `transferOwnership(address newOwner)`.
- **Renounceable**: Ownership can be renounced via `renounceOwnership()`, setting the owner to `address(0)`.

2.4. Reserve Mechanism

- **Unlock Time:** Configurable timestamp set at deployment.
-
- **Unlock Function:** unlockReserve() allows the owner to mint the remaining 8 Million tokens only after unlock time has passed.
-
- **One Time Mint:** Once reserve is minted, reserveAmount is set to 0 and cannot be reused.

3. Technical Structure

3.1. BEP20 Implementation

The contract fully adheres to the BEP-20 standard and includes additional features for improved usability:

State Variables

- `_balances: mapping(address => uint256)` - Tracks token balances for each address.
- `_allowances: mapping(address => mapping(address => uint256))` - Tracks spender allowances for each owner.
- `MAX_SUPPLY: uint256` - Stores the max token supply cap (10,000,000 * 10^18).
- `_totalSupply: uint256` - Stores the total token supply (2,000,000 * 10^18).
- `_name: string` - Immutable token name ("HealingSpaces").
- `_symbol: string` - Immutable token symbol ("HSOC").
- `_decimals: uint8` - Immutable decimals value (18).

Core Functions

- `totalSupply()`: Returns the total token supply.
- `balanceOf(address account)`: Returns the balance of a specified address.
- `transfer(address to, uint256 value)`: Transfers tokens from the caller to the specified address.
- `approve(address spender, uint256 value)`: Sets an allowance for a spender.
- `transferFrom(address from, address to, uint256 value)`: Transfers tokens on behalf of an owner using an allowance.
- `_mint(address account, uint256 value)`: Internal function to mint tokens (called only in the constructor).
- `_burn(address account, uint256 value)`: Internal function to burn tokens (not publicly exposed).

Events

- `Transfer(address indexed from, address indexed to, uint256 value)`: Emitted for transfers, mints, and burns.
- `Approval(address indexed owner, address indexed spender, uint256 value)`: Emitted when an allowance is updated.
- `ReserveUnlocked(address indexed to, uint256 amount)`: Emitted when reserve unlocked.

Error Handling

- Implements ERC-6093 (custom errors) for detailed and gas efficient error reporting:
  - `BEP20InsufficientBalance(address sender, uint256 balance, uint256 needed)`
  - `BEP20InvalidSender(address sender)`
  - `BEP20InvalidReceiver(address receiver)`
  - `BEP20InsufficientAllowance(address spender, uint256 allowance, uint256 needed)`

3.2. Ownable Implementation

The `Ownable` contract provides basic access control:

State Variables

- `_owner: address` - Stores the current owner's address.

Core Functions

- `owner()`: Returns the current owner's address.
- `transferOwnership(address newOwner)`: Transfers ownership to a new address (emits `OwnershipTransferred`).
- `renounceOwnership()`: Sets the owner to `address(0)` (emits `OwnershipTransferred`).

Modifiers

- `onlyOwner`: Restricts function access to the current owner.

Events

- `OwnershipTransferred(address indexed previousOwner, address indexed newOwner)`: Emitted on ownership changes.

3.3. HSPC Contract Details

- **Constructor**:
  - Sets token name, symbol, and decimals.
  - Mints 2M tokens to initialOwner.
  - Stores unlockTime and reserveAmount.
- **Reserve Unlocking**:
  - Function unlockReserve() allows the owner to mint the remaining 8M tokens only after unlockTime.
  - Emits ReserveUnlocked event.
- **Burn Functionality**:
  - Any holder can call burn(uint256 value) to reduce supply.

4. Security Analysis

4.1. Potential Vulnerabilities

Reentrancy

- **Risk**: Low
- **Reason**: No external calls are made in state changing functions like `transfer` or `transferFrom`, mitigating reentrancy risks.

Integer Overflow/Underflow

- **Risk**: Mitigated
- **Reason**: Solidity 0.8.30 includes automatic overflow checks. The contract also uses `unchecked` blocks in safe scenarios (e.g., after balance checks) to optimize gas.

Front Running (Allowance Race Condition)

- **Risk**: Medium
- **Reason**: The `approve` function is vulnerable to front running if an allowance is updated without resetting it to zero first. Users should follow the recommended practice of setting allowances to zero before updating them.

Denial of Service (DoS)

- **Risk**: Low
- **Reason**: No loops or unbounded operations exist, minimizing DoS risks.

Minting Control

- Risk: Controlled
- Only initial mint (2M) and one time reserve mint (8M). No further minting possible.

4.2. Gas Efficiency

- **Solidity 0.8.30**: Benefits from compiler level gas optimizations.
- **Unchecked Blocks**: Used in `_update` and `_spendAllowance` to avoid redundant overflow checks, reducing gas costs without compromising safety.
- **Minimalistic Design**: Fewer functions and no complex logic contribute to lower gas usage.

4.3. Ecosystem Compatibility

- Fully BEP-20 compliant.
- **DeFi Integration**: Fully BEP-20 compliant, ensuring compatibility with wallets, exchanges, and DeFi protocols.
- **Limitations**: Lack of features like `permit` (EIP-2612) or token recovery may restrict advanced use cases.

5. Key Observations

5.1. Supply Model

- Max Supply Fixed at 10M HSPC.
- Owner receives 2M immediately, while 8M is time locked.

5.2. Ownership Implications

- **Renouncement**: Renouncing ownership decentralizes the contract but eliminates future administrative control.
- **Single Owner**: No multi admin or role based access control is implemented, relying solely on the `onlyOwner` modifier.

5.3. Minimalistic Approach

- The contract avoids unnecessary complexity, reducing the attack surface but also limiting functionality (e.g., no pause mechanism or advanced approval features).

5.4. Flexibility

- Burn function provides supply reduction mechanism.
- No pausing or freezing mechanism.

6. Recommendations

6.1. Security Enhancements

- **Allowance Safety**: Add `increaseAllowance` and `decreaseAllowance` to mitigate front running risks in `approve`.
- **Timelock/Multi Sig**: Consider transferring ownership to a timelock or multi signature wallet for enhanced security.

6.2. Feature Additions

- **EIP-2612 (Permit)**: Implement `permit` for gasless approvals, improving DeFi compatibility.
- **Minting Flexibility**: Add an `onlyOwner` minting function for controlled supply adjustments if needed.
- **Pause Functionality**: Include an emergency pause mechanism to halt transfers during security incidents.
- **Token Recovery**: Add a function to recover tokens accidentally sent to the contract address.

6.3. Deployment Considerations

- **Ownership Plan**: Define a post deployment strategy (e.g., multi sig ownership or governance contract).
- **Testing**: Conduct thorough unit and integration tests, including edge cases and interactions with other contracts.
- Use multi signature wallet for owner operations and unlocking reserves.

6.4. AUDIT COMMENTS

Smart Contract audit comment for a non technical perspective:

🟢 Owner can renounce and transfer ownership

🟢 Owner cannot mint beyond max supply cap

🟢 Owner cannot burn user tokens

🟢 Owner cannot pause contract

🟢 Owner cannot block users

🔵 Centralization Risk HSPC-001 as 80% supply controlled by owner via reserve unlock mechanism poses centralization risk.

7. Conclusion

The HSPC Smart Contract is a secure, BEP-20 compliant token implementation built on strong OpenZeppelin style contracts, ensuring a fixed maximum supply of 10M tokens with a time locked reserve mechanism and optional supply reduction via burning. The contract demonstrates a high level of security with minimal attack surface; however, centralization risk exists as 80% of the supply is controlled by the owner through reserve unlocks. To further enhance resilience and governance, it is recommended to add features such as permit, pausing, and multi signature control, particularly for owner operations and reserve unlocking.

Final Audit Verdict: ✅ PASSED!

KODE

v.0.0.1

KODE