DESSERT
FINANCE

QubeToken (Qube)

BEP-20 & ERC-20 Audit

Performed at BSC block
11997780

**PERFORMED BY DESSERT FINANCE
ON CONTRACT ADDRESSES:**

0x1be7a6158c88647033b97e424763fb1a0853ad90
0x4b7b96b5a8af31b9986477bbc6ecb27ef44421dd
0x2fb2ff80191f897dacee74b4069794732266ca08
0x690821468421A59f663E7E3d1736cf9aE5A81Ff8
0x9f7ec80c1466045bdd6444ca91dfc47179ecd6b3
0xcaa5b6b6fa05555f37d6d8423e68b9c45283a259
0xdcE6e7E4467348d5AefA14407cE6c7f49585005d
0x2A0ADd12EDd08858a6b54a8179bD0b673b5Cd1eF
0x92682941480d4466017b32eFEA9f4c1ba3F2f0ad
0x5dC557d1440010784EB482e414B07CE2143c1d38

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.

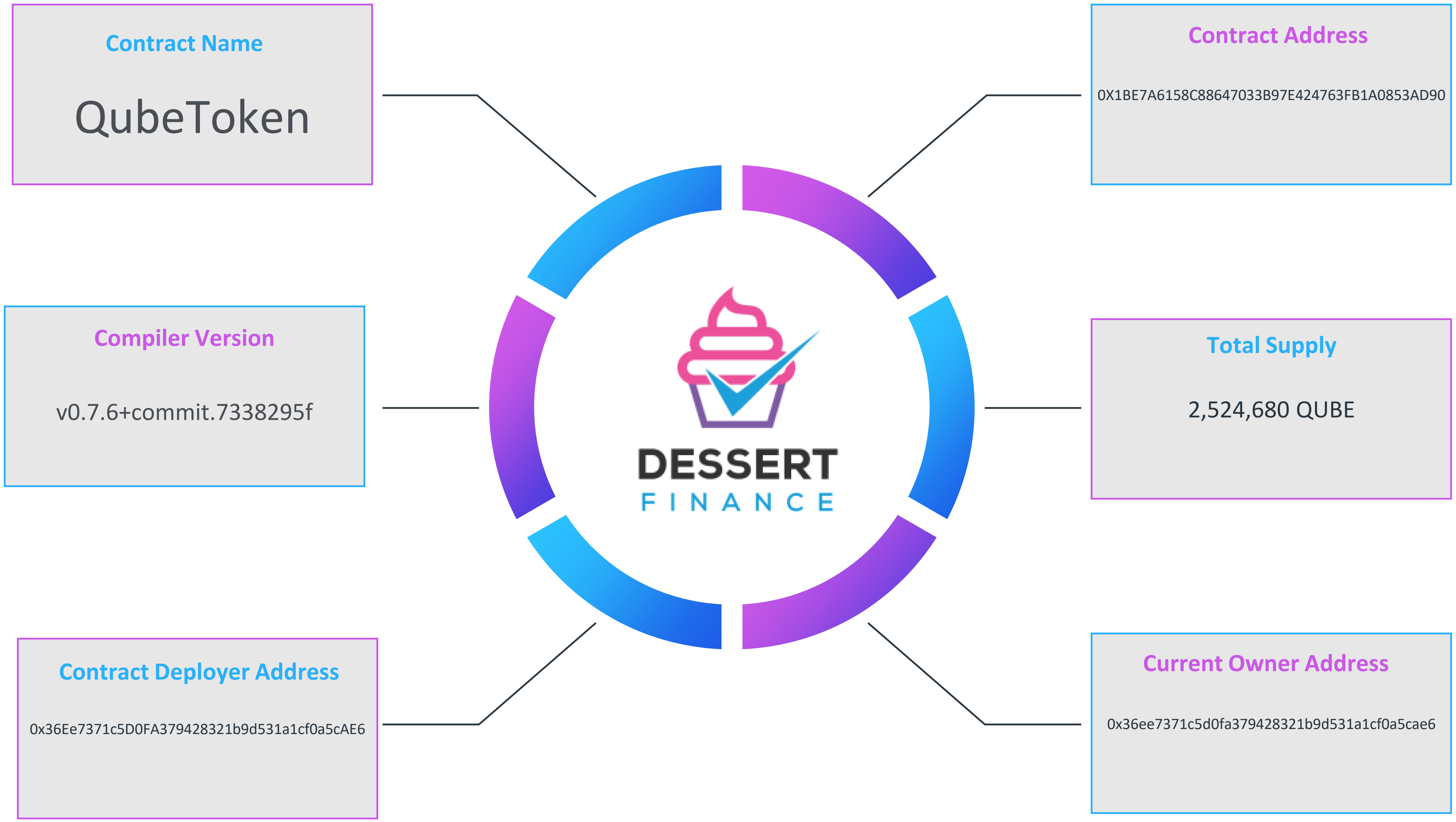
Dessert Finance does the legwork and provides public information about the project in an easy-to-understand format for the common person.

Agreeing to a project audit can be seen as a sign of confidence and is generally the first sign of trust for a project, but in no way guarantees that a team will not remove ***all*** liquidity ("Rug Pull"), sell off tokens, 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.

Table of Contents

ERC-20 Contract Code Audit – Token Overview



Contract Name

QubeToken

Compiler Version

v0.7.6+commit.7338295f

Contract Deployer Address

0x36Ee7371c5D0FA379428321b9d531a1cf0a5cAE6

Contract Address

0X1BE7A6158C88647033B97E424763FB1A0853AD90

Total Supply

2,524,680 QUBE

Current Owner Address

0x36ee7371c5d0fa379428321b9d531a1cf0a5cae6

Contract Code Audit – Overview

Dessert Finance was commissioned to perform an audit on QubeToken (Qube)

```
*Submitted for verification at Etherscan.io on 2021-08-30
*/

/ SPDX-License-Identifier: GPL-3.0-or-later

ragma solidity >=0.7.6;

*
* @dev Provides information about the current execution context, including the
* sender of the transaction and its data. While these are generally available
* via msg.sender and msg.data, they should not be accessed in such a direct
* manner, since when dealing with GSN 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.
*/
ontract Context {
   // Empty internal constructor, to prevent people from mistakenly deploying
   // an instance of this contract, which should be used via inheritance.
   constructor() {}

   function _msgSender() internal view returns (address payable) {
       return msg.sender;
   }

   function _msgData() internal view returns (bytes memory) {
       this; // silence state mutability warning without generating bytecode - see https://github.com/ethereum/solidity/issues/2691
       return msg.data;
   }
}

/**
* @dev Contract module which provides a basic access control mechanism, where
* there is an account (an owner) that can be granted exclusive access to
* specific functions.
*
* By default, the owner account will be the one that deploys the contract. This
* can later be changed with {transferOwnership}.
*
* This module is used through inheritance. It will make available the modifier
* `onlyOwner`, which can be applied to your functions to restrict their use to
* the owner.
*/
ontract Ownable is Context {

   address private _owner;

   event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);

   /**
    * @dev Initializes the contract setting the deployer as the initial owner.
    */
   constructor() {
       address msgSender = _msgSender();
       _owner = msgSender;
       emit OwnershipTransferred(address(0), msgSender);
   }

   /**
    * @dev Returns the address of the current owner.
    */
   function owner() public view returns (address) {
       return _owner;
   }

   /**
    * @dev Throws if called by any account other than the owner.
    */
   modifier onlyOwner() {
```

Contract Address
0x1BE7A6158c88647033b97e424763fB1A0853AD90

TokenTracker
QubeToken (Qube)

Contract Creator
0x36ee7371c5d0fa379428321b9d531a1cf0a5cae6

Source Code
Contract Source Code Verified

Contract Name
QubeToken

Other Settings
default evmVersion, None

Compiler Version
v0.7.6+commit.7338295f

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.]

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

ERC-20 Contract Code Audit – Vulnerabilities Checked

Vulnerability Tested	AI Scan	Human Review	Result
Integer Overflow	Complete	Complete	✓ Low / No Risk
Integer Underflow	Complete	Complete	✓ Low / No Risk
Correct Token Standards Implementation	Complete	Complete	✓ Low / No Risk
Timestamp Dependency for Crucial Functions	Complete	Complete	✓ Low / No Risk
Exposed _Transfer Function	Complete	Complete	✓ Low / No Risk
Transaction-Ordering Dependency	Complete	Complete	✓ Low / No Risk
Unchecked Call Return Variable	Complete	Complete	✓ Low / No Risk
Use of Deprecated Functions	Complete	Complete	✓ Low / No Risk
Unprotected SELFDESTRUCT Instruction	Complete	Complete	✓ Low / No Risk
State Variable Default Visibility	Complete	Complete	✓ Low / No Risk
Deployer Can Access User Funds	Complete	Complete	✓ Low / No Risk

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

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

QubeToken

ERC-20 Contract Code Audit – Contract Ownership

Contract Ownership has not been renounced at the time of Audit

The contract ownership is not currently renounced.

We have placed the contract owner address below for your viewing:

0x36ee7371c5d0fa379428321b9d531a1cf0a5cae6

ERC-20 Liquidity Ownership – Locked / Unlocked

No locked liquidity information available.

This project is still on the Testnet. At this time liquidity locking information is not available as there is no liquidity be locked at this stage of development.

Contract Code Audit – Overview

Dessert Finance was commissioned to perform an audit on MasterChef

```
*Submitted for verification at Etherscan.io on 2021-08-30
*/

// SPDX-License-Identifier: UNLICENSED

// File: @openzeppelin/contracts/token/ERC20/IERC20.sol

pragma solidity ^0.6.0
pragma experimental ABIEncoderV2;

/**
 * @dev Interface of the ERC20 standard as defined in the EIP.
 */
interface IERC20 {
    /**
     * @dev Returns the amount of tokens in existence.
     */
    function totalSupply() external view returns (uint256);

    /**
     * @dev Returns the amount of tokens owned by `account`.
     */
    function balanceOf(address account) external view returns (uint256);

    /**
     * @dev Moves `amount` tokens from the caller's account to `recipient`.
     *
     * Returns a boolean value indicating whether the operation succeeded.
     *
     * Emits a {Transfer} event.
     */
    function transfer(address recipient, uint256 amount) external returns (bool);

    /**
     * @dev Returns the remaining number of tokens that `spender` will be
     * allowed to spend on behalf of `owner` through {transferFrom}. This is
     * zero by default.
     *
     * This value changes when {approve} or {transferFrom} are called.
     */
    function allowance(address owner, address spender) external view returns (uint256);

    /**
     * @dev Sets `amount` as the allowance of `spender` over the caller's tokens.
     *
     * Returns a boolean value indicating whether the operation succeeded.
     *
     * IMPORTANT: Beware that changing an allowance with this method brings the risk
     * that someone may use both the old and the new allowance by unfortunate
     * transaction ordering. One possible solution to mitigate this race
     * condition is to first reduce the spender's allowance to 0 and set the
     * desired value afterwards:
     * https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
     *
     * Emits an {Approval} event.
     */
    function approve(address spender, uint256 amount) external returns (bool);

    /**
     * @dev Moves `amount` tokens from `sender` to `recipient` using the
     * allowance mechanism. `amount` is then deducted from the caller's
     * allowance.
     *
     * Returns a boolean value indicating whether the operation succeeded.
     *
     * Emits a {Transfer} event.
     */
    function transferFrom(address sender, address recipient, uint256 amount) external returns (bool);
```

Contract Address
0x4B7B96B5A8AF31B9986477BBc6ECb27eF44421dD

Contract Creator
0x36ee7371c5d0fa379428321b9d531a1cf0a5cae6

Source Code
Contract Source Code Verified

Contract Name
MasterChef

Other Settings
default evmVersion, None

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.

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

ERC-20 Contract Code Audit – Vulnerabilities Checked

Vulnerability Tested	AI Scan	Human Review	Result
Integer Overflow	Complete	Complete	✓ Low / No Risk
Integer Underflow	Complete	Complete	✓ Low / No Risk
Correct Token Standards Implementation	Complete	Complete	✓ Low / No Risk
Timestamp Dependency for Crucial Functions	Complete	Complete	✓ Low / No Risk
Exposed _Transfer Function	Complete	Complete	✓ Low / No Risk
Transaction-Ordering Dependency	Complete	Complete	✓ Low / No Risk
Unchecked Call Return Variable	Complete	Complete	✓ Low / No Risk
Use of Deprecated Functions	Complete	Complete	✓ Low / No Risk
Unprotected SELFDESTRUCT Instruction	Complete	Complete	✓ Low / No Risk
State Variable Default Visibility	Complete	Complete	✓ Low / No Risk
Deployer Can Access User Funds	Complete	Complete	✓ Low / No Risk

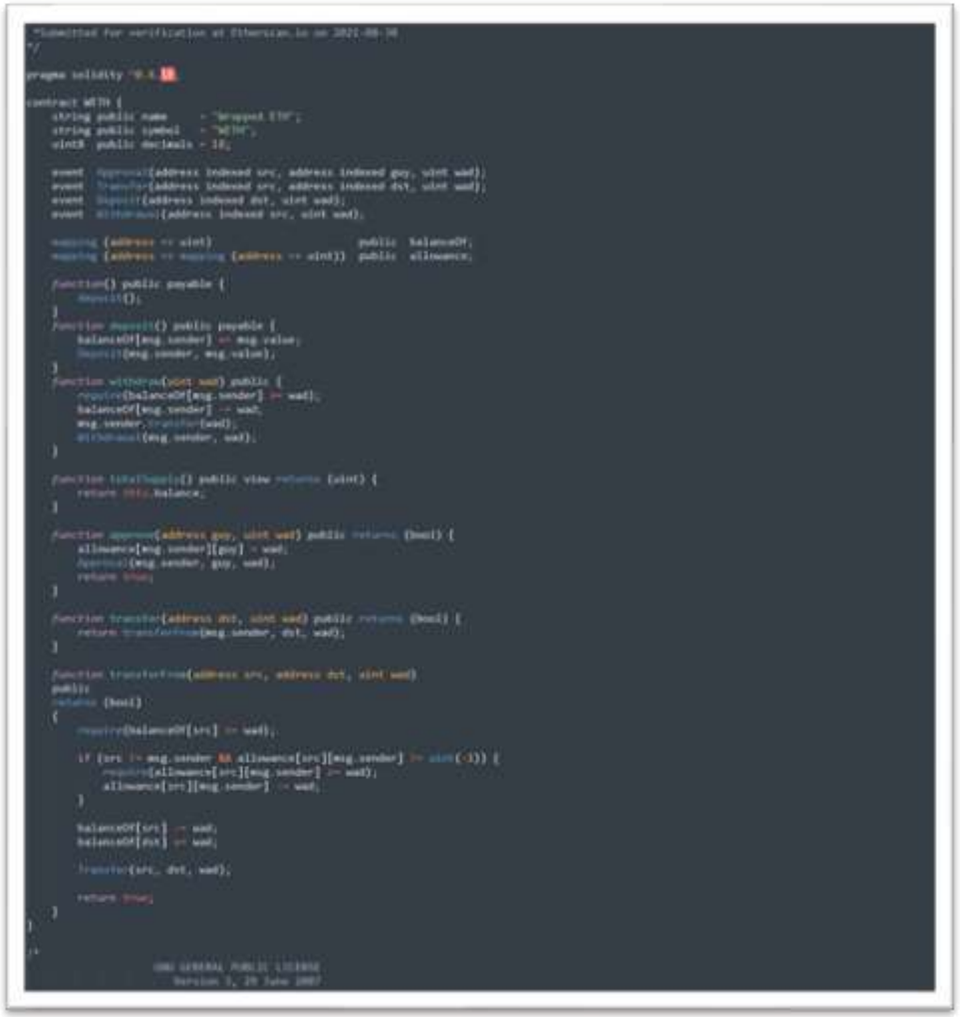
The contract code is **verified** on EtherScan.

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

MasterChef

Contract Code Audit – Overview

Dessert Finance was commissioned to perform an audit on Wrapped ETH (WETH)



```
*Submitted for verification at Etherscan.io on 2021-08-30
*/

pragma solidity ^0.4.18;

contract WETH {
    string public name     = "Wrapped ETH";
    string public symbol   = "WETH";
    uint8  public decimals = 18;

    event  Approval(address indexed src, address indexed guy, uint wad);
    event  Transfer(address indexed src, address indexed dst, uint wad);
    event  Deposit(address indexed dst, uint wad);
    event  Withdrawal(address indexed src, uint wad);

    mapping (address => uint)                       public  balanceOf;
    mapping (address => mapping (address => uint))  public  allowance;

    function() public payable {
        deposit();
    }
    function deposit() public payable {
        balanceOf[msg.sender] += msg.value;
        Deposit(msg.sender, msg.value);
    }
    function withdraw(uint wad) public {
        require(balanceOf[msg.sender] >= wad);
        balanceOf[msg.sender] -= wad;
        msg.sender.transfer(wad);
        Withdrawal(msg.sender, wad);
    }

    function totalSupply() public view returns (uint) {
        return this.balance;
    }

    function approve(address guy, uint wad) public returns (bool) {
        allowance[msg.sender][guy] = wad;
        Approval(msg.sender, guy, wad);
        return true;
    }

    function transfer(address dst, uint wad) public returns (bool) {
        return transferFrom(msg.sender, dst, wad);
    }

    function transferFrom(address src, address dst, uint wad)
    public
    returns (bool)
    {
        require(balanceOf[src] >= wad);

        if (src != msg.sender && allowance[src][msg.sender] != uint(-1)) {
            require(allowance[src][msg.sender] >= wad);
            allowance[src][msg.sender] -= wad;
        }

        balanceOf[src] -= wad;
        balanceOf[dst] += wad;

        Transfer(src, dst, wad);

        return true;
    }
}

/*
                    GNU GENERAL PUBLIC LICENSE
                       Version 3, 29 June 2007
```

Contract Address
0x2fB2fF80191F897DaCeE74B4069794732266CA08

TokenTracker
Wrapped ETH (WETH)

Contract Creator
0x36ee7371c5d0fa379428321b9d531a1cf0a5cae6

Source Code
Contract Source Code Verified

Contract Name
WETH

Other Settings
default evmVersion, None

Compiler Version
v0.4.26+commit.4563c3fc

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.

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

ERC-20 Contract Code Audit – Vulnerabilities Checked

Vulnerability Tested	AI Scan	Human Review	Result
Integer Overflow	Complete	Complete	✓ Low / No Risk
Integer Underflow	Complete	Complete	✓ Low / No Risk
Correct Token Standards Implementation	Complete	Complete	✓ Low / No Risk
Timestamp Dependency for Crucial Functions	Complete	Complete	✓ Low / No Risk
Exposed _Transfer Function	Complete	Complete	✓ Low / No Risk
Transaction-Ordering Dependency	Complete	Complete	✓ Low / No Risk
Unchecked Call Return Variable	Complete	Complete	✓ Low / No Risk
Use of Deprecated Functions	Complete	Complete	✓ Low / No Risk
Unprotected SELFDESTRUCT Instruction	Complete	Complete	✓ Low / No Risk
State Variable Default Visibility	Complete	Complete	✓ Low / No Risk
Deployer Can Access User Funds	Complete	Complete	✓ Low / No Risk

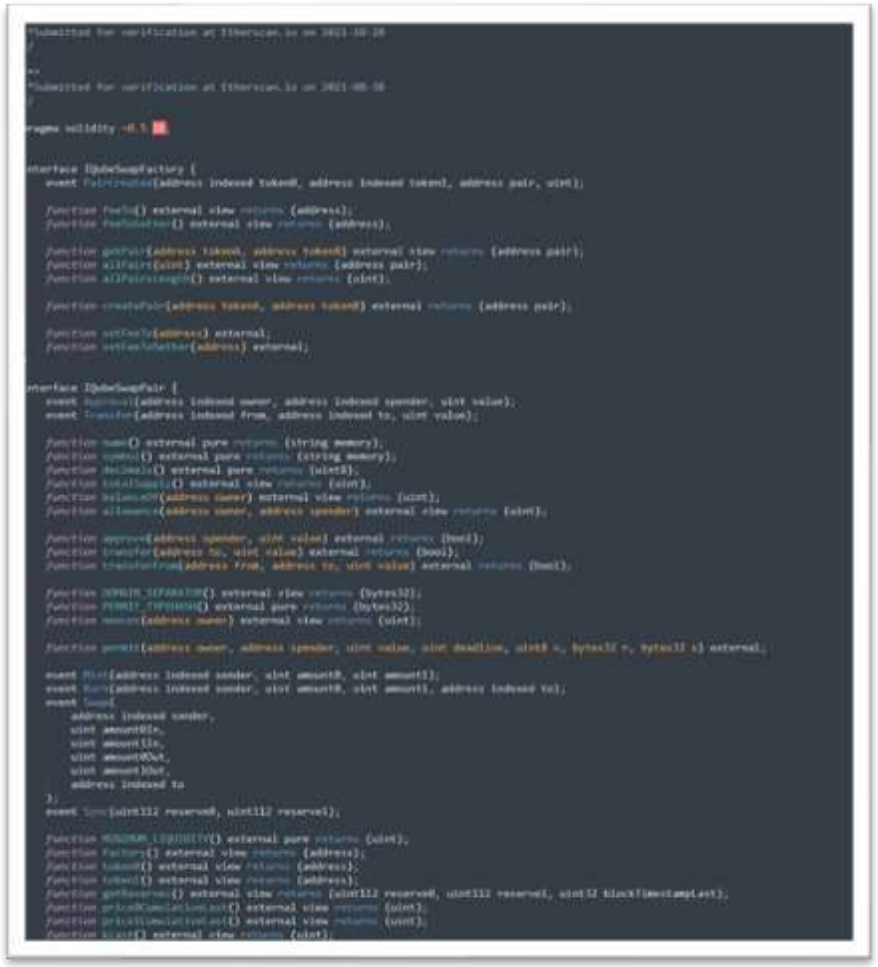
The contract code is **verified** on EtherScan.

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

Wrapped ETH (WETH)

Contract Code Audit – Overview

Dessert Finance was commissioned to perform an audit on QubeSwapFactory



```
*Submitted for verification at Etherscan.io on 2021-10-28
/

**
*Submitted for verification at Etherscan.io on 2021-08-30
/

ragma solidity =0.5.16;

nterface IQubeSwapFactory {
    event PairCreated(address indexed token0, address indexed token1, address pair, uint);

    function feeTo() external view returns (address);
    function feeToSetter() external view returns (address);

    function getPair(address tokenA, address tokenB) external view returns (address pair);
    function allPairs(uint) external view returns (address pair);
    function allPairsLength() external view returns (uint);

    function createPair(address tokenA, address tokenB) external returns (address pair);

    function setFeeTo(address) external;
    function setFeeToSetter(address) external;


nterface IQubeSwapPair {
    event Approval(address indexed owner, address indexed spender, uint value);
    event Transfer(address indexed from, address indexed to, uint value);

    function name() external pure returns (string memory);
    function symbol() external pure returns (string memory);
    function decimals() external pure returns (uint8);
    function totalSupply() external view returns (uint);
    function balanceOf(address owner) external view returns (uint);
    function allowance(address owner, address spender) external view returns (uint);

    function approve(address spender, uint value) external returns (bool);
    function transfer(address to, uint value) external returns (bool);
    function transferFrom(address from, address to, uint value) external returns (bool);

    function DOMAIN_SEPARATOR() external view returns (bytes32);
    function PERMIT_TYPEHASH() external pure returns (bytes32);
    function nonces(address owner) external view returns (uint);

    function permit(address owner, address spender, uint value, uint deadline, uint8 v, bytes32 r, bytes32 s) external;

    event Mint(address indexed sender, uint amount0, uint amount1);
    event Burn(address indexed sender, uint amount0, uint amount1, address indexed to);
    event Swap(
        address indexed sender,
        uint amount0In,
        uint amount1In,
        uint amount0Out,
        uint amount1Out,
        address indexed to
    );
    event Sync(uint112 reserve0, uint112 reserve1);

    function MINIMUM_LIQUIDITY() external pure returns (uint);
    function factory() external view returns (address);
    function token0() external view returns (address);
    function token1() external view returns (address);
    function getReserves() external view returns (uint112 reserve0, uint112 reserve1, uint32 blockTimestampLast);
    function price0CumulativeLast() external view returns (uint);
    function price1CumulativeLast() external view returns (uint);
    function kLast() external view returns (uint);
```

Contract Address
0x690821468421A59f663E7E3d1736cf9aE5A81Ff8

Contract Creator
0x36ee7371c5d0fa379428321b9d531a1cf0a5cae6

Source Code
Contract Source Code Verified

Contract Name
QubeSwapFactory

Other Settings
default evmVersion, None

Compiler Version
v0.5.16+commit.9c3226ce

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.

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

ERC-20 Contract Code Audit – Vulnerabilities Checked

Vulnerability Tested	AI Scan	Human Review	Result
Integer Overflow	Complete	Complete	✓ Low / No Risk
Integer Underflow	Complete	Complete	✓ Low / No Risk
Correct Token Standards Implementation	Complete	Complete	✓ Low / No Risk
Timestamp Dependency for Crucial Functions	Complete	Complete	✓ Low / No Risk
Exposed _Transfer Function	Complete	Complete	✓ Low / No Risk
Transaction-Ordering Dependency	Complete	Complete	✓ Low / No Risk
Unchecked Call Return Variable	Complete	Complete	✓ Low / No Risk
Use of Deprecated Functions	Complete	Complete	✓ Low / No Risk
Unprotected SELFDESTRUCT Instruction	Complete	Complete	✓ Low / No Risk
State Variable Default Visibility	Complete	Complete	✓ Low / No Risk
Deployer Can Access User Funds	Complete	Complete	✓ Low / No Risk

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

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

QubeSwapFactory

Contract Code Audit – Overview

Dessert Finance was commissioned to perform an audit on QubeSwapRouter

```
Submitted for verification at Etherscan.io on 2021-10-20

 File: @qubeswap\lib\contracts\libraries\TransferHelper.sol

agma solidity >=0.6.0;

 helper methods for interacting with ERC20 tokens and sending ETH that do not consistently return true/false
brary TransferHelper {
  function safeApprove(address token, address to, uint value) internal {
      // bytes4(keccak256(bytes('approve(address,uint256)')));
      (bool success, bytes memory data) = token.call(abi.encodeWithSelector(0x095ea7b3, to, value));
      require(success && (data.length == 0 || abi.decode(data, (bool))), 'TransferHelper: APPROVE_FAILED');
  }

  function safeTransfer(address token, address to, uint value) internal {
      // bytes4(keccak256(bytes('transfer(address,uint256)')));
      (bool success, bytes memory data) = token.call(abi.encodeWithSelector(0xa9059cbb, to, value));
      require(success && (data.length == 0 || abi.decode(data, (bool))), 'TransferHelper: TRANSFER_FAILED');
  }

  function safeTransferFrom(address token, address from, address to, uint value) internal {
      // bytes4(keccak256(bytes('transferFrom(address,address,uint256)')));
      (bool success, bytes memory data) = token.call(abi.encodeWithSelector(0x23b872dd, from, to, value));
      require(success && (data.length == 0 || abi.decode(data, (bool))), 'TransferHelper: TRANSFER_FROM_FAILED');
  }

  function safeTransferETH(address to, uint value) internal {
      (bool success,) = to.call{value:value}(new bytes(0));
      require(success, 'TransferHelper: ETH_TRANSFER_FAILED');
  }


 File: contracts\interfaces\IQubeSwapRouter01.sol

agma solidity >=0.6.2;

terface IQubeSwapRouter01 {
  function factory() external pure returns (address);
  function WETH() external pure returns (address);

  function addLiquidity(
      address tokenA,
      address tokenB,
      uint amountADesired,
      uint amountBDesired,
      uint amountAMin,
      uint amountBMin,
      address to,
      uint deadline
  ) external returns (uint amountA, uint amountB, uint liquidity);
  function addLiquidityETH(
      address token,
      uint amountTokenDesired,
      uint amountTokenMin,
      uint amountETHMin,
      address to,
      uint deadline
  ) external payable returns (uint amountToken, uint amountETH, uint liquidity);
  function removeLiquidity(
      address tokenA,
      address tokenB,
      uint liquidity,
      uint amountAMin,
      uint amountBMin,
      address to,
      uint deadline
  ) external returns (uint amountA, uint amountB);
  function removeLiquidityETH(
      address token,
```

Contract Address
0x9f7Ec80C1466045BDD6444cA91DFc47179ECd6b3

Contract Creator
0x36ee7371c5d0fa379428321b9d531a1cf0a5cae6

Source Code
Contract Source Code Verified

Contract Name
QubeSwapRouter

Other Settings
default evmVersion, None

Compiler Version
v0.6.6+commit.6c089d02

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.

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

ERC-20 Contract Code Audit – Vulnerabilities Checked

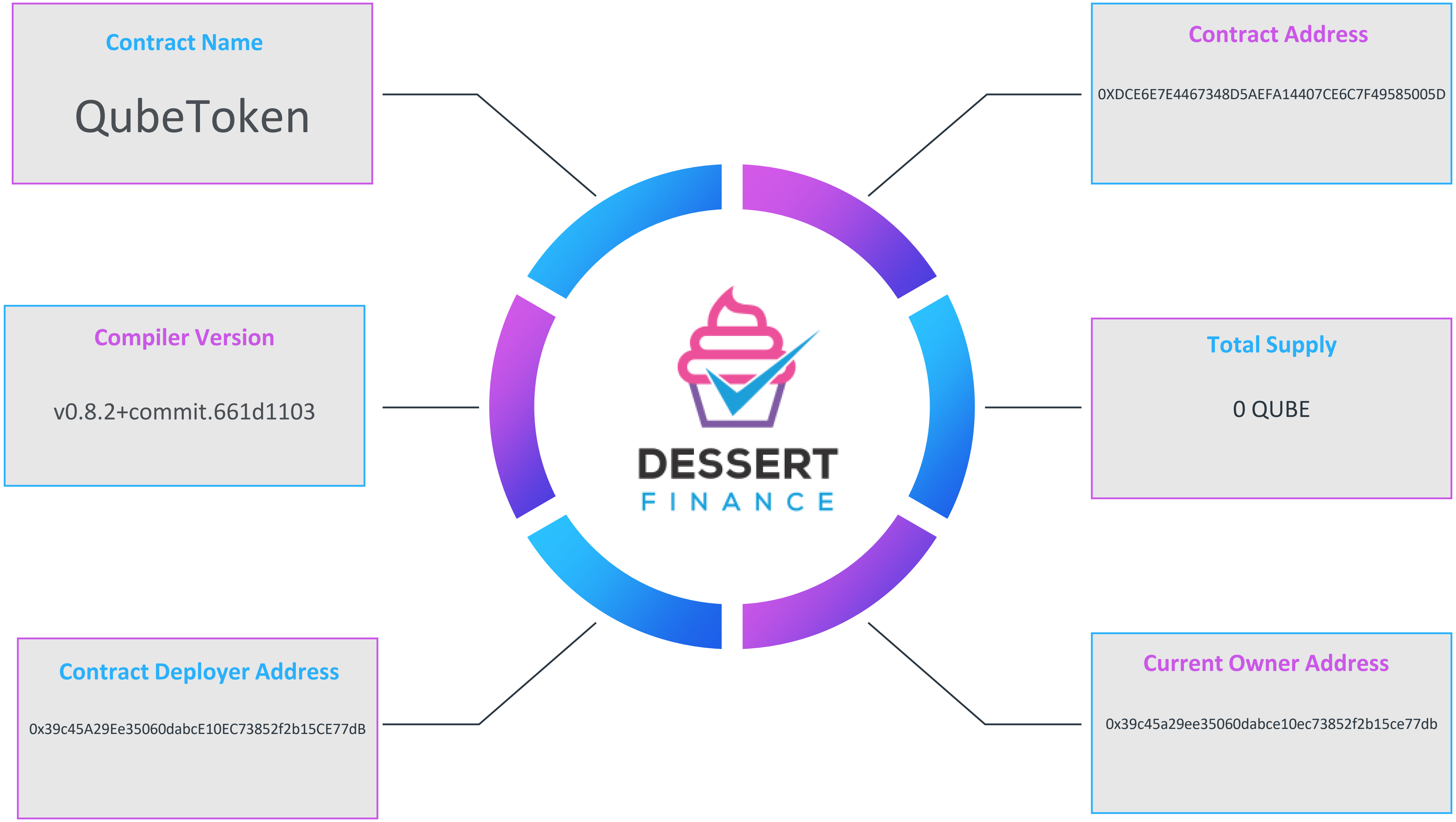
Vulnerability Tested	AI Scan	Human Review	Result
Integer Overflow	Complete	Complete	✓ Low / No Risk
Integer Underflow	Complete	Complete	✓ Low / No Risk
Correct Token Standards Implementation	Complete	Complete	✓ Low / No Risk
Timestamp Dependency for Crucial Functions	Complete	Complete	✓ Low / No Risk
Exposed _Transfer Function	Complete	Complete	✓ Low / No Risk
Transaction-Ordering Dependency	Complete	Complete	✓ Low / No Risk
Unchecked Call Return Variable	Complete	Complete	✓ Low / No Risk
Use of Deprecated Functions	Complete	Complete	✓ Low / No Risk
Unprotected SELFDESTRUCT Instruction	Complete	Complete	✓ Low / No Risk
State Variable Default Visibility	Complete	Complete	✓ Low / No Risk
Deployer Can Access User Funds	Complete	Complete	✓ Low / No Risk

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

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

QubeSwapRouter

BEP-20 Contract Code Audit – Token Overview



Contract Name

QubeToken

Contract Address

0XDCE6E7E4467348D5AEFA14407CE6C7F49585005D

Compiler Version

v0.8.2+commit.661d1103

Total Supply

0 QUBE

Contract Deployer Address

0x39c45A29Ee35060dabcE10EC73852f2b15CE77dB

Current Owner Address

0x39c45a29ee35060dabce10ec73852f2b15ce77db

Contract Code Audit – Overview

Dessert Finance was commissioned to perform an audit on QubeToken (Qube)

```
*Submitted for verification at BscScan.com on 2021-10-20
*/

// File: @openzeppelin/contracts/token/BEP20/IBEP20.sol

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

/**
 * @dev Interface of the BEP20 standard as defined in the EIP.
 */
interface IBEP20 {
    /**
     * @dev Returns the amount of tokens in existence.
     */
    function totalSupply() external view returns (uint256);

    /**
     * @dev Returns the amount of tokens owned by `account`.
     */
    function balanceOf(address account) external view returns (uint256);

    /**
     * @dev Moves `amount` tokens from the caller's account to `recipient`.
     *
     * Returns a boolean value indicating whether the operation succeeded.
     *
     * Emits a {Transfer} event.
     */
    function transfer(address recipient, uint256 amount) external returns (bool);

    /**
     * @dev Returns the remaining number of tokens that `spender` will be
     * allowed to spend on behalf of `owner` through {transferFrom}. This is
     * zero by default.
     *
     * This value changes when {approve} or {transferFrom} are called.
     */
    function allowance(address owner, address spender) external view returns (uint256);

    /**
     * @dev Sets `amount` as the allowance of `spender` over the caller's tokens.
     *
     * Returns a boolean value indicating whether the operation succeeded.
     *
     * IMPORTANT: Beware that changing an allowance with this method brings the risk
     * that someone may use both the old and the new allowance by unfortunate
     * transaction ordering. One possible solution to mitigate this race
     * condition is to first reduce the spender's allowance to 0 and set the
     * desired value afterwards:
     * https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
     *
     * Emits an {Approval} event.
     */
    function approve(address spender, uint256 amount) external returns (bool);

    /**
     * @dev Moves `amount` tokens from `sender` to `recipient` using the
     * allowance mechanism. `amount` is then deducted from the caller's
     * allowance.
     *
     * Returns a boolean value indicating whether the operation succeeded.
     *
     * Emits a {Transfer} event.
     */
    function transferFrom(
        address sender,
        address recipient,
        uint256 amount
    ) external returns (bool);
```

Contract Address
0xdcE6e7E4467348d5AefA14407cE6c7f49585005d

TokenTracker
QUBE TOKEN (QUBE)

Contract Creator
0x39c45a29ee35060dabce10ec73852f2b15ce77db

Source Code
Contract Source Code Verified

Contract Name
QubeToken

Other Settings
default evmVersion, MIT

Compiler Version
v0.8.2+commit.661d1103

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.]

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

BEP-20 Contract Code Audit – Vulnerabilities Checked

Vulnerability Tested	AI Scan	Human Review	Result
Integer Overflow	Complete	Complete	✓ Low / No Risk
Integer Underflow	Complete	Complete	✓ Low / No Risk
Correct Token Standards Implementation	Complete	Complete	✓ Low / No Risk
Timestamp Dependency for Crucial Functions	Complete	Complete	✓ Low / No Risk
Exposed _Transfer Function	Complete	Complete	✓ Low / No Risk
Transaction-Ordering Dependency	Complete	Complete	✓ Low / No Risk
Unchecked Call Return Variable	Complete	Complete	✓ Low / No Risk
Use of Deprecated Functions	Complete	Complete	✓ Low / No Risk
Unprotected SELFDESTRUCT Instruction	Complete	Complete	✓ Low / No Risk
State Variable Default Visibility	Complete	Complete	✓ Low / No Risk
Deployer Can Access User Funds	Complete	Complete	✓ Low / No Risk

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

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

QubeToken

BEP-20 Contract Code Audit – Contract Ownership

Contract Ownership has not been renounced at the time of Audit

The contract ownership is not currently renounced.

We have placed the contract owner address below for your viewing:

0x39c45a29ee35060dabce10ec73852f2b15ce77db

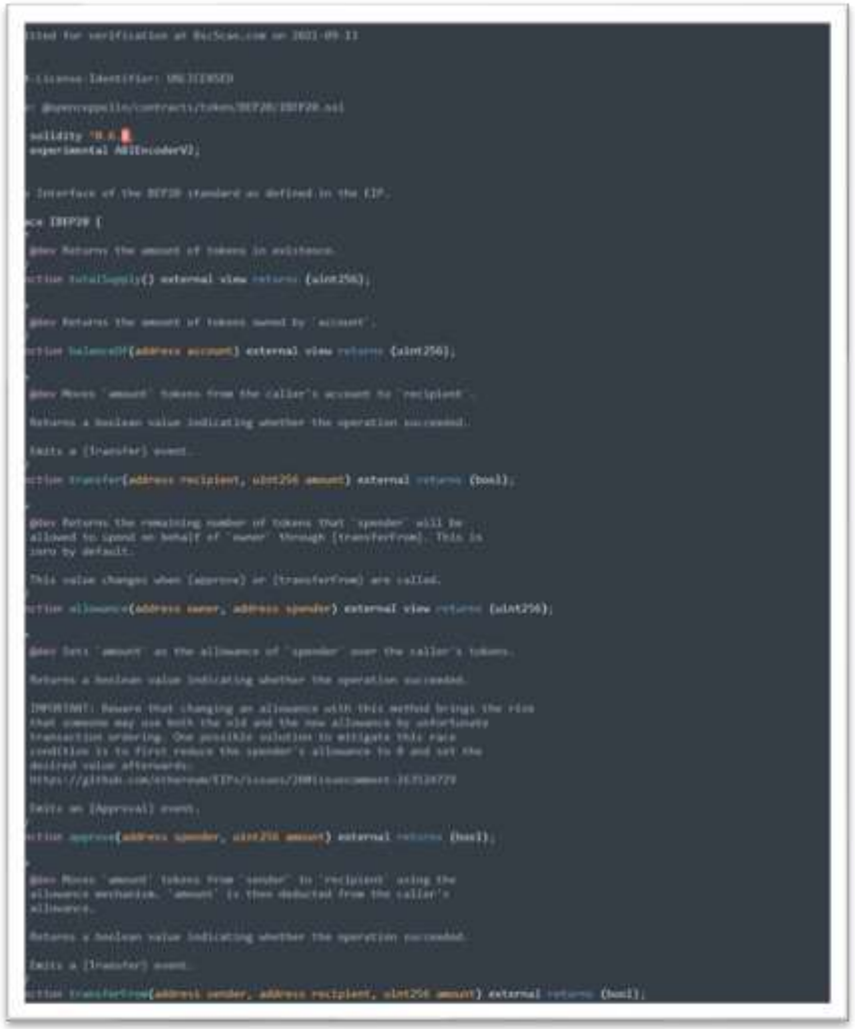
BEP-20 Liquidity Ownership – Locked / Unlocked

No locked liquidity information available.

This project is still on the Testnet. At this time liquidity locking information is not available as there is no liquidity be locked at this stage of development.

Contract Code Audit – Overview

Dessert Finance was commissioned to perform an audit on MasterChef



```
itted for verification at BscScan.com on 2021-09-11

-License-Identifier: UNLICENSED

: @openzeppelin/contracts/token/BEP20/IBEP20.sol

 solidity "0.6.[illegible]
 experimental ABIEncoderV2;

 Interface of the BEP20 standard as defined in the EIP.

ce IBEP20 {

 @dev Returns the amount of tokens in existence.

ction totalSupply() external view returns (uint256);

 @dev Returns the amount of tokens owned by `account`.

ction balanceOf(address account) external view returns (uint256);

 @dev Moves `amount` tokens from the caller's account to `recipient`.

 Returns a boolean value indicating whether the operation succeeded.

 Emits a {Transfer} event.

ction transfer(address recipient, uint256 amount) external returns (bool);

 @dev Returns the remaining number of tokens that `spender` will be
 allowed to spend on behalf of `owner` through {transferFrom}. This is
 zero by default.

 This value changes when {approve} or {transferFrom} are called.

ction allowance(address owner, address spender) external view returns (uint256);

 @dev Sets `amount` as the allowance of `spender` over the caller's tokens.

 Returns a boolean value indicating whether the operation succeeded.

 IMPORTANT: Beware that changing an allowance with this method brings the risk
 that someone may use both the old and the new allowance by unfortunate
 transaction ordering. One possible solution to mitigate this race
 condition is to first reduce the spender's allowance to 0 and set the
 desired value afterwards:
 https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729

 Emits an {Approval} event.

ction approve(address spender, uint256 amount) external returns (bool);

 @dev Moves `amount` tokens from `sender` to `recipient` using the
 allowance mechanism. `amount` is then deducted from the caller's
 allowance.

 Returns a boolean value indicating whether the operation succeeded.

 Emits a {Transfer} event.

ction transferFrom(address sender, address recipient, uint256 amount) external returns (bool);
```

Contract Address
0xCAA5B6b6fa05555F37d6d8423e68b9c45283A259

Contract Creator
0x36ee7371c5d0fa379428321b9d531a1cf0a5cae6

Source Code
Contract Source Code Verified

Contract Name
MasterChef

Other Settings
default evmVersion, None

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.

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

BEP-20 Contract Code Audit – Vulnerabilities Checked

Vulnerability Tested	AI Scan	Human Review	Result
Integer Overflow	Complete	Complete	✓ Low / No Risk
Integer Underflow	Complete	Complete	✓ Low / No Risk
Correct Token Standards Implementation	Complete	Complete	✓ Low / No Risk
Timestamp Dependency for Crucial Functions	Complete	Complete	✓ Low / No Risk
Exposed _Transfer Function	Complete	Complete	✓ Low / No Risk
Transaction-Ordering Dependency	Complete	Complete	✓ Low / No Risk
Unchecked Call Return Variable	Complete	Complete	✓ Low / No Risk
Use of Deprecated Functions	Complete	Complete	✓ Low / No Risk
Unprotected SELFDESTRUCT Instruction	Complete	Complete	✓ Low / No Risk
State Variable Default Visibility	Complete	Complete	✓ Low / No Risk
Deployer Can Access User Funds	Complete	Complete	✓ Low / No Risk

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

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

MasterChef

Contract Code Audit – Overview

Dessert Finance was commissioned to perform an audit on QubeSwapRouter

```
r verification at BscScan.com on 2021-10-18

eSwap\lib\contracts\libraries\TransferHelper.sol

ty >=0.6.[illegible]

ods for interacting with ERC20 tokens and sending BNB that do not consistently return true/false
ferHelper {
safeApprove(address token, address to, uint value) internal {
bes4(keccak256(bytes('approve(address,uint256)')));
 success, bytes memory data) = token.call(abi.encodeWithSelector(0x095ea7b3, to, value));
re(success && (data.length == 0 || abi.decode(data, (bool))), 'TransferHelper: APPROVE_FAILED');

safeTransfer(address token, address to, uint value) internal {
tes4(keccak256(bytes('transfer(address,uint256)')));
 success, bytes memory data) = token.call(abi.encodeWithSelector(0xa9059cbb, to, value));
re(success && (data.length == 0 || abi.decode(data, (bool))), 'TransferHelper: TRANSFER_FAILED');

safeTransferFrom(address token, address from, address to, uint value) internal {
tes4(keccak256(bytes('transferFrom(address,address,uint256)')));
 success, bytes memory data) = token.call(abi.encodeWithSelector(0x23b872dd, from, to, value));
re(success && (data.length == 0 || abi.decode(data, (bool))), 'TransferHelper: TRANSFER_FROM_FAILED');

safeTransferBNB(address to, uint value) internal {
 success,) = to.call{value:value}(new bytes(0));
re(success, 'TransferHelper: BNB_TRANSFER_FAILED');


racts\interfaces\IQubeSwapRouter01.sol

ty >=0.6.[illegible]

beSwapRouter01 {
factory() external pure returns (address);
WBNB() external pure returns (address);

addLiquidity(
ss tokenA,
ss tokenB,
amountADesired,
amountBDesired,
amountAMin,
amountBMin,
ss to,
deadline
) returns (uint amountA, uint amountB, uint liquidity);
addLiquidityBNB(
ss token,
amountTokenDesired,
amountTokenMin,
amountBNBMin,
ss to,
deadline
) payable returns (uint amountToken, uint amountBNB, uint liquidity);
removeLiquidity(
ss tokenA,
ss tokenB,
liquidity,
amountAMin,
amountBMin,
ss to,
deadline
) returns (uint amountA, uint amountB);
```

Contract Address
0x2A0ADd12EDd08858a6b54a8179bD0b673b5Cd1eF

Contract Creator
0x39c45a29ee35060dabce10ec73852f2b15ce77db

Source Code
Contract Source Code Verified

Contract Name
QubeSwapRouter

Other Settings
default evmVersion, MIT

Compiler Version
v0.6.6+commit.6c089d02

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.

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

BEP-20 Contract Code Audit – Vulnerabilities Checked

Vulnerability Tested	AI Scan	Human Review	Result
Integer Overflow	Complete	Complete	✓ Low / No Risk
Integer Underflow	Complete	Complete	✓ Low / No Risk
Correct Token Standards Implementation	Complete	Complete	✓ Low / No Risk
Timestamp Dependency for Crucial Functions	Complete	Complete	✓ Low / No Risk
Exposed _Transfer Function	Complete	Complete	✓ Low / No Risk
Transaction-Ordering Dependency	Complete	Complete	✓ Low / No Risk
Unchecked Call Return Variable	Complete	Complete	✓ Low / No Risk
Use of Deprecated Functions	Complete	Complete	✓ Low / No Risk
Unprotected SELFDESTRUCT Instruction	Complete	Complete	✓ Low / No Risk
State Variable Default Visibility	Complete	Complete	✓ Low / No Risk
Deployer Can Access User Funds	Complete	Complete	✓ Low / No Risk

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

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

QubeSwapRouter

Contract Code Audit – Overview

Dessert Finance was commissioned to perform an audit on QubeSwapFactory

```
rification at BscScan.com on 2021-10-28

.5.16;

pFactory {
ted(address indexed token0, address indexed token1, address pair, uint);

() external view returns (address);
etter() external view returns (address);

ir(address tokenA, address tokenB) external view returns (address pair);
irs(uint) external view returns (address pair);
irsLength() external view returns (uint);

ePair(address tokenA, address tokenB) external returns (address pair);

eTo(address) external;
eToSetter(address) external;


pPair {
(address indexed owner, address indexed spender, uint value);
(address indexed from, address indexed to, uint value);

) external pure returns (string memory);
l() external pure returns (string memory);
als() external pure returns (uint8);
Supply() external view returns (uint);
ceOf(address owner) external view returns (uint);
ance(address owner, address spender) external view returns (uint);

ve(address spender, uint value) external returns (bool);
fer(address to, uint value) external returns (bool);
ferFrom(address from, address to, uint value) external returns (bool);

N_SEPARATOR() external view returns (bytes32);
T_TYPEHASH() external pure returns (bytes32);
s(address owner) external view returns (uint);

t(address owner, address spender, uint value, uint deadline, uint8 v, bytes32 r, bytes32 s) external;

ress indexed sender, uint amount0, uint amount1);
ress indexed sender, uint amount0, uint amount1, address indexed to);

dexed sender,
t0In,
t1In,
t0Out,
t1Out,
dexed to

t112 reserve0, uint112 reserve1);

UM_LIQUIDITY() external pure returns (uint);
ry() external view returns (address);
0() external view returns (address);
1() external view returns (address);
serves() external view returns (uint112 reserve0, uint112 reserve1, uint32 blockTimestampLast);
0CumulativeLast() external view returns (uint);
1CumulativeLast() external view returns (uint);
() external view returns (uint);

address to) external returns (uint liquidity);
address to) external returns (uint amount0, uint amount1);
uint amount0Out, uint amount1Out, address to, bytes calldata data) external;
address to) external;
) external;
```

Contract Address
0x92682941480d4466017b32eFEA9f4c1ba3F2f0ad

Contract Creator
0x39c45a29ee35060dabce10ec73852f2b15ce77db

Source Code
Contract Source Code Verified

Contract Name
QubeSwapFactory

Other Settings
default evmVersion, MIT

Compiler Version
v0.5.16+commit.9c3226ce

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.

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

BEP-20 Contract Code Audit – Vulnerabilities Checked

Vulnerability Tested	AI Scan	Human Review	Result
Integer Overflow	Complete	Complete	✓ Low / No Risk
Integer Underflow	Complete	Complete	✓ Low / No Risk
Correct Token Standards Implementation	Complete	Complete	✓ Low / No Risk
Timestamp Dependency for Crucial Functions	Complete	Complete	✓ Low / No Risk
Exposed _Transfer Function	Complete	Complete	✓ Low / No Risk
Transaction-Ordering Dependency	Complete	Complete	✓ Low / No Risk
Unchecked Call Return Variable	Complete	Complete	✓ Low / No Risk
Use of Deprecated Functions	Complete	Complete	✓ Low / No Risk
Unprotected SELFDESTRUCT Instruction	Complete	Complete	✓ Low / No Risk
State Variable Default Visibility	Complete	Complete	✓ Low / No Risk
Deployer Can Access User Funds	Complete	Complete	✓ Low / No Risk

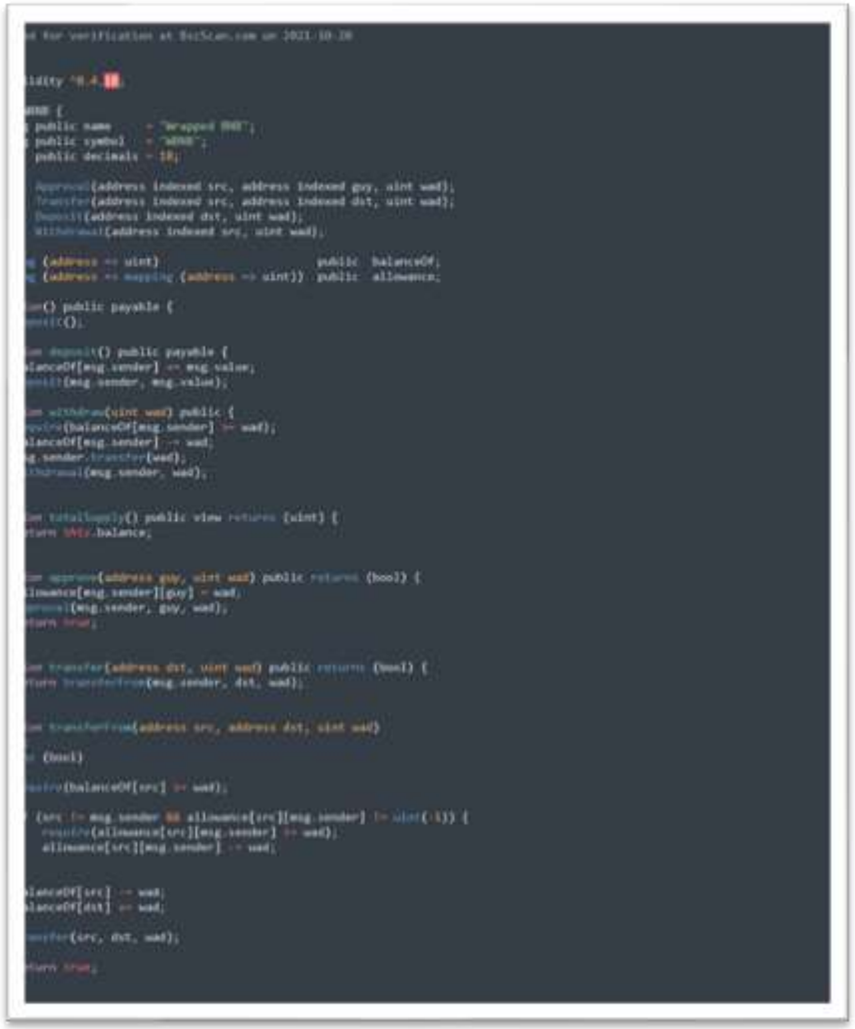
The contract code is **verified** on BSCScan.

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

QubeSwapFactory

Contract Code Audit – Overview

Dessert Finance was commissioned to perform an audit on WBNB



```
d for verification at BscScan.com on 2021-10-28

lidity ^0.4.18;

WBNB {
g public name     = "Wrapped BNB";
g public symbol   = "WBNB";
 public decimals = 18;

 Approval(address indexed src, address indexed guy, uint wad);
 Transfer(address indexed src, address indexed dst, uint wad);
 Deposit(address indexed dst, uint wad);
 Withdrawal(address indexed src, uint wad);

g (address => uint)                       public  balanceOf;
g (address => mapping (address => uint))  public  allowance;

ion() public payable {
posit();

ion deposit() public payable {
alanceOf[msg.sender] += msg.value;
posit(msg.sender, msg.value);

ion withdraw(uint wad) public {
quire(balanceOf[msg.sender] >= wad);
alanceOf[msg.sender] -= wad;
g.sender.transfer(wad);
ithdrawal(msg.sender, wad);


ion totalSupply() public view returns (uint) {
turn this.balance;


ion approve(address guy, uint wad) public returns (bool) {
llowance[msg.sender][guy] = wad;
proval(msg.sender, guy, wad);
turn true;


ion transfer(address dst, uint wad) public returns (bool) {
turn transferFrom(msg.sender, dst, wad);


ion transferFrom(address src, address dst, uint wad)

s (bool)

quire(balanceOf[src] >= wad);

f (src != msg.sender && allowance[src][msg.sender] != uint(-1)) {
  require(allowance[src][msg.sender] >= wad);
  allowance[src][msg.sender] -= wad;


alanceOf[src] -= wad;
alanceOf[dst] += wad;

ansfer(src, dst, wad);

turn true;
```

Contract Address
0x5dC557d1440010784EB482e414B07CE2143c1d38

Contract Creator
0x39c45a29ee35060dabce10ec73852f2b15ce77db

Source Code
Contract Source Code Verified

Contract Name
WBNB

Other Settings
default evmVersion, MIT

Compiler Version
v0.4.18+commit.9cf6e910

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.

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

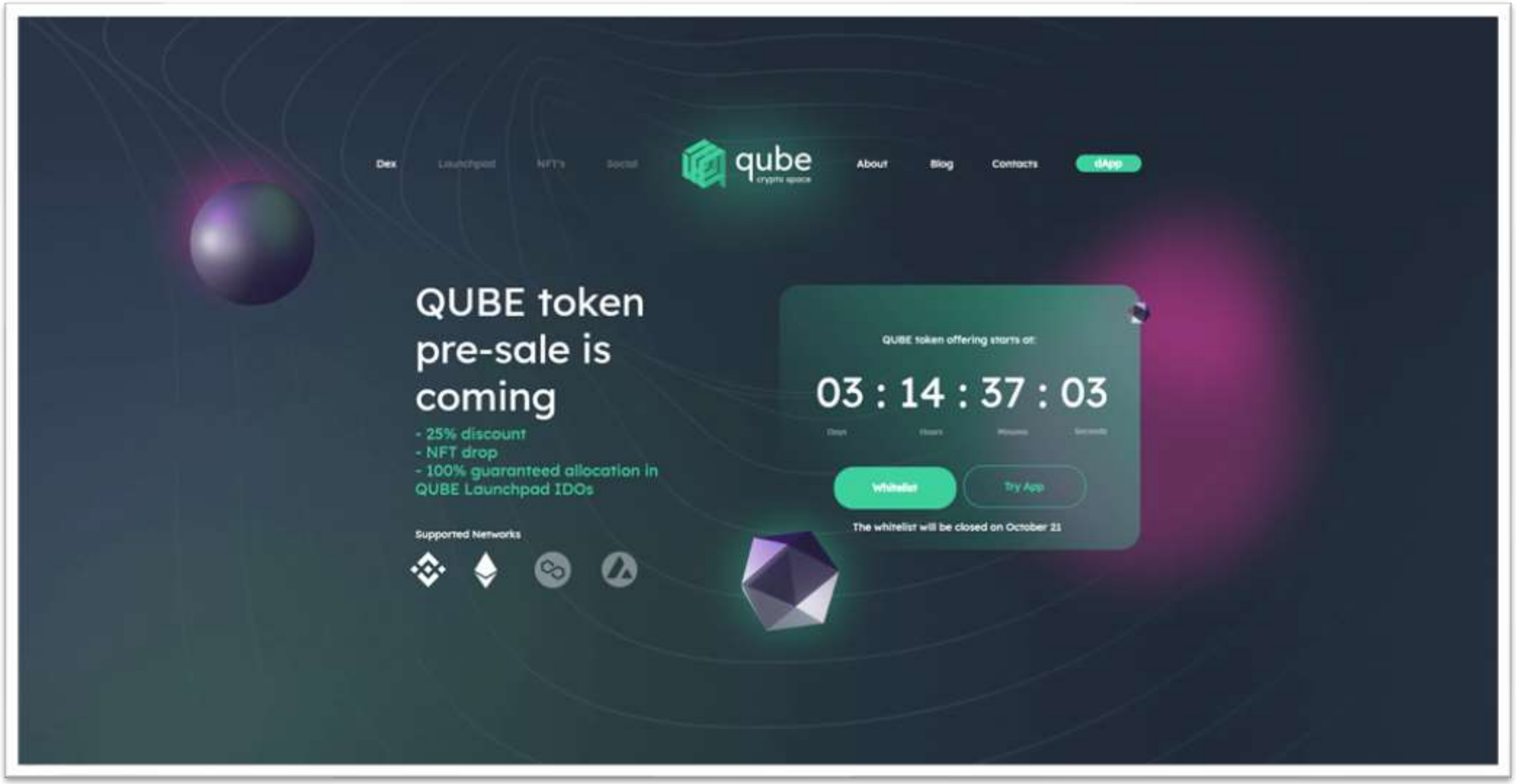
BEP-20 Contract Code Audit – Vulnerabilities Checked

Vulnerability Tested	AI Scan	Human Review	Result
Integer Overflow	Complete	Complete	✓ Low / No Risk
Integer Underflow	Complete	Complete	✓ Low / No Risk
Correct Token Standards Implementation	Complete	Complete	✓ Low / No Risk
Timestamp Dependency for Crucial Functions	Complete	Complete	✓ Low / No Risk
Exposed _Transfer Function	Complete	Complete	✓ Low / No Risk
Transaction-Ordering Dependency	Complete	Complete	✓ Low / No Risk
Unchecked Call Return Variable	Complete	Complete	✓ Low / No Risk
Use of Deprecated Functions	Complete	Complete	✓ Low / No Risk
Unprotected SELFDESTRUCT Instruction	Complete	Complete	✓ Low / No Risk
State Variable Default Visibility	Complete	Complete	✓ Low / No Risk
Deployer Can Access User Funds	Complete	Complete	✓ Low / No Risk

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

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

Website Part 1 – Overview
www.theqube.cc



QUBE token pre-sale is coming

- 25% discount
- NFT drop
- 100% guaranteed allocation in QUBE Launchpad IDOs

QUBE token offering starts at:

03 : 14 : 37 : 03

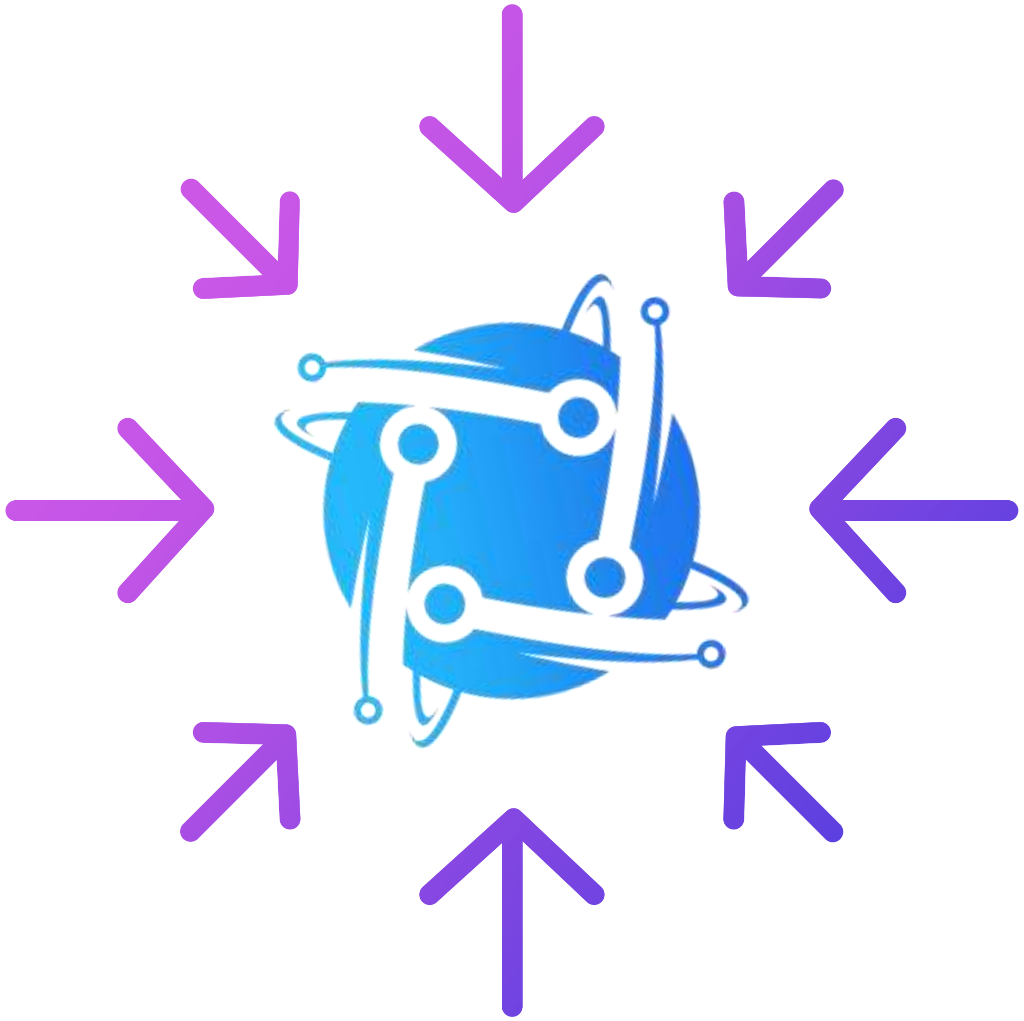
The whitelist will be closed on October 21

Above images are actual snapshots of the current live website of the project.

Website was registered on 03/19/2021, registration expires 03/19/2023.

✗ This does not meet the 3 year minimum we like to see on new projects.

Website Part 2 – Checklist



- ✓ Mobile Friendly
- ✗ No JavaScript Errors
- ✓ Spell Check
- ✓ SSL Certificate

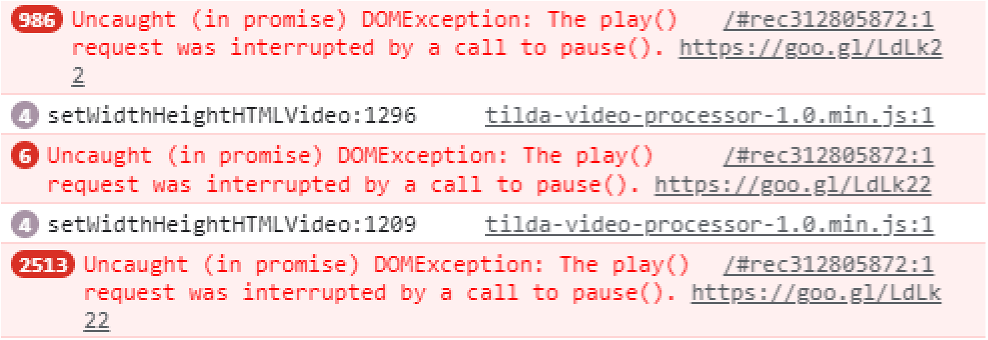
The website contained some JavaScript errors. No typos, or grammatical errors were present, and we found a valid SSL certificate allowing for access via https.

No additional issues were found on the website.

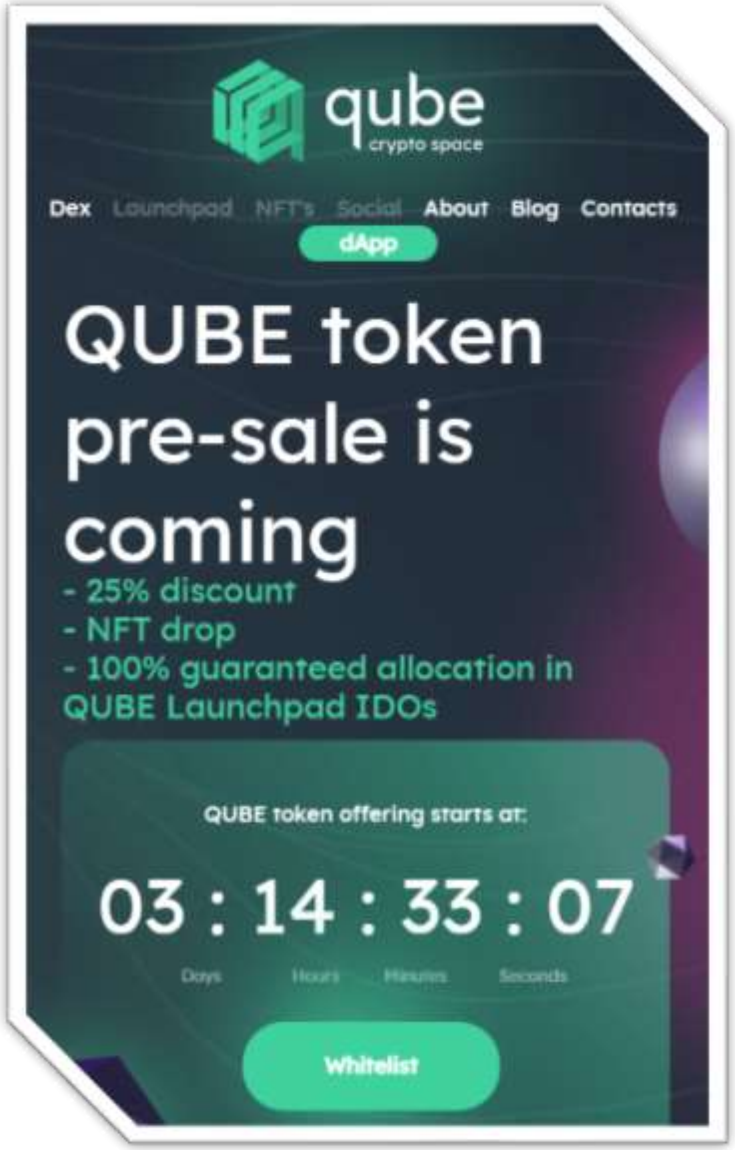
Website Part 3 – Responsive HTML5 & CSS3

No issues were found on the Mobile Friendly check for the website. All elements loaded properly and browser resize was not an issue. The team has put a considerable amount of thought and effort into making sure their website looks great on all screens.

Some JavaScript errors were found. No issues with loading elements, code, or stylesheets.



```
986 Uncaught (in promise) DOMException: The play()    /#rec312805872:1
    request was interrupted by a call to pause(). https://goo.gl/LdLk2
    2
4 setWidthHeightHTMLVideo:1296      tilda-video-processor-1.0.min.js:1
6 Uncaught (in promise) DOMException: The play()      /#rec312805872:1
  request was interrupted by a call to pause(). https://goo.gl/LdLk22
4 setWidthHeightHTMLVideo:1209      tilda-video-processor-1.0.min.js:1
2513 Uncaught (in promise) DOMException: The play()   /#rec312805872:1
     request was interrupted by a call to pause(). https://goo.gl/LdLk
     22
```



QUBE token pre-sale is coming
- 25% discount
- NFT drop
- 100% guaranteed allocation in QUBE Launchpad IDOs

QUBE token offering starts at:

03 : 14 : 33 : 07

Website Part 4 (GWS) – General Web Security

SSL CERTIFICATE

A valid SSL certificate was found. Details are as follows:

Offered to: theqube.cc

Issued by: R3

Valid Until: 01/04/2022

✓

CONTACT EMAIL

A valid contact email was found on the official website. Contact email is listed as shown below:

Contact

Contact Form

✓

SPAM / MALWARE / POPUPS

No malware found

No injected spam found

No internal server errors

No popups found

Domain is marked clean by Google, McAfee, Sucuri Labs, & ESET

✓

Social Media

We were able to locate a variety of Social Media networks for the project.

All links have been conveniently placed below.

Twitter

Telegram Ann.
Telegram

Reddit

Medium

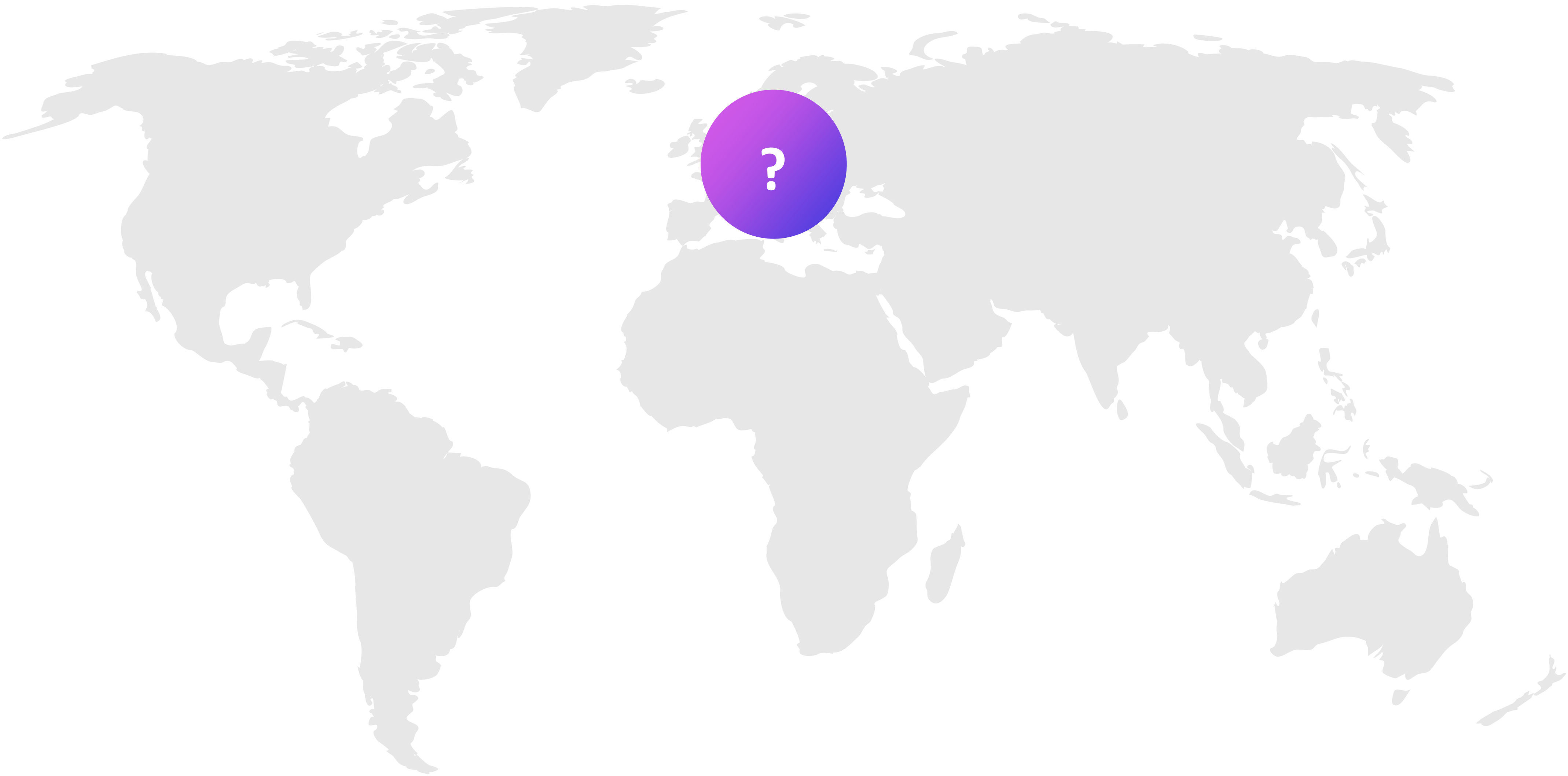
✓ **At least 3 social media networks were found.**

Top Token Holders

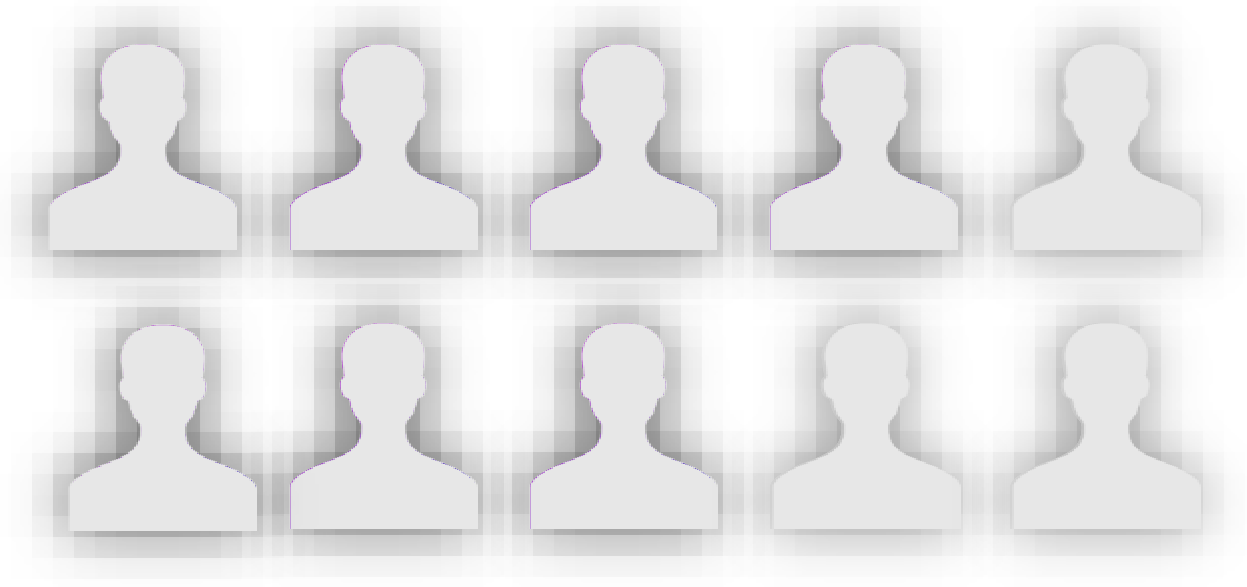
This project has not launched and is still on the Testnet. At the time of audit we are unable to provide top token holders because this project is not live on the Mainnet.

Location Audit

We were unable to identify a primary location for the project at this time or a location has not been declared.



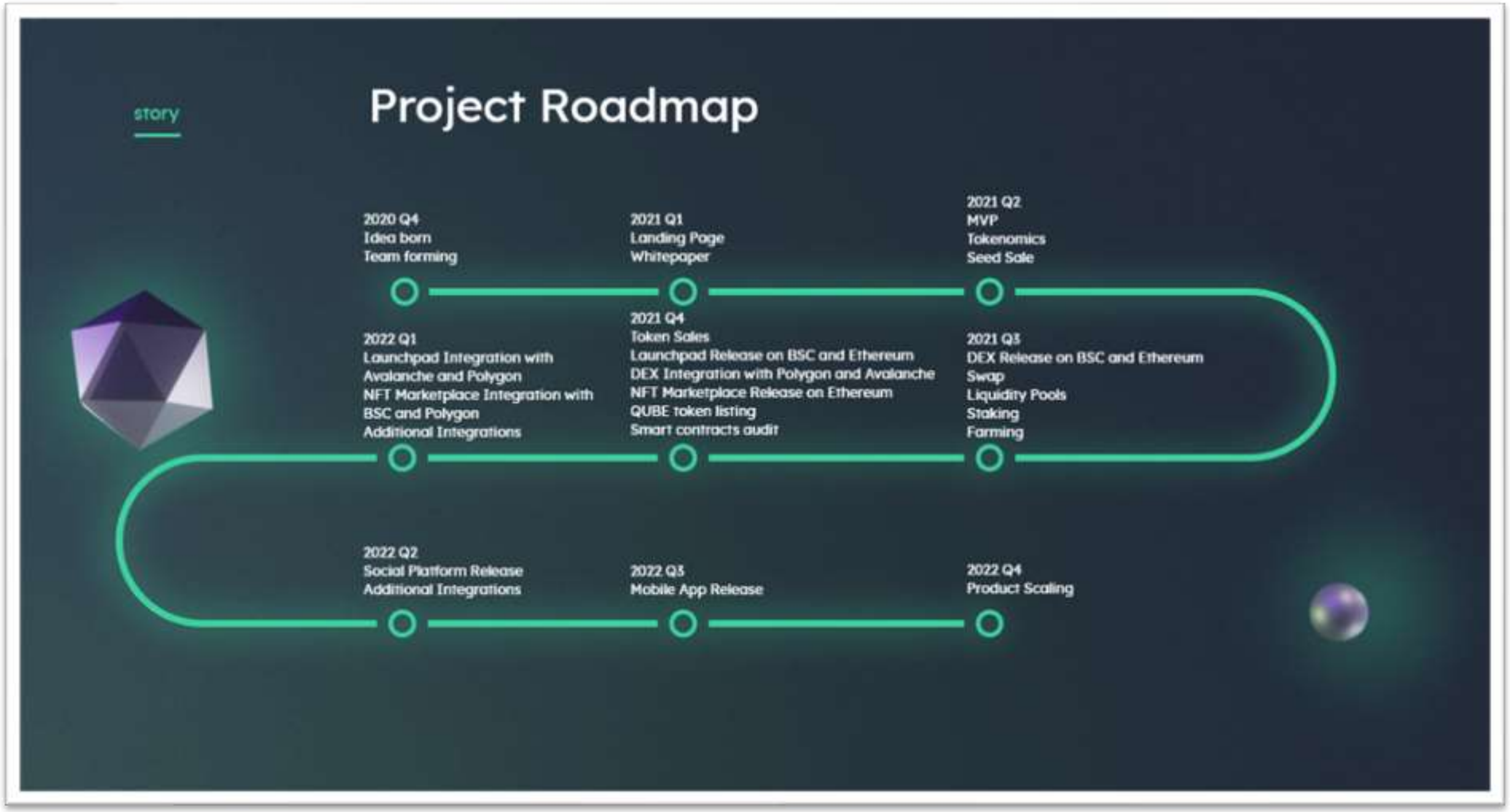
Team Overview



We are unable to find any information about the team on the website at this time. Projects may choose to stay anonymous for a myriad of reasons.

Roadmap

A roadmap was found on the official website, we have conveniently placed it on this page for your viewing.



story

Project Roadmap

2020 Q4
Idea born
Team forming

2021 Q1
Landing Page
Whitepaper

2021 Q2
MVP
Tokenomics
Seed Sale

2021 Q3
DEX Release on BSC and Ethereum
Swap
Liquidity Pools
Staking
Farming

2021 Q4
Token Sales
Launchpad Release on BSC and Ethereum
DEX Integration with Polygon and Avalanche
NFT Marketplace Release on Ethereum
QUBE token listing
Smart contracts audit

2022 Q1
Launchpad Integration with Avalanche and Polygon
NFT Marketplace Integration with BSC and Polygon
Additional Integrations

2022 Q2
Social Platform Release
Additional Integrations

2022 Q3
Mobile App Release

2022 Q4
Product Scaling

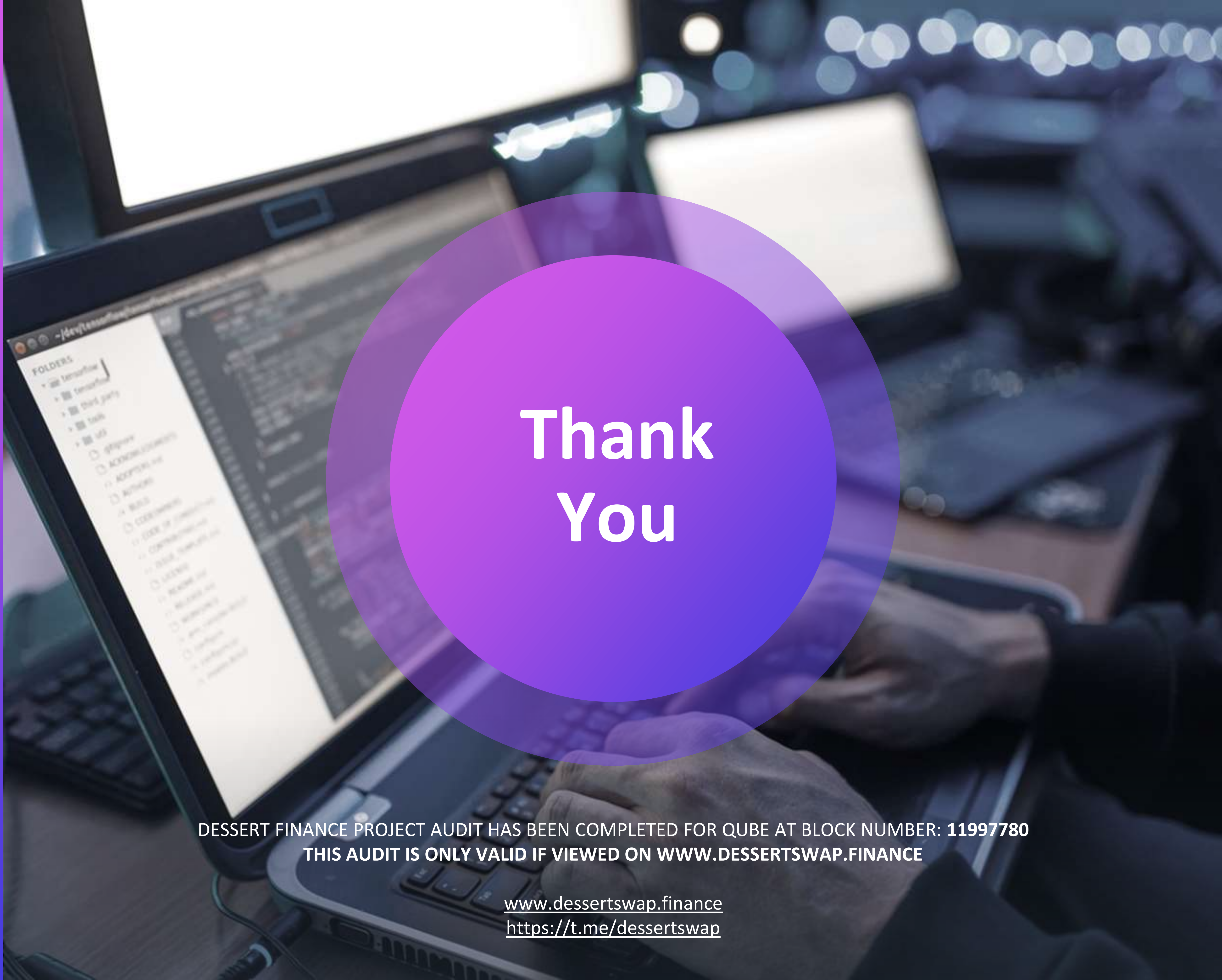
Disclaimer

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 BSC 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 PROJECT AUDIT HAS BEEN COMPLETED FOR QUBE AT BLOCK NUMBER: **11997780**
THIS AUDIT IS ONLY VALID IF VIEWED ON WWW.DESSERTSWAP.FINANCE

www.dessertswap.finance
https://t.me/dessertswap