



# INFINITY

## BLOCKCHAIN SOLUTIONS

**Smart contract Audit**  
**Full detailed Report**



**[infinityblockchainsolutions.com](https://infinityblockchainsolutions.com)**



**[contact@infinityblockchainsolutions.com](mailto:contact@infinityblockchainsolutions.com)**



**[t.me/InfinityDev\\_77](https://t.me/InfinityDev_77)**



# Table of contents

<i>Contract Review .....</i>	<i>3</i>
<i>Audit Overview .....</i>	<i>4</i>
<i>Functions overview .....</i>	<i>4</i>
<i>Use of Dependencies .....</i>	<i>6</i>
<i>IBS Risk Analysis.....</i>	<i>6</i>
<i>Severity Definitions .....</i>	<i>7</i>
<i>Findings Break Down .....</i>	<i>8</i>
Centralization .....	8
Graphical Findings .....	10
Slither Findings Log .....	11
Solidity Static Analysis .....	12
Solhint Linter .....	15
<i>Final Summary .....</i>	<i>17</i>
<i>Disclaimer .....</i>	<i>18</i>
<i>About Infinity Blockchain Solutions .....</i>	<i>19</i>

# Contract Review

- BN.GAME is an ERC20-based standard smart contract deployed on the Ethereum blockchain.

Contract Name	BNGAME
Compiler Version	v0.8.24+commit.e11b9ed9
Optimization	Yes with 50 runs
Explorer Link	<a href="https://etherscan.io/address/0x3C43452707E95878CedfDeb24D5488c31cE669F2">https://etherscan.io/address/0x3C43452707E95878CedfDeb24D5488c31cE669F2</a>
Contract Address	0x3C43452707E95878CedfDeb24D5488c31cE669F2
Network	ETHEREUM
Symbol	\$BNG
Decimals	18
Supply	777,777,777 \$BNG
File Name	BNGAME.sol
Audit Date	Oct 18, 2024

## Audit Overview

This Audit's purpose was to assess any security issues, logic concerns, and potential improvements. After our audit assessment, we have labelled the security state of the contract of BN.GAME to be **"SECURED"**.

We have made the use of tools such as Solidity Static analysis, Remix IDE, Slither, Surya along with manual code analysis to inspect the token code.

The details of our findings are presented below.

## Functions overview

Sr.	Functions	Type	Observation	Result
1	constructor	write	Passed	Cleared
2.	name	read	Passed	Cleared
3.	symbol	read	Passed	Cleared
4.	decimals	read	Passed	Cleared
5.	balanceOf	Read	Passed	Cleared
6.	transfer	write	Passed	Cleared
7.	allowance	read	Passed	Cleared
8.	approve	write	Passed	Cleared
9.	transferFrom	write	Passed	Cleared

This Audit Report was created and compiled by Infinity Blockchain Solutions

Email: [contact@infinityblockchainsolutions.com](mailto:contact@infinityblockchainsolutions.com)

10.	_approve	private	Passed	Cleared
11.	_transfer	private	Passed	Cleared
12.	swapTokensForEth	private	Passed	Cleared
13.	sendETHToFee	private	Passed	Cleared
14.	min	read	Passed	Cleared
15.	manualSwap	write	Access only tax wallet	Cleared
16.	changeTaxWallet	write	Access only owner	Cleared
17.	whiteListFromFee	write	Access only owner	Cleared
18.	includeInFee	write	Access only owner	Cleared
19.	modifyTaxes	write	Access only owner	Cleared
20.	lockTheSwap	modifier	Passed	Cleared
21.	onlyOwner	modifier	Passed	Cleared
22.	owner	read	Passed	Cleared
23.	checkOwner	internal	Passed	Cleared
24.	renounceOwnership	write	Access only owner	Cleared
25.	transferOwnership	write	Access only owner	Cleared
26.	_transferOwnership	internal	Passed	Cleared










This Audit Report was created and compiled by Infinity Blockchain Solutions

Email: [contact@infinityblockchainsolutions.com](mailto:contact@infinityblockchainsolutions.com)

## Use of Dependencies

As per our observation, the libraries are used in this smart contract infrastructure that are on industry standard libraries like Openzeppelin.

## IBS Risk Analysis

Category	Result
 Buy Fee	7%
 Sell Fee	7%
 Cannot Buy	False
 Cannot Sell	False
 Maximum Tax Cap	25%
 Tax Modifiable?	Present
 Fee Check	Present
 Honeypot Issue?	Not Detected
 Trading Cooldown	Not Detected
 Trade Pausable?	No
 Transfer Pausable?	No
 Is it Anti-whale?	No

This Audit Report was created and compiled by Infinity Blockchain Solutions

Email: [contact@infinityblockchainsolutions.com](mailto:contact@infinityblockchainsolutions.com)

● Is Anti-bot?	Not Detected
● Can Addresses be Blacklisted?	Not Detected
● Blacklist Check	Passed
● Mint After deployment?	No
● Is it Proxy?	No
● Hidden Owner?	Not Detected
● Self-Destruction?	Not Detected

## Risk Analysis Result: **PASSED**

### Severity Definitions

Risk Level	Description
<b>Critical</b>	Critical vulnerabilities can lead to token loss etc.
<b>High</b>	Will definitely cause problems, this needs to be adjusted.
<b>Medium</b>	Will likely cause problems and it is recommended to adjust
<b>Low</b>	Won't cause any problems, but can be adjusted for improvement
<b>Informational</b>	Does not compromise the functionality of the contract in any way

This Audit Report was created and compiled by Infinity Blockchain Solutions

Email: [contact@infinityblockchainsolutions.com](mailto:contact@infinityblockchainsolutions.com)

# Findings Break Down

Risk Level	Unresolved	Acknowledged	Resolved
Critical	0	0	0
High	0	0	0
Medium	0	0	0
Low	0	0	0
Informational	1	1	0

## Centralization

Criticality	Informational
Status	Acknowledged

### Description:

This Audit Report was created and compiled by Infinity Blockchain Solutions

Email: [contact@infinityblockchainsolutions.com](mailto:contact@infinityblockchainsolutions.com)

This smart contract has some functions which can be executed by the Admin (Owner) only. If the owner wallet private key would be compromised, then it would create trouble.

### **Following are Owner functions:**

#### **Ownable.sol**

1. `renounceOwnership`: Deleting ownership will leave the contract without an owner, removing any owner-only functionality.
2. `transferOwnership`: The current owner can transfer ownership of the contract to a new account.

#### **BNGAME.sol:**

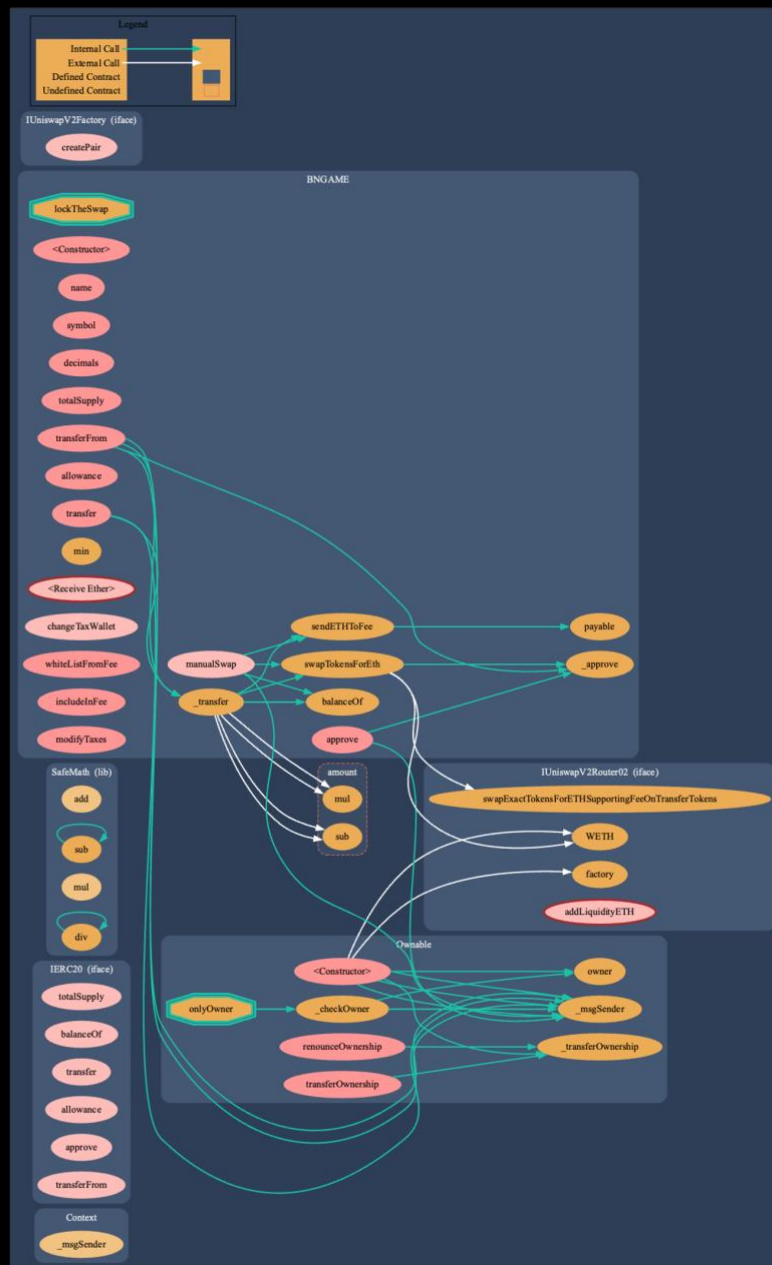
1. `changeTaxWallet`: Can be used to update Tax collection wallet address.
2. `whiteListFromFee`: Can be used to exclude addresses from fees.
3. `IncludeInFee`: Can be used to include addresses in fees.
4. `modifyTaxes`: Can be used to modify buy and sell fees.

### **Solution**

To make the smart contract 100% decentralized, we suggest renouncing ownership of the smart contract once its function is completed.

# Graphical Findings

## Call Graph Diagram – BNGAME.sol



This Audit Report was created and compiled by Infinity Blockchain Solutions

Email: [contact@infinityblockchainsolutions.com](mailto:contact@infinityblockchainsolutions.com)

# Slither Findings Log

**Slither** is a static analysis tool specifically designed for auditing Solidity smart contracts. It provides fast and comprehensive security analysis, helping developers and auditors identify potential vulnerabilities and bugs in smart contracts before deployment.

## Slither Log of BNGAME.sol:

```
BNGAME.sendETHToFee(uint256) (contracts/BNGAME.sol#365-370) sends eth to arbitrary user
  Dangerous calls:
  - (callSuccess) = address(_bnGameFeeCollector).call{value: amount}() (contracts/BNGAME.sol#366-368)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations

Reentrancy in BNGAME._transfer(address,address,uint256) (contracts/BNGAME.sol#306-342):
  External calls:
  - swapTokensForEth(contractTokenBalance) (contracts/BNGAME.sol#327)
  - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/BNGAME.sol#356-362)
  - sendETHToFee(address(this).balance) (contracts/BNGAME.sol#330)
  - (callSuccess) = address(_bnGameFeeCollector).call{value: amount}() (contracts/BNGAME.sol#366-368)
  External calls sending eth:
  - sendETHToFee(address(this).balance) (contracts/BNGAME.sol#330)
  - (callSuccess) = address(_bnGameFeeCollector).call{value: amount}() (contracts/BNGAME.sol#366-368)
  State variables written after the call(s):
  - _balances[address(this)] = _balances[address(this)].add(taxAmount) (contracts/BNGAME.sol#336)
  - _balances[from] = _balances[from].sub(amount) (contracts/BNGAME.sol#339)
  - _balances[to] = _balances[to].add(amount.sub(taxAmount)) (contracts/BNGAME.sol#340)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities

BNGAME.allowance(address,address).owner (contracts/BNGAME.sol#268) shadows:
  - Ownable.owner() (contracts/BNGAME.sol#112-114) (function)
BNGAME._approve(address,address,uint256).owner (contracts/BNGAME.sol#299) shadows:
  - Ownable.owner() (contracts/BNGAME.sol#112-114) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

BNGAME.modifyTaxes(uint256,uint256) (contracts/BNGAME.sol#398-403) should emit an event for:
  - _buyingFee = buyFee (contracts/BNGAME.sol#401)
  - _sellingFee = sellFee (contracts/BNGAME.sol#402)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
```

```
BNGAME.changeTaxWallet(address).newWallet (contracts/BNGAME.sol#386) lacks a zero-check on :
  - _bnGameFeeCollector = newWallet (contracts/BNGAME.sol#387)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in BNGAME.transferFrom(address,address,uint256) (contracts/BNGAME.sol#282-297):
  External calls:
  - _transfer(sender,recipient,amount) (contracts/BNGAME.sol#287)
  - (callSuccess) = address(_bnGameFeeCollector).call{value: amount}() (contracts/BNGAME.sol#366-368)
  - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/BNGAME.sol#356-362)
  External calls sending eth:
  - _transfer(sender,recipient,amount) (contracts/BNGAME.sol#287)
  - (callSuccess) = address(_bnGameFeeCollector).call{value: amount}() (contracts/BNGAME.sol#366-368)
  State variables written after the call(s):
  - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20: transfer amount exceeds allowance)) (contracts/BNGAME.sol#288-295)
  - _allowances[owner][spender] = amount (contracts/BNGAME.sol#302)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in BNGAME._transfer(address,address,uint256) (contracts/BNGAME.sol#306-342):
  External calls:
  - swapTokensForEth(contractTokenBalance) (contracts/BNGAME.sol#327)
  - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/BNGAME.sol#356-362)
  - sendETHToFee(address(this).balance) (contracts/BNGAME.sol#330)
  - (callSuccess) = address(_bnGameFeeCollector).call{value: amount}() (contracts/BNGAME.sol#366-368)
  External calls sending eth:
  - sendETHToFee(address(this).balance) (contracts/BNGAME.sol#330)
  - (callSuccess) = address(_bnGameFeeCollector).call{value: amount}() (contracts/BNGAME.sol#366-368)
  Event emitted after the call(s):
  - Transfer(from,address(this),taxAmount) (contracts/BNGAME.sol#337)
  - Transfer(from,to,amount.sub(taxAmount)) (contracts/BNGAME.sol#341)
```

This Audit Report was created and compiled by Infinity Blockchain Solutions

Email: [contact@infinityblockchainsolutions.com](mailto:contact@infinityblockchainsolutions.com)

# Solidity Static Analysis

Static code analysis is a technique used to detect common coding issues before the release of a program. It involves reviewing the code either manually or through the use of automated tools. These tools can scan the code without the need for execution, identifying potential problems in advance.

## BNGAME.sol

### Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in BNGAME.swapTokensForEth(uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

[more](#)

Pos: 282:4:

### Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.

[more](#)

Pos: 293:12:

### Low level calls:

Use of "call": should be avoided whenever possible. It can lead to unexpected behavior if return value is not handled properly. Please use Direct Calls via specifying the called contract's interface.

[more](#)

Pos: 298:31:

**Gas costs:**

Gas requirement of function BNGAME.symbol is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

Pos: 198:4:

**Gas costs:**

Gas requirement of function BNGAME.totalSupply is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

Pos: 206:4:

**Gas costs:**

Gas requirement of function BNGAME.transfer is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

Pos: 214:4:

**Constant/View/Pure functions:**

IERC20.approve(address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.

[more](#)

Pos: 15:4:

**Constant/View/Pure functions:**

IERC20.transferFrom(address,address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.

[more](#)

Pos: 16:4:

**No return:**

IERC20.transfer(address,uint256): Defines a return type but never explicitly returns a value.

Pos: 13:4:

**No return:**

IERC20.allowance(address,address): Defines a return type but never explicitly returns a value.

Pos: 14:4:

**Guard conditions:**

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

[more](#)

Pos: 33:8:

**Guard conditions:**

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

[more](#)

Pos: 43:8:

**Data truncated:**

Division of integer values yields an integer value again. That means e.g.  $10 / 100 = 0$  instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.

Pos: 43:16:

**Data truncated:**

Division of integer values yields an integer value again. That means e.g.  $10 / 100 = 0$  instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.

Pos: 53:20:

# Solhint Linter

Linters are utility tools designed to analyze source code and identify programming errors, bugs, and stylistic issues.

## BN.GAME.sol

Error message for require is too long  
Pos: 9:42

Explicitly mark visibility in function (Set ignoreConstructors to true if using solidity >=0.7.0)  
Pos: 5:66

Error message for require is too long  
Pos: 9:108

Function name must be in mixedCase  
Pos: 5:136

Constant name must be in capitalized SNAKE\_CASE  
Pos: 5:154

Constant name must be in capitalized SNAKE\_CASE  
Pos: 5:155

Constant name must be in capitalized SNAKE\_CASE  
Pos: 5:156

Constant name must be in capitalized SNAKE\_CASE  
Pos: 5:157

Explicitly mark visibility in function (Set ignoreConstructors to true if using solidity >=0.7.0)  
Pos: 5:175

Error message for require is too long  
Pos: 9:234

Error message for require is too long  
Pos: 9:235

Error message for require is too long  
Pos: 9:241

Error message for require is too long  
Pos: 9:242

Error message for require is too long  
Pos: 9:243

Avoid making time-based decisions in your business logic  
Pos: 13:292

Code contains empty blocks  
Pos: 32:302

Provide an error message for require  
Pos: 9:305

**Software analysis result:**

These tools reported many **false positive results** and some are informational issues.

So, those issues can be **safely ignored**.

Software analysis result: **PASSED**

## Final Summary

This audit investigated any possible issues inside BNGAME token contract. Our analysis reported no major issues or critical errors. One point to be noted is that the contract owner can access some functions but they cannot be used in a malicious way to disturb user's transactions.

Given that potential test cases for such smart contract protocols can be limitless, we cannot guarantee future outcomes. We have utilized the latest static tools and conducted thorough manual reviews to cover as many test cases as possible.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

As already stated above, we have concluded that the contract's security state is **"SECURED"**.

# Disclaimer

The information provided in this audit report is based on a thorough analysis conducted by **Infinity Blockchain Solutions**. This audit is intended to evaluate the security, functionality, and best practices of the token's smart contract code.

Our audit is limited to the analysis of the smart contract code provided to us. It does not cover vulnerabilities that may arise from external systems, third-party integrations, or the operational environment in which the token will be deployed.

While we employ industry-standard methodologies and tools, including both manual and automated processes, this audit cannot guarantee the complete absence of vulnerabilities. There may be undiscovered security flaws that were not identified during the review process.

The audit reflects the state of the token's smart contract at the time of review. We cannot predict or protect against future vulnerabilities or exploits that may emerge due to changes in the contract, its environment, or advancements in attack techniques.

Implementing the audit recommendations is the sole responsibility of the project team. Infinity Blockchain Solutions is not liable for any losses or damages resulting from the failure to address issues identified in the audit or any future vulnerabilities that arise post-audit.

This audit does not constitute financial advice. The evaluation is focused solely on the technical aspects of the smart contract. Investors and stakeholders should conduct their own independent research and due diligence before making any financial decisions related to the audited token.

Infinity Blockchain Solutions makes no warranties or representations, either express or implied, regarding the safety, reliability, or security of the audited smart contract. We are not responsible for any loss, damage, or legal issues that may arise from the use or misuse of the token.



# About Infinity Blockchain Solutions

Infinity Blockchain Solutions is a leading Web 3.0 development company dedicated to advancing the blockchain ecosystem. Founded with a mission to empower the future of digital innovation, Infinity Blockchain Solutions offers a comprehensive suite of services, including Crypto Token creation, Website development, ICO creation, smart contract audits and more.

With a reputation for excellence and a commitment to security, we have collaborated with numerous projects, contributing to the growth and integrity of the blockchain space. Our expert team provides reliable solutions that ensure the success and safety of our clients' ventures, securing their digital assets and fostering innovation in the decentralized landscape.



[infinityblockchainsolutions.com](https://infinityblockchainsolutions.com)