

DESSERT  
FINANCE



**SnowKing (SnowKing)**

Light Audit

Performed at block **15987545**

PERFORMED BY DESSERT FINANCE  
FOR CONTRACT ADDRESS: **0x7185Fc499Ee5486b290B5e90e33abd3BcC0aeBE6**

## INITIAL DISCLAIMER

Dessert Finance provides due-diligence project audits for various projects. Dessert Finance in no way guarantees that a project will not remove liquidity, sell off team supply, or otherwise exit scam.

Dessert Finance does the legwork and provides public information about the project in an easy-to-understand format for the common person.

Agreeing to an audit in no way guarantees that a team will not remove *all* liquidity (“Rug Pull”), remove liquidity slowly, sell off tokens, quit the project, or completely exit scam. There is also no way to prevent private sale holders from selling off their tokens. It is ultimately your responsibility to read through all documentation, social media posts, and contract code of each individual project to draw your own conclusions and set your own risk tolerance.

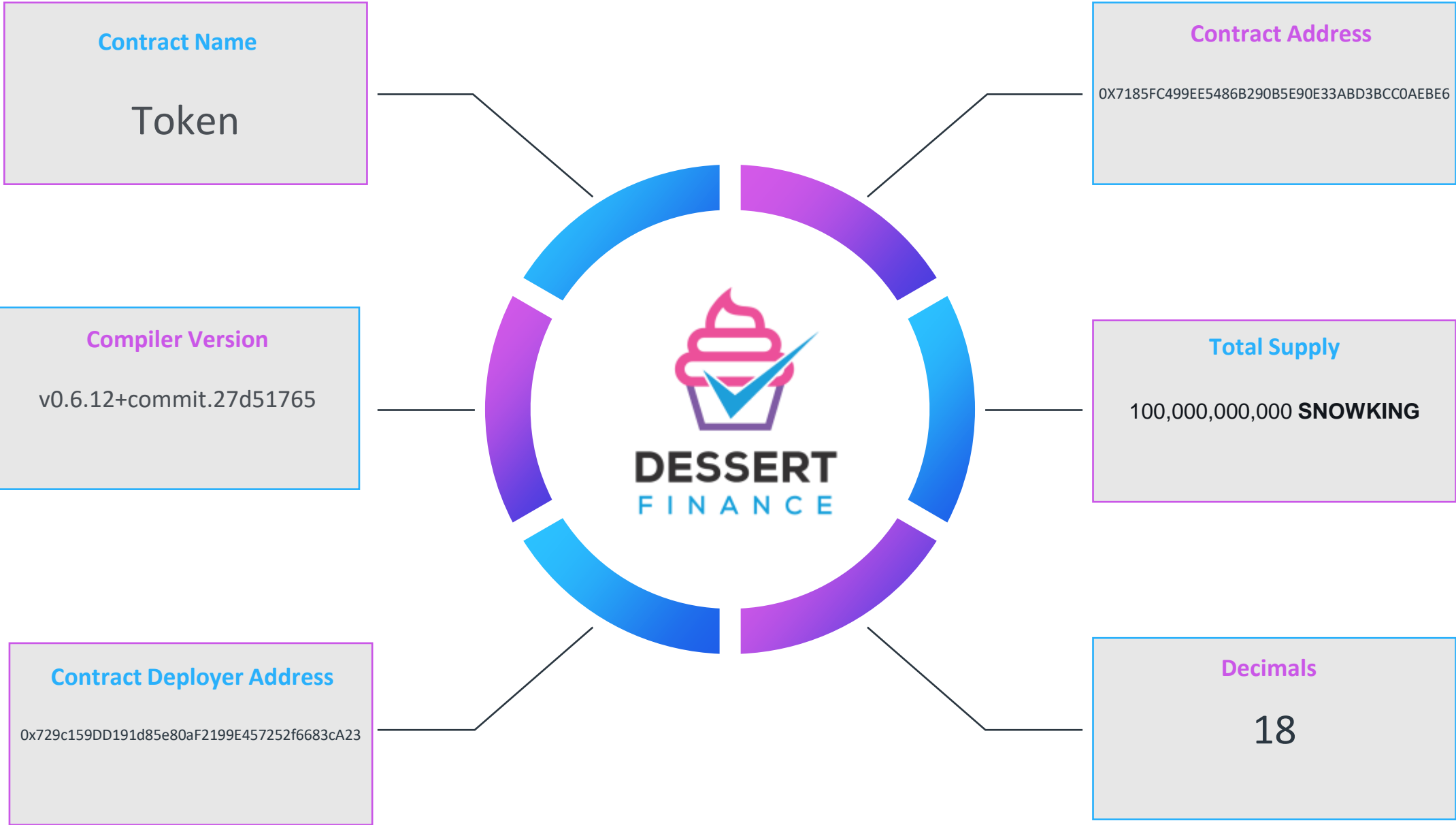
Dessert Finance in no way takes responsibility for any losses, nor does Dessert Finance encourage any speculative investments. The information provided in this audit is for information purposes only and should not be considered investment advice. Dessert Finance does not endorse, recommend, support, or suggest any projects that have been audited. An audit is an informational report based on our findings, We recommend you do your own research, we will never endorse any project to invest in.

# Table of Contents



1. Contract Code Audit – Token Overview
2. BEP-20 Contract Code Audit – Overview
3. BEP-20 Contract Code Audit – Vulnerabilities Checked
4. Disclaimers

# Contract Code Audit – Token Overview





# BEP-20 Contract Code Audit – Overview

Dessert Finance was commissioned to perform an audit on SnowKing (SnowKing)

```
pragma solidity ^0.8.0;

// Provides information about the current execution context, including the
// sender of the transaction and its data, block hashes, etc. While these are generally available
// as msg.sender and msg.data, they should not be accessed in such a direct
// manner, since when dealing with meta-transactions the account sending and
// paying for execution may not be the actual sender (as far as an application
// is concerned).

// This contract is only required for intermediate, library-like contracts.
abstract contract Context {
    function _msgSender() internal view virtual returns (address) {
        return msg.sender;
    }

    function _msgData() internal view virtual returns (bytes calldata) {
        // // solidity:0.8.0: warning: deprecated: use https://github.com/ethereum/solidity/issues/2901
        return msg.data;
    }
}

// ERC-20 Implementation: MIT
pragma solidity ^0.8.0;

import "JRC20.sol";
import "SafeMath.sol";
import "SafeMathInt.sol";
import "SafeMath256.sol";
import "DividendPayingTokenInterface.sol";
import "DividendPayingTokenOptionalInterface.sol";
import "Ownable.sol";

// ERC-20 Implementation: MIT
// Author: Roger Wu (https://github.com/rwgr-wu)
// This is a minimal ERC-20 token that allows anyone to pay and distribute ether
// to token holders as dividends and allows token holders to withdraw their dividends.
// Reference: the source code of FUDN: https://etherscan.io/address/0x8177196837822a3d9475ab03fca3d99e1f64c6
// ERC-20 Implementation: MIT
contract DividendPayingToken is
    JRC20,
    Ownable,
    DividendPayingTokenInterface,
    DividendPayingTokenOptionalInterface {
    using SafeMath for uint256;
    using SafeMathInt for int256;
    using SafeMath256 for uint256;

    address public CAKE; //CAKE

    // ERC-20 Implementation: MIT
    // For more discussion about choosing the value of 'magnitude',
    // see https://github.com/ethereum/EIP/issues/172#issuecomment-472351726
    uint256 internal constant magnitude = 2**128;

    int256 internal magnitudeDividendPerShare;

    // About DividendCorrection:
    // If the token balance of a 'user' is never changed, the dividend of 'user' can be computed with:
    // dividendOf('user') = dividendPerShare * balanceOf('user')
    // When 'balanceOf('user')' is changed (via minting/burning/transferring tokens),
    // dividendOf('user') should not be changed,
    // but the computed value of 'dividendPerShare * balanceOf('user')' is changed.
    // To keep the 'dividendOf('user')' consistent, we set a correction term.
```

## Contract Address

0x7185Fc499Ee5486b290B5e90e33abd3BcC0aeBE6

## TokenTracker

SnowKing (SnowKing)

## Contract Creator

0x729c159dd191d85e80af2199e457252f6683ca23

## Source Code

Contract Source Code Verified

## Contract Name

Token

## Other Settings

default evmVersion, MIT

## Compiler Version

v0.6.12+commit.27d51765

## Optimization Enabled

Yes with 200 runs

Code is truncated to fit the constraints of this document.

[The code in its entirety can be viewed here.](#)

# BEP-20 Contract Code Audit – Vulnerabilities Checked

| Vulnerability Tested                       | Scan     | Result          |
|--|----------|-----------------|
| Compiler Errors                            | Complete | ✓ Low / No Risk |
| Outdated Compiler Version                  | Complete | ✓ Low / No Risk |
| Integer Overflow                           | Complete | ✓ Low / No Risk |
| Integer Underflow                          | Complete | ✓ Low / No Risk |
| Floating Pragma                            | Complete | ✓ Low Risk      |
| Timestamp Dependency for Crucial Functions | Complete | ✓ Low / No Risk |
| Exposed _Transfer Function                 | Complete | ✓ Low / No Risk |
| Transaction-Ordering Dependency            | Complete | ✓ Low / No Risk |
| Unchecked Call Return Variable             | Complete | ✓ Low / No Risk |
| Use of Deprecated Functions                | Complete | ✓ Low / No Risk |
| Unprotected SELFDESTRUCT Instruction       | Complete | ✓ Low / No Risk |
| State Variable Default Visibility          | Complete | ✓ Low / No Risk |

The contract code is **verified** on BSCScan.

The vulnerabilities listed above were not found in the token's Smart Contract.

# Disclaimer



The opinions expressed in this document are for general informational purposes only and are **not intended to provide specific advice or recommendations for any individual or on any specific investment**. It is only intended to provide education and public knowledge regarding projects. This audit is only applied to the type of auditing specified in this report and the scope of given in the results. Other unknown security vulnerabilities are beyond responsibility. Dessert Finance only issues this report based on the attacks or vulnerabilities that already existed or occurred before the issuance of this report. For the emergence of new attacks or vulnerabilities that exist or occur in the future, Dessert Finance lacks the capability to judge its possible impact on the security status of smart contracts, thus taking no responsibility for them. The smart contract analysis and other contents of this report are based solely on the documents and materials that the contract provider has provided to Dessert Finance or was publicly available before the issuance of this report (issuance of report recorded via block number on cover page), if the documents and materials provided by the contract provider are missing, tampered, deleted, concealed or reflected in a situation that is inconsistent with the actual situation, or if the documents and materials provided are changed after the issuance of this report, Dessert Finance assumes no responsibility for the resulting loss or adverse effects. Due to the technical limitations of any organization, this report conducted by Dessert Finance still has the possibility that the entire risk cannot be completely detected. Dessert Finance disclaims any liability for the resulting losses.

Dessert Finance provides no guarantees against the sale of team tokens or the removal of liquidity by the project audited in this document. Even projects with a low risk score have been known to pull liquidity, sell all team tokens, or exit-scam. Please exercise caution when dealing with any cryptocurrency related platforms.

The final interpretation of this statement belongs to Dessert Finance.

Dessert Finance highly advises against using cryptocurrencies as speculative investments and they should be used solely for the utility they aim to provide.



# Thank You

DESSERT FINANCE LIGHT AUDIT HAS BEEN COMPLETED FOR **SNOWKING (SNOWKING)**

THIS AUDIT IS ONLY VALID IF VIEWED ON [HTTPS://WWW.DSSERTSWAP.FINANCE](https://www.dessertswap.finance)

[www.dessertswap.finance](http://www.dessertswap.finance)  
<https://t.me/dessertswap>