SMART CONTRACT SECURITY AUDIT OF

BCNFT

Summary

Auditing Firm	Defense Network
Architecture	Defense "Echelon" Auditing Standard
Smart Contract Audit Approved By	Chris	Blockchain Specialist at Defense Network
Project Overview Approved By	Albert	Marketing Specialist at Defense Network
Platform	Solidity
Mandatory Audit Check	Static, Software, Auto Intelligent & Manual Analysis
Consultation Request Date	April 01, 2022
Report Date	April 07, 2022

Audit Summary

Defense team has performed a line-by-line manual analysis and automated review of the smart contract. The smart contract was analyzed mainly for common smart contract vulnerabilities, exploits, and manipulation hacks. According to the smart contract audit:

- ❖ **BCNFT's token smart contract source code has LOW RISK SEVERITY.**
- ❖ **BCNFT has PASSED the smart contract audit.**

For the detailed understanding of risk severity, source code vulnerability, and functional test, kindly refer to the audit.

✅ Verify the authenticity of this report on Defense's GitHub:
https://github.com/techloyaking/Defense-Network-

Table Of Contents

Defense
NETWORK

Project Overview

Defense was consulted by BCNFT to conduct the smart contract security audit of their token source code.

About BCNFT

The BILLIONAIRE CLUB is a private collection of 10 000 billionaire apes NFTs—unique digital collectibles. The apes are stored as ERC-721 tokens on the Ethereum blockchain and hosted on IPFS.

With more than 180+ hand drawn traits, each NFT is unique and comes with a membership to an exclusive group of successful investors. Join an ambitious ever-growing community with multiple benefits and utilities.

Earlier this year, a mysterious force coming from another galaxy summoned the most influential people on planet Earth. They received the geographic coordinates of a secret location for the party : the Billionaire Club. There, they will have the opportunity to exchange, discuss and collaborate in order to become even bigger than they are. During this party, the host will reveal its identity and the main reason for this invitation.

Project	BCNFT
Blockchain	Binance Smart Chain
Language	Solidity
Contract	0xE5677103191711bad85cf99E0dfC9B35449B790B
Website	https://bmnft.world
Telegram	https://t.me/btc100future
Twitter	@BCNFTGlobal

Defense
NETWORK

Public logo

Solidity Source Code On Blockchain (Verified Contract Source Code)

https://bscscan.com/address/0xe5677103191711bad85cf99e0dfc9b35449b790b#code

Contract Name: BCNFT

Symbol: STN

Compiler Version: v0.6.12

Optimization Enabled: Yes with 90 runs

Solidity Source Code On Defense GitHub

https://github.com/Defensenetwork/audited-codes/blob/main/BCNFT.sol

SHA- 1Hash

Solidity source code is audited at hash
0x406b22eb2ee60f27924b591803d90ad1601178b42cc2d15b8a9ee2a5f16d7cbd

Audit Scope & Methodology

The scope of this report is to audit the smart contract source code of BCNFT's token. Defense has scanned the contract and reviewed the project for common vulnerabilities, exploits, hacks, and back-doors. Below is the list of commonly known smart contract vulnerabilities, exploits, and hacks:

Category

Smart Contract Vulnerabilities

- ❖ Re-entrancy
- ❖ Unhandled Exceptions
- ❖ Transaction Order Dependency
- ❖ Integer Overflow
- ❖ Unrestricted Action
- ❖ Incorrect Inheritance Order
- ❖ Typographical Errors
- ❖ Requirement Violation

Source Code Review

- ❖ Ownership Takeover
- ❖ Gas Limit and Loops
- ❖ Deployment Consistency
- ❖ Repository Consistency
- ❖ Data Consistency
- ❖ Token Supply Manipulation

Functional Assessment

- ❖ Access Control and Authorization
- ❖ Operations Trail and Event Generation
- ❖ Assets Manipulation
- ❖ Liquidity Access

Defense's Echelon Audit Standard

The aim of Defense's "Echelon" standard is to analyze the smart contract and identify the vulnerabilities and the hacks in the smart contract. Mentioned are the steps used by ECHELON-1 to assess the smart contract:

1. Solidity smart contract source code reviewal:
   - ❖ Review of the specifications, sources, and instructions provided to Defense to make sure we understand the size, scope, and functionality of the smart contract.
   - ❖ Manual review of code, which is the process of reading source code line-byline to identify potential vulnerabilities.

2. Static, Manual, and Software analysis:
   - ❖ Test coverage analysis, which is the process of determining whether the test cases are covering the code and how much code is exercised when we run those test cases.
   - ❖ Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.

3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.

4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts

Automated 3P frameworks used to assess the smart contract vulnerabilities

- ❖ Slither
- ❖ Consensys MythX
- ❖ Consensys Surya
- ❖ Open Zeppelin Code Analyzer
- ❖ Solidity Code Complier

Defense's Risk Classification

Smart contracts are generally designed to manipulate and hold funds denominated in ETH/BNB. This makes them very tempting attack targets, as a successful attack may allow the attacker to directly steal funds from the contract. Below are the typical risk levels of a smart contract:

Vulnerable: A contract is vulnerable if it has been flagged by a static analysis tool as such. As we will see later, this means that some contracts may be vulnerable because of a false-positive.

Exploitable: A contract is exploitable if it is vulnerable and the vulnerability could be exploited by an external attacker. For example, if the "vulnerability" flagged by a tool is in a function which requires to own the contract, it would be vulnerable but not exploitable.

Exploited: A contract is exploited if it received a transaction on the main network which triggered one of its vulnerabilities. Therefore, a contract can be vulnerable or even exploitable without having been exploited.

Risk severity	Meaning
! Critical	This level vulnerabilities could be exploited easily, and can lead to asset loss, data loss, asset manipulation, or data manipulation. They should be fixed right away.
! High	This level vulnerabilities are hard to exploit but very important to fix, they carry an elevated risk of smart contract manipulation, which can lead to critical risk severity
! Medium	This level vulnerabilities are should be fixed, as they carry an inherent risk of future exploits, and hacks which may or may not impact the smart contract execution.
! Low	This level vulnerabilities can be ignored. They are code style violations, and informational statements in the code. They may not affect the smart contract execution

Smart Contract – Static Analysis

Symbol	Meaning
🔴	Function can be modified
💰	Function is payable
🔒	Function is locked
🔓	Function can be accessed
❗	Important functionality

```
| **ISupportingTokenInjection** | Interface |   |||
| └ | depositTokens | External ❗ | 🔴  |NO❗ |
| └ | burn | External ❗ | 🔴  |NO❗ |
||||||
| **IPancakeRouter01** | Interface |   |||
| └ | factory | External ❗ |    |NO❗ |
| └ | WETH | External ❗ |    |NO❗ |
| └ | addLiquidity | External ❗ | 🔴  |NO❗ |
| └ | addLiquidityETH | External ❗ | 💰 |NO❗ |
| └ | removeLiquidity | External ❗ | 🔴  |NO❗ |
| └ | removeLiquidityETH | External ❗ | 🔴  |NO❗ |
| └ | removeLiquidityWithPermit | External ❗ | 🔴  |NO❗ |
| └ | removeLiquidityETHWithPermit | External ❗ | 🔴  |NO❗ |
| └ | swapExactTokensForTokens | External ❗ | 🔴  |NO❗ |
| └ | swapTokensForExactTokens | External ❗ | 🔴  |NO❗ |
| └ | swapExactETHForTokens | External ❗ | 💰 |NO❗ |
| └ | swapTokensForExactETH | External ❗ | 🔴  |NO❗ |
| └ | swapExactTokensForETH | External ❗ | 🔴  |NO❗ |
| └ | swapETHForExactTokens | External ❗ | 💰 |NO❗ |
| └ | quote | External ❗ |    |NO❗ |
| └ | getAmountOut | External ❗ |    |NO❗ |
| └ | getAmountIn | External ❗ |    |NO❗ |
| └ | getAmountsOut | External ❗ |    |NO❗ |
| └ | getAmountsIn | External ❗ |    |NO❗ |
||||||
| **IPancakeRouter02** | Interface | IPancakeRouter01 |||
| └ | removeLiquidityETHSupportingFeeOnTransferTokens | External ❗| 🔴  |NO❗ |
| └ | removeLiquidityETHWithPermitSupportingFeeOnTransferTokens | External ❗| 🔴  |NO❗ |
| └ | swapExactTokensForTokensSupportingFeeOnTransferTokens | External ❗| 🔴  |NO❗ |
| └ | swapExactETHForTokensSupportingFeeOnTransferTokens | External ❗| 💰 |NO❗ |
| └ | swapExactTokensForETHSupportingFeeOnTransferTokens | External ❗| 🔴  |NO❗ |
```

Defense
N E T W O R K

```
||||||
| **IPancakeFactory** | Interface |   |||
| └ | feeTo | External ❗️ |   |NO❗️ |
| └ | feeToSetter | External ❗️ |   |NO❗️ |
| └ | getPair | External ❗️ |   |NO❗️ |
| └ | allPairs | External ❗️ |   |NO❗️ |
| └ | allPairsLength | External ❗️ |   |NO❗️ |
| └ | createPair | External ❗️ | 🔘 |NO❗️ |
| └ | setFeeTo | External ❗️ | 🔘 |NO❗️ |
| └ | setFeeToSetter | External ❗️ | 🔘 |NO❗️ |
||||||
| **IPancakePair** | Interface |   |||
| └ | name | External ❗️ |   |NO❗️ |
| └ | symbol | External ❗️ |   |NO❗️ |
| └ | decimals | External ❗️ |   |NO❗️ |
| └ | totalSupply | External ❗️ |   |NO❗️ |
| └ | balanceOf | External ❗️ |   |NO❗️ |
| └ | allowance | External ❗️ |   |NO❗️ |
| └ | approve | External ❗️ | 🔘 |NO❗️ |
| └ | transfer | External ❗️ | 🔘 |NO❗️ |
| └ | transferFrom | External ❗️ | 🔘 |NO❗️ |
| └ | DOMAIN_SEPARATOR | External ❗️ |   |NO❗️ |
| └ | PERMIT_TYPEHASH | External ❗️ |   |NO❗️ |
| └ | nonces | External ❗️ |   |NO❗️ |
| └ | permit | External ❗️ | 🔘 |NO❗️ |
| └ | MINIMUM_LIQUIDITY | External ❗️ |   |NO❗️ |
| └ | factory | External ❗️ |   |NO❗️ |
| └ | token0 | External ❗️ |   |NO❗️ |
| └ | token1 | External ❗️ |   |NO❗️ |
| └ | getReserves | External ❗️ |   |NO❗️ |
| └ | price0CumulativeLast | External ❗️ |   |NO❗️ |
| └ | price1CumulativeLast | External ❗️ |   |NO❗️ |
| └ | kLast | External ❗️ |   |NO❗️ |
| └ | mint | External ❗️ | 🔘 |NO❗️ |
| └ | burn | External ❗️ | 🔘 |NO❗️ |
| └ | swap | External ❗️ | 🔘 |NO❗️ |
| └ | skim | External ❗️ | 🔘 |NO❗️ |
| └ | sync | External ❗️ | 🔘 |NO❗️ |
| └ | initialize | External ❗️ | 🔘 |NO❗️ |
||||||
| **IBEP20** | Interface |   |||
| └ | totalSupply | External ❗️ |   |NO❗️ |
| └ | decimals | External ❗️ |   |NO❗️ |
| └ | symbol | External ❗️ |   |NO❗️ |
| └ | name | External ❗️ |   |NO❗️ |
| └ | getOwner | External ❗️ |   |NO❗️ |
| └ | balanceOf | External ❗️ |   |NO❗️ |
| └ | transfer | External ❗️ | 🔘 |NO❗️ |
| └ | allowance | External ❗️ |   |NO❗️ |
| └ | approve | External ❗️ | 🔘 |NO❗️ |
```

```
|  └ | transferFrom | External ‼ |  🔴  |NO‼ |
||||||
| **LockableFunction** | Implementation |   |||
||||||
| **IUserInfoManager** | Interface |   |||
||||||
| **IAuthorizedList** | Interface |   |||
|  └ | authorizeCaller | External ‼ |  🔴  |NO‼ |
||||||
| **IAuthorizedListExt** | Interface |   |||
|  └ | authorizeByAuthorized | External ‼ |  🔴  |NO‼ |
||||||
| **Context** | Implementation |   |||
|  └ | <Constructor> | Internal 🔒 |  🔴  | |
|  └ | _msgSender | Internal 🔒 |   | |
|  └ | _msgData | Internal 🔒 |   | |
||||||
| **Ownable** | Implementation | Context |||
|  └ | <Constructor> | Internal 🔒 |  🔴  | |
|  └ | owner | Public ‼ |   |NO‼ |
|  └ | renounceOwnership | Public ‼ |  🔴  | onlyOwner |
|  └ | transferOwnership | Public ‼ |  🔴  | onlyOwner |
|  └ | geUnlockTime | Public ‼ |   |NO‼ |
|  └ | lock | Public ‼ |  🔴  | onlyOwner |
|  └ | unlock | Public ‼ |  🔴  |NO‼ |
||||||
| **IBaseDistributor** | Interface |   |||
|  └ | deposit | External ‼ |  💰 |NO‼ |
|  └ | rewardCurrency | External ‼ |   |NO‼ |
||||||
| **ISmartLottery** | Interface | IBaseDistributor, IAuthorizedListExt, IAuthorizedList |||
|  └ | draw | External ‼ |   |NO‼ |
|  └ | jackpotAmount | External ‼ |   |NO‼ |
|  └ | isJackpotReady | External ‼ |   |NO‼ |
|  └ | setJackpot | External ‼ |  🔴  |NO‼ |
|  └ | checkAndPayJackpot | External ‼ |  🔴  |NO‼ |
|  └ | excludeFromJackpot | External ‼ |  🔴  |NO‼ |
|  └ | setMaxAttempts | External ‼ |  🔴  |NO‼ |
|  └ | setJackpotToCurrency | External ‼ |  🔴  |NO‼ |
|  └ | setJackpotToToken | External ‼ |  🔴  |NO‼ |
|  └ | setJackpotEligibilityCriteria | External ‼ |  🔴  |NO‼ |
|  └ | logTransfer | External ‼ |  🔴  |NO‼ |
||||||
| **Address** | Library |   |||
|  └ | isContract | Internal 🔒 |   | |
|  └ | sendValue | Internal 🔒 |  🔴  | |
|  └ | functionCall | Internal 🔒 |  🔴  | |
|  └ | functionCall | Internal 🔒 |  🔴  | |
|  └ | functionCallWithValue | Internal 🔒 |  🔴  | |
|  └ | functionCallWithValue | Internal 🔒 |  🔴  | |
```

```
| └ | _functionCallWithValue | Private 🔐 | 🔴 | | |
||||||
| **AuthorizedList** | Implementation | IAuthorizedList, Context, Ownable |||
| └ | <Constructor> | Public ❗ | 🔴 | Ownable |
| └ | authorizeCaller | External ❗ | 🔴 | onlyOwner |
| └ | renounceAuthorization | External ❗ | 🔴 | authorized |
||||||
| **AntiLPSniper** | Implementation | AuthorizedList |||
| └ | banHammer | Internal 🔒 | 🔴 | |
| └ | updateBlacklist | External ❗ | 🔴 | onlyOwner |
| └ | enableAntiSniper | External ❗ | 🔴 | authorized |
||||||
| **IBCNFTReflector** | Interface | IBaseDistributor, IAuthorizedListExt, IAuthorizedList |||
| └ | setDistributionCriteria | External ❗ | 🔴 |NO❗ |
| └ | setShare | External ❗ | 🔴 |NO❗ |
| └ | process | External ❗ | 🔴 |NO❗ |
| └ | setRewardToCurrency | External ❗ | 🔴 |NO❗ |
| └ | setRewardToToken | External ❗ | 🔴 |NO❗ |
| └ | excludeFromReward | External ❗ | 🔴 |NO❗ |
| └ | claimDividendFor | External ❗ | 🔴 |NO❗ |
||||||
| **SafeMath** | Library | |||
| └ | add | Internal 🔒 | | |
| └ | sub | Internal 🔒 | | |
| └ | sub | Internal 🔒 | | |
| └ | mul | Internal 🔒 | | |
| └ | div | Internal 🔒 | | |
| └ | div | Internal 🔒 | | |
||||||
| **LPSwapSupport** | Implementation | AuthorizedList |||
| └ | _approve | Internal 🔒 | 🔴 | |
| └ | updateRouter | Public ❗ | 🔴 | authorized |
| └ | updateLiquidityReceiver | External ❗ | 🔴 | onlyOwner |
| └ | updateRouterAndPair | Public ❗ | 🔴 | authorized |
| └ | updateLPPair | Public ❗ | 🔴 | authorized |
| └ | setSwapAndLiquifyEnabled | Public ❗ | 🔴 | authorized |
| └ | swapAndLiquify | Internal 🔒 | 🔴 | |
| └ | swapTokensForCurrency | Internal 🔒 | 🔴 | |
| └ | swapTokensForCurrencyAdv | Internal 🔒 | 🔴 | |
| └ | addLiquidity | Private 🔐 | 🔴 | |
| └ | swapCurrencyForTokens | Internal 🔒 | 🔴 | |
| └ | swapCurrencyForTokensAdv | Internal 🔒 | 🔴 | |
| └ | updateSwapRange | External ❗ | 🔴 | authorized |
||||||
| **SmartBuyback** | Implementation | AuthorizedList, LPSwapSupport |||
| └ | shouldBuyback | Public ❗ | |NO❗ |
| └ | doBuyback | Internal 🔒 | 🔴 | |
| └ | _doBuyback | Private 🔐 | 🔴 | lockTheSwap |
| └ | _doBuybackNoLimits | Private 🔐 | 🔴 | lockTheSwap |
| └ | simpleBuyback | Private 🔐 | 🔴 | |
```

Defense
NETWORK

```
| └ | dynamicBuyback | Private 🔐 | 🍴 | |
| └ | enableAutoBuybacks | External ‼️ | 🍴 | authorized |
| └ | updateBuybackSettings | External ‼️ | 🍴 | authorized |
| └ | updateBuybackLimits | External ‼️ | 🍴 | authorized |
| └ | forceBuyback | External ‼️ | 🍴 | authorized |
| └ | updateBuybackReceiver | External ‼️ | 🍴 | onlyOwner |
| | | | | | |
| **BCNFT** | Implementation | IBEP20, ISupportingTokenInjection, AntiLPSniper, LockableFunction, SmartBuyback | | |
| └ | <Constructor> | Public ‼️ | 🛑 | AuthorizedList |
| └ | init | External ‼️ | 🛑 | authorized |
| └ | <Fallback> | External ‼️ | 💵 |NO‼️ |
| └ | <Receive Ether> | External ‼️ | 💵 |NO‼️ |
| └ | name | Public ‼️ | |NO‼️ |
| └ | symbol | Public ‼️ | |NO‼️ |
| └ | decimals | Public ‼️ | |NO‼️ |
| └ | totalSupply | Public ‼️ | |NO‼️ |
| └ | balanceOf | Public ‼️ | |NO‼️ |
| └ | transfer | Public ‼️ | 🛑 |NO‼️ |
| └ | allowance | Public ‼️ | |NO‼️ |
| └ | approve | Public ‼️ | 🛑 |NO‼️ |
| └ | transferFrom | Public ‼️ | 🛑 |NO‼️ |
| └ | increaseAllowance | Public ‼️ | 🛑 |NO‼️ |
| └ | decreaseAllowance | Public ‼️ | 🛑 |NO‼️ |
| └ | setMarketingWallet | Public ‼️ | 🛑 | onlyOwner |
| └ | setGamingWallet | Public ‼️ | 🛑 | onlyOwner |
| └ | getOwner | External ‼️ | |NO‼️ |
| └ | updateGasForProcessing | Public ‼️ | 🛑 | authorized |
| └ | excludeFromFee | Public ‼️ | 🛑 | authorized |
| └ | _calculateFees | Private 🔐 | | |
| └ | _takeFees | Private 🔐 | 🛑 | |
| └ | updateTransferFees | External ‼️ | 🛑 | onlyOwner |
| └ | updateBuySellFees | External ‼️ | 🛑 | onlyOwner |
| └ | setTokenSwapThreshold | Public ‼️ | 🛑 | authorized |
| └ | burn | Public ‼️ | 🛑 |NO‼️ |
| └ | _burn | Private 🔐 | 🛑 | |
| └ | _approve | Internal 🛡️ | 🛑 | |
| | | | | |
| └ | authorizeCaller | External ‼️ | 🛑 | onlyOwner |
| └ | updateLPPair | Public ‼️ | 🛑 | authorized |
| └ | registerPairAddress | Public ‼️ | 🛑 | authorized |
| └ | _transferStandard | Private 🔐 | 🛑 | |
| └ | openTrading | External ‼️ | 🛑 | authorized |
| └ | updateReflectionContract | External ‼️ | 🛑 | authorized |
| └ | updateLotteryContract | External ‼️ | 🛑 | authorized |
| └ | excludeFromJackpot | External ‼️ | 🛑 | authorized |
| └ | excludeFromRewards | External ‼️ | 🛑 | authorized |
| └ | reflections | External ‼️ | |NO‼️ |
| └ | jackpot | External ‼️ | |NO‼️ |
| └ | depositTokens | External ‼️ | 🛑 |NO‼️ |
```

Defense NETWORK

Smart Contract – Software Analysis

Function Signatures

```
6279055  =>  isContract(address)
39509351 =>  increaseAllowance(address,uint256)
88889129 =>  updateBuySellFees(uint256,uint256,uint256,uint256,uint256,uint256,uint256)
4291ee3d =>  depositTokens(uint256,uint256,uint256,uint256)
42966c68 =>  burn(uint256)
c45a0155 =>  factory()
ad5c4648 =>  WETH()
e8e33700 =>  addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256)
f305d719 =>  addLiquidityETH(address,uint256,uint256,uint256,address,uint256)
baa2abde =>  removeLiquidity(address,address,uint256,uint256,uint256,address,uint256)
02751cec =>  removeLiquidityETH(address,uint256,uint256,uint256,address,uint256)
2195995c =>
removeLiquidityWithPermit(address,address,uint256,uint256,uint256,address,uint256,bool,uint8,bytes3
2,bytes32)
ded9382a =>
removeLiquidityETHWithPermit(address,uint256,uint256,uint256,address,uint256,bool,uint8,bytes32,byt
es32)
38ed1739 =>  swapExactTokensForTokens(uint256,uint256,address[],address,uint256)
8803dbee =>  swapTokensForExactTokens(uint256,uint256,address[],address,uint256)
7ff36ab5 =>  swapExactETHForTokens(uint256,address[],address,uint256)
4a25d94a =>  swapTokensForExactETH(uint256,uint256,address[],address,uint256)
18cbafe5 =>  swapExactTokensForETH(uint256,uint256,address[],address,uint256)
fb3bdb41 =>  swapETHForExactTokens(uint256,address[],address,uint256)
ad615dec =>  quote(uint256,uint256,uint256)
054d50d4 =>  getAmountOut(uint256,uint256,uint256)
85f8c259 =>  getAmountIn(uint256,uint256,uint256)
d06ca61f =>  getAmountsOut(uint256,address[])
1f00ca74 =>  getAmountsIn(uint256,address[])
af2979eb =>
removeLiquidityETHSupportingFeeOnTransferTokens(address,uint256,uint256,uint256,address,uint256)
5b0d5984 =>
removeLiquidityETHWithPermitSupportingFeeOnTransferTokens(address,uint256,uint256,uint256,address,u
int256,bool,uint8,bytes32,bytes32)
5c11d795 =>
swapExactTokensForTokensSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256)
b6f9de95 =>  swapExactETHForTokensSupportingFeeOnTransferTokens(uint256,address[],address,uint256)
791ac947 =>
swapExactTokensForETHSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256)
017e7e58 =>  feeTo()
094b7415 =>  feeToSetter()
e6a43905 =>  getPair(address,address)
1e3dd18b =>  allPairs(uint256)
574f2ba3 =>  allPairsLength()
c9c65396 =>  createPair(address,address)
f46901ed =>  setFeeTo(address)
```

Defense
NETWORK

```
a2e74af6  =>  setFeeToSetter(address)
06fdde03  =>  name()
95d89b41  =>  symbol()
313ce567  =>  decimals()
18160ddd  =>  totalSupply()
70a08231  =>  balanceOf(address)
dd62ed3e  =>  allowance(address,address)
095ea7b3  =>  approve(address,uint256)
a9059cbb  =>  transfer(address,uint256)
23b872dd  =>  transferFrom(address,address,uint256)
3644e515  =>  DOMAIN_SEPARATOR()
30adf81f  =>  PERMIT_TYPEHASH()
7ecebe00  =>  nonces(address)
d505accf  =>  permit(address,address,uint256,uint256,uint8,bytes32,bytes32)
ba9a7a56  =>  MINIMUM_LIQUIDITY()
0dfe1681  =>  token0()
d21220a7  =>  token1()
0902f1ac  =>  getReserves()
5909c0d5  =>  price0CumulativeLast()
5a3d5493  =>  price1CumulativeLast()
7464fc3d  =>  kLast()
6a627842  =>  mint(address)
89afcb44  =>  burn(address)
022c0d9f  =>  swap(uint256,uint256,address,bytes)
bc25cf77  =>  skim(address)
fff6cae9  =>  sync()
485cc955  =>  initialize(address,address)
893d20e8  =>  getOwner()
969cd8fe  =>  logTransfer(address,uint256,address,uint256)
fcc55e58  =>  authorizeCaller(address,bool)
5528ebd7  =>  authorizeByAuthorized(address)
119df25f  =>  _msgSender()
8b49d47e  =>  _msgData()
8da5cb5b  =>  owner()
715018a6  =>  renounceOwnership()
f2fde38b  =>  transferOwnership(address)
b6c52324  =>  geUnlockTime()
dd467064  =>  lock(uint256)
a69df4b5  =>  unlock()
d0e30db0  =>  deposit()
d6c7cc8b  =>  rewardCurrency()
0eecae21  =>  draw()
b1eac37e  =>  jackpotAmount()
d847b8e3  =>  isJackpotReady()
59992cc8  =>  setJackpot(uint256)
3d092c6d  =>  checkAndPayJackpot()
a2340af8  =>  excludeFromJackpot(address,bool)
992bfe27  =>  setMaxAttempts(uint256)
9d24dbd5  =>  setJackpotToCurrency(bool)
688491fc  =>  setJackpotToToken(address,bool)
```

Defense
NETWORK

```
a55a54e7  =>  setJackpotEligibilityCriteria(uint256,uint256,uint256)
24a084df  =>  sendValue(address,uint256)
a0b5ffb0  =>  functionCall(address,bytes)
241b5886  =>  functionCall(address,bytes,string)
2a011594  =>  functionCallWithValue(address,bytes,uint256)
d525ab8a  =>  functionCallWithValue(address,bytes,uint256,string)
36455e42  =>  _functionCallWithValue(address,bytes,uint256,string)
b8544dd7  =>  renounceAuthorization()
23ac4903  =>  banHammer(address)
9155e083  =>  updateBlacklist(address,bool)
6fbff064  =>  enableAntiSniper(bool)
2d48e896  =>  setDistributionCriteria(uint256,uint256)
14b6ca96  =>  setShare(address,uint256)
ffb2c479  =>  process(uint256)
8f7c9f8c  =>  setRewardToCurrency(bool)
b68a0d25  =>  setRewardToToken(address,bool)
35c6f9bc  =>  excludeFromReward(address,bool)
edf35253  =>  claimDividendFor(address)
```

Inheritance Graph
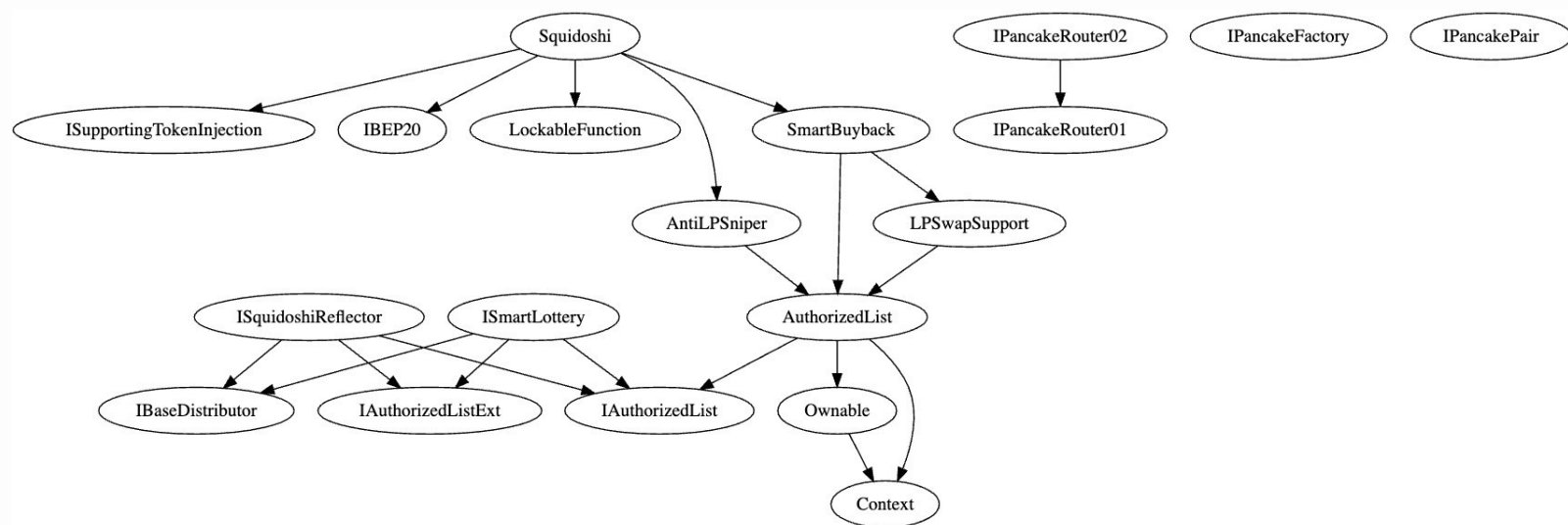
Smart Contract – Manual Analysis

Function	Description	Tested	Verdict
Total Supply	provides information about the total token supply	Yes	**Passed**
Balance Of	provides account balance of the owner's account	Yes	**Passed**
Transfer	executes transfers of a specified number of tokens to a specified address	Yes	**Passed**
Approve	allow a spender to withdraw a set number of tokens from a specified account	Yes	**Passed**
Allowance	returns a set number of tokens from a spender to the owner	Yes	**Passed**
Buy Back	is an action in which the project buys back its tokens from the existing holders usually at a market price	Yes	**Passed**
Burn	executes transfers of a specified number of tokens to a burn address	Yes	**Passed**
Mint	executes creation of a specified number of tokens and adds it to the total supply	NA	NA
Rebase	circulating token supply adjusts (increases or decreases) automatically according to a token's price fluctuations	NA	NA
Blacklist	stops specified wallets from interacting with the smart contract function modules	Yes	**Passed**
Lock	stops or locks all function modules of the smart contract	Yes	**Passed**

Defense
NETWORK

Review

❖ Active smart contract owner: **0x03E2427859119E497EB856a166F616a2Ce5f8c88**

❖ **Be aware that active smart contract owner privileges constitute an elevated impact to smart contract's safety and security.**

❖ **Smart contract can be locked by the owner. This stops or locks all function modules of the smart contract.**

❖ **Smart contract owner can blacklist certain wallets from interacting with the contract function modules.**

❖ **Smart contract owner can buy back the tokens from the total supply.**

❖ **Smart contract uses anti-snipe function module, this may raise the transaction tax, or block an address from doing a transaction.**

❖ Owner can-not lock or burn user assets.

❖ Owner can-not stop or pause the smart contract.

❖ Owner can-not mint tokens after launch.

❖ The smart contract utilizes "SafeMath" function to avoid common smart contract vulnerabilities.

```solidity
string private _name = "BCNFT";

library SafeMath {
function add(uint256 a, uint256 b) internal pure returns (uint256) {
    uint256 c = a + b;
    require(c >= a, "SafeMath: addition overflow");

function sub(uint256 a, uint256 b) internal pure returns (uint256) {
    return sub(a, b, "SafeMath: subtraction overflow");

    uint256 c = a * b;
    require(c / a == b, "SafeMath: multiplication overflow");

    return c;

function div(uint256 a, uint256 b) internal pure returns (uint256) {
    return div(a, b, "SafeMath: division by zero");

function mod(uint256 a, uint256 b) internal pure returns (uint256) {
    return mod(a, b, "SafeMath: modulo by zero");
```

❖ The smart contract has 1low severity issue which may or may not create any functional vulnerability.

{

    "resource": " /BCNFT.sol",

    "owner": "_generated_diagnostic_collection_name_#0",

    "severity": 8, (! Low Severity)

   **" Expected token Comma got 'Identifier"**,

    "source": "solc",

}

Smart Contract – SWC Attacks

SWC ID	Description	Verdict
SWC-101	Integer Overflow and Underflow	Passed
SWC-102	Outdated Compiler Version	! Low
SWC-103	Floating Pragma	Passed
SWC-104	Unchecked Call Return Value	Passed
SWC-105	Unprotected Ether Withdrawal	Passed
SWC-106	Unprotected SELFDESTRUCT Instruction	Passed
SWC-107	Re-entrancy	Passed
SWC-108	State Variable Default Visibility	Passed
SWC-109	Uninitialized Storage Pointer	Passed
SWC-110	Assert Violation	Passed
SWC-111	Use of Deprecated Solidity Functions	Passed
SWC-112	Delegate Call to Untrusted Callee	Passed
SWC-113	DoS with Failed Call	Passed
SWC-114	Transaction Order Dependence	Passed
SWC-115	Authorization through tx.origin	Passed
SWC-116	Block values as a proxy for time	Passed
SWC-117	Signature Malleability	Passed
SWC-118	Incorrect Constructor Name	Passed

Defense
N E T W O R K

SWC-119	Shadowing State Variables	Passed
SWC-120	Weak Sources of Randomness from Chain Attributes	Passed
SWC-121	Missing Protection against Signature Replay Attacks	Passed
SWC-122	Lack of Proper Signature Verification	Passed
SWC-123	Requirement Violation	Passed
SWC-124	Write to Arbitrary Storage Location	Passed
SWC-125	Incorrect Inheritance Order	! Low
SWC-126	Insufficient Gas Griefing	Passed
SWC-127	Arbitrary Jump with Function Type Variable	Passed
SWC-128	DoS With Block Gas Limit	Passed
SWC-129	Typographical Error	Passed
SWC-130	Right-To-Left-Override control character (U+202E)	Passed
SWC-131	Presence of unused variables	Passed
SWC-132	Unexpected Ether balance	Passed
SWC-133	Hash Collisions With Multiple Variable Length Arguments	Passed
SWC-134	Message call with hardcoded gas amount	Passed
SWC-135	Code With No Effects (Irrelevant/Dead Code)	Passed
SWC-136	Unencrypted Private Data On-Chain	Passed

Defense
NETWORK

Smart Contract - Risk Status & Radar Chart

Risk Severity	Status
! Critical	None critical severity issues identified
! High	None high severity issues identified
! Medium	None medium severity issues identified
! Low	**1 low severity issue identified**
Passed	**44 functions and instances verified and passed**

Score out of 100
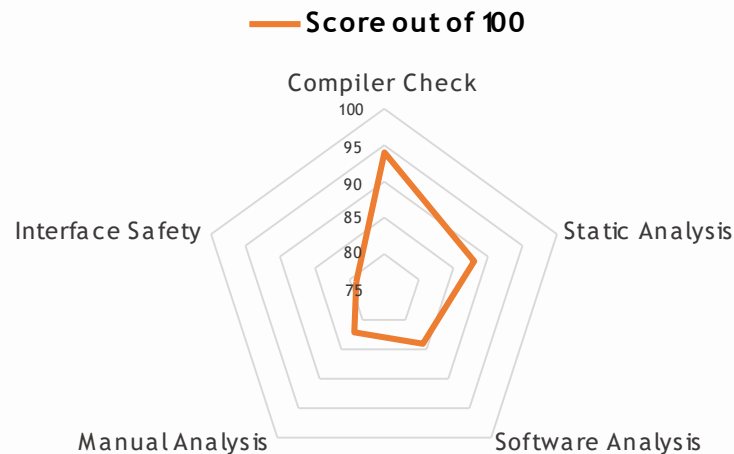
Compiler Check	94
Static Analysis	88
Software Analysis	84
Manual Analysis	82
Interface Safety	79

Defense
NETWORK

Auditor's Verdict

Defense team has performed a line-by-line manual analysis and automated review of the smart contract. The smart contract was analyzed mainly for common smart contract vulnerabilities, exploits, and manipulation hacks.

BCNFT's token smart contract source code has LOW RISK SEVERITY.

BCNFT has PASSED the smart contract audit.

Note for stakeholders

❖ Be aware that active smart contract owner privileges constitute an elevated impact on smart contract's safety and security.

❖ Make sure that the project team's KYC/identity is verified by an independent firm, e.g., Defense.

❖ Always check if the contract's liquidity is locked. A longer liquidity lock plays an important role in project's longevity. It is recommended to have multiple liquidity providers.

❖ Examine the unlocked token supply in the owner, developer, or team's private wallets. Understand the project's tokenomics, and make sure the tokens outside of the LP Pair are vested or locked for a longer period of time.

❖ Ensure that the project's official website is hosted on a trusted platform, and is using an active SSL certificate. The website's domain should be registered for a longer period of time.

Important Disclaimer

Defense Network provides contract auditing and project verification services for blockchain projects. The purpose of the audit is to analyse the on-chain smart contract source code, and to provide basic overview of the project. **This report should not be transmitted, disclosed, referred to, or relied upon by any person for any purposes without Defense's prior written consent.**

Defense provides the easy-to-understand assessment of the project, and the smart contract (otherwise known as the source code). The audit makes no statements or warranties on the security of the code. It also cannot be considered as an enough assessment regarding the utility and safety of the code, bug-free status, or any other statements of the contract. While we have used all the data at our disposal to provide the transparent analysis, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts. **Be aware that smart contracts deployed on a blockchain aren't resistant from external vulnerability, or a hack. Be aware that active smart contract owner privileges constitute an elevated impact to smart contract's safety and security. Therefore, Defense does not guarantee the explicit security of the audited smart contract.**

The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.

This report should not be considered as an endorsement or disapproval of any project or team. The information provided on this report does not constitute investment advice, financial advice, trading advice, or any other sort of advice and you should not treat any of the report's content as such. Do conduct your own due diligence and consult your financial advisor before making any investment decisions.

Defense
NETWORK

About Defense Network

Defense Network provides intelligent blockchain solutions. Defense is developing an ecosystem that is seamless and responsive. Some of our services: Blockchain Security, Token Launchpad, NFT Marketplace, etc. **Defense's mission is to interconnect multiple services like Blockchain Security, DeFi, Gaming, and Marketplace under one ecosystem that is seamless, multi-chain compatible, scalable, secure, fast, responsive, and easy-to-use.**

Defense is built by a decentralized team of UI experts, contributors, engineers, and enthusiasts from all over the world. Our team currently consists of 6+ core team members, and 10+ casual contributors. **Defense provides manual, static, and automatic smart contract analysis, to ensure that project is checked against known attacks and potential vulnerabilities.**

To learn more, visit www.deepsethi.com/

To view our audit portfolio, visit

https://github.com/techloyaking/Defense-Network-

To book an audit,