InterFi NETWORK

@INTERFINETWORK

SMART CONTRACT SECURITY AUDIT OF

NUTGAIN

SMART CONTRACT AUDIT | SOLIDITY DEVELOPMENT & TESTING | KYC | PROJECT EVALUATION

RELENTLESSLY SECURING THE PUBLIC BLOCKCHAIN

Summary

Auditing Firm	InterFi Network
Client Firm	NutGain
Architecture	InterFi "Echelon" Auditing Standard
Language	Solidity
Mandatory Audit Check	Static, Software, Auto Intelligent & Manual Analysis
Final Report Date	January 01, 2022

Audit Summary

InterFi 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:

- NutGain's smart contract source code has **MEDIUM RISK SEVERITY**
- NutGain's smart contract has an **ACTIVE OWNERSHIP**
- Important owner privileges – **LOCK CONTRACT, SET BUY & SELL FEES, SET SELL MULTIPLIER, SET MAX BUY – MAX SELL, SET MAX WALLET**

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

🚨 Contract address: **0x69dF29576044dfa9221b5c551D762b18De52252f**

⛓️ Blockchain: **Binance Smart Chain**

✅ Verify the authenticity of this report on InterFi's GitHub: **https://github.com/interfinetwork**

Table Of Contents

Project Overview

InterFi was consulted by NutGain to conduct the smart contract security audit of their solidity source code.

About NutGain

NutGain is a security protocol offering users and developers powerful tools including Web3, DeFi, Metaverse, NFT, Crypto Wallet, D-Hyper Ecommerce and dApps.

Project	NutGain
Blockchain	Binance Smart Chain
Language	Solidity
Contract	0x69dF29576044dfa9221b5c551D762b18De52252f
Website	https://www.nutgain.io
Telegram	https://t.me/Nutgaincommunity
Twitter	https://twitter.com/NutGainOfficial
Instagram	https://www.instagram.com/nutgainofficial/
Reddit	https://www.reddit.com/user/Nutgainofficial
Facebook	https://www.facebook.com/NutGain/
Linkedin	https://www.linkedin.com/company/nutgain

Project Logo

Solidity Source Code On Blockchain (Verified Contract Source Code)

https://bscscan.com/address/0x69dF29576044dfa9221b5c551D762b18De52252f#code

Contract Name: NUTG

Compiler Version: v0.8.7

Optimization Enabled: Yes with 200 runs

Solidity Source Code On InterFi GitHub

https://github.com/interfinetwork/audited-codes/blob/main/NutGain.sol

SHA-1 Hash

Solidity source code is audited at hash #8edc90ba305a3742df2223eac8bfa14037de9524

Audit Scope & Methodology

The scope of this report is to audit the smart contract source code of NutGain. InterFi 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 ❖ Ownership Takeover ❖ Gas Limit and Loops
Source Code Review	❖ Deployment Consistency ❖ Repository Consistency ❖ Data Consistency ❖ Token Supply Manipulation
Functional Assessment	❖ Access Control and Authorization ❖ Operations Trail and Event Generation ❖ Assets Manipulation ❖ Liquidity Access

InterFi's Echelon Audit Standard

The aim of InterFi'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 InterFi 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, Mythril
- ❖ SWC Registry
- ❖ Solidity Coverage
- ❖ Open Zeppelin Code Analyzer
- ❖ Solidity Code Complier

InterFi'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

Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
Ownable	Implementation	Context		
└	<Constructor>	Public !	🛑	NO!
└	owner	Public !		NO!
└	renounceOwnership	Public !	🛑	onlyOwner
└	transferOwnership	Public !	🛑	onlyOwner
└	lock	Public !	🛑	onlyOwner
INutGain	Interface			
└	totalSupply	External !		NO!
└	balanceOf	External !		NO!
└	transfer	External !	🛑	NO!
└	allowance	External !		NO!
└	approve	External !	🛑	NO!
└	transferFrom	External !	🛑	NO!
NutGain	Implementation	Context, INutGain		
└	<Constructor>	Public !	🛑	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❗
└	_transfer	Internal 🔒	🛑	
└	_mint	Internal 🔒	🛑	
└	_burn	Internal 🔒	🛑	
└	_approve	Internal 🔒	🛑	
└	_setupDecimals	Internal 🔒	🛑	
└	_beforeTokenTransfer	Internal 🔒	🛑	
IDividendPayingToken	Interface				
└	dividendOf	External ❗		NO❗
└	withdrawDividend	External ❗	🛑	NO❗
IDividendPayingTokenOptional	Interface				
└	withdrawableDividendOf	External ❗		NO❗
└	withdrawnDividendOf	External ❗		NO❗
└	accumulativeDividendOf	External ❗		NO❗
DividendPayingToken	Implementation	NutGain, IDividendPayingToken, IDividendPayingTokenOptional, Ownable			
└	<Constructor>	Public ❗	🛑	NutGain
└	<Receive Ether>	External ❗	💵	NO❗
└	distributeDividends	Public ❗	🛑	onlyOwner
└	withdrawDividend	Public ❗	🛑	NO❗
└	setDividendTokenAddress	External ❗	🛑	NO❗
└	_withdrawDividendOfUser	Internal 🔒	🛑	
└	dividendOf	Public ❗		NO❗
└	withdrawableDividendOf	Public ❗		NO❗
└	withdrawnDividendOf	Public ❗		NO❗
└	accumulativeDividendOf	Public ❗		NO❗
└	_transfer	Internal 🔒	🛑	
└	_mint	Internal 🔒	🛑	
└	_burn	Internal 🔒	🛑	
└	_setBalance	Internal 🔒	🛑	
IUniswapV2Factory	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❗
IUniswapV2Pair	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❗
IUniswapV2Router01	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❗
IterableMapping	Library				
└	get	Public ❗		NO❗
└	getIndexOfKey	Public ❗		NO❗
└	getKeyAtIndex	Public ❗		NO❗
└	size	Public ❗		NO❗
└	set	Public ❗	🛑	NO❗

└	remove	Public ❗	🛑	NO❗
SafeMath	Library				
└	tryAdd	Internal 🔒			
└	trySub	Internal 🔒			
└	tryMul	Internal 🔒			
└	tryDiv	Internal 🔒			
└	tryMod	Internal 🔒			
└	add	Internal 🔒			
└	sub	Internal 🔒			
└	mul	Internal 🔒			
└	div	Internal 🔒			
└	mod	Internal 🔒			
└	sub	Internal 🔒			
└	div	Internal 🔒			
└	mod	Internal 🔒			
SafeMathInt	Library				
└	mul	Internal 🔒			
└	div	Internal 🔒			
└	sub	Internal 🔒			
└	add	Internal 🔒			
└	toUint256Safe	Internal 🔒			
SafeMathUint	Library				
└	toInt256Safe	Internal 🔒			
NUTG	Implementation	NutGain, Ownable			
└	<Constructor>	Public ❗	🛑	NutGain
└	<Receive Ether>	External ❗	💵	NO❗
└	whitelistPresale	External ❗	🛑	onlyOwner
└	prepareForPartnerOrExchangeListing	External ❗	🛑	onlyOwner
└	setMaxBuyTransaction	External ❗	🛑	onlyOwner
└	setLpReceiver	External ❗	🛑	onlyOwner
└	setMaxSellTransaction	External ❗	🛑	onlyOwner
└	updatenutgDividentToken	External ❗	🛑	onlyOwner
└	updateDevelopmentWallet	External ❗	🛑	onlyOwner
└	setMaxWalletToken	External ❗	🛑	onlyOwner
└	setSwapTokensAtAmount	External ❗	🛑	onlyOwner
└	setSellTransactionMultiplier	External ❗	🛑	onlyOwner
└	afterPreSale	External ❗	🛑	onlyOwner
└	setTradingIsEnabled	External ❗	🛑	onlyOwner
└	setBuyBackAndLiquifyEnabled	External ❗	🛑	onlyOwner
└	setnutgDividendEnabled	External ❗	🛑	onlyOwner
└	setDevelopmentEnabled	External ❗	🛑	onlyOwner
└	updatenutgDividendTracker	External ❗	🛑	onlyOwner
└	setBuyFee	External ❗	🛑	onlyOwner
└	setSellFee	External ❗	🛑	onlyOwner
└	setTransferFee	External ❗	🛑	onlyOwner
└	updateUniswapV2Router	External ❗	🛑	onlyOwner

└	setExcludeFromFees	Public ❗	🛑	onlyOwner
└	excludeFromDividend	Public ❗	🛑	onlyOwner
└	excludeMultipleAccountsFromFees	External ❗	🛑	onlyOwner
└	setAutomatedMarketMakerPair	Public ❗	🛑	onlyOwner
└	_setAutomatedMarketMakerPair	Private 🔐	🛑	onlyOwner
└	updateGasForProcessing	External ❗	🛑	onlyOwner
└	updateMinimumBalanceForDividends	External ❗	🛑	onlyOwner
└	updateClaimWait	External ❗	🛑	onlyOwner
└	getEthClaimWait	External ❗		NO❗
└	getTotalnutgDividendsDistributed	External ❗		NO❗
└	getIsExcludedFromFees	Public ❗		NO❗
└	withdrawablenutgDividendOf	External ❗		NO❗
└	nutgDividentTokenBalanceOf	External ❗		NO❗
└	getAccountnutgDividendsInfo	External ❗		NO❗
└	getAccountnutgDividendsInfoAtIndex	External ❗		NO❗
└	processDividendTracker	External ❗	🛑	onlyOwner
└	rand	Internal 🔒		
└	claim	External ❗	🛑	NO❗
└	getLastnutgDividendProcessedIndex	External ❗		NO❗
└	getNumberOfnutgDividentTokenHolders	External ❗		NO❗
└	getAllFees	Private 🔐		
└	_transfer	Internal 🔒	🛑	
└	swapAndLiquify	Private 🔐	🛑	
└	addLiquidity	Private 🔐	🛑	
└	buyBackAndBurn	Private 🔐	🛑	
└	swapTokensForBNB	Private 🔐	🛑	
└	swapTokensForDividendToken	Private 🔐	🛑	
└	swapAndSendnutgDividends	Private 🔐	🛑	
└	transferToWallet	Private 🔐	🛑	
└	transferDividends	Private 🔐	🛑	
NutgDividendTracker	Implementation	Ownable, DividendPayingToken		
└	<Constructor>	Public ❗	🛑	DividendPayingToken
└	_transfer	Internal 🔒		
└	withdrawDividend	Public ❗		NO❗
└	setDividendTokenAddress	External ❗	🛑	onlyOwner
└	updateMinimumTokenBalanceForDividends	External ❗	🛑	onlyOwner
└	excludeFromDividends	External ❗	🛑	onlyOwner
└	includeFromDividens	External ❗	🛑	onlyOwner
└	updateClaimWait	External ❗	🛑	onlyOwner
└	getLastProcessedIndex	External ❗		NO❗
└	getNumberOfTokenHolders	External ❗		NO❗
└	getAccount	Public ❗		NO❗
└	getAccountAtIndex	Public ❗		NO❗
└	canAutoClaim	Private 🔐		
└	setBalance	External ❗	🛑	onlyOwner
└	process	Public ❗	🛑	NO❗
└	processAccount	Public ❗	🛑	onlyOwner

Smart Contract – Software Analysis

Function Signatures

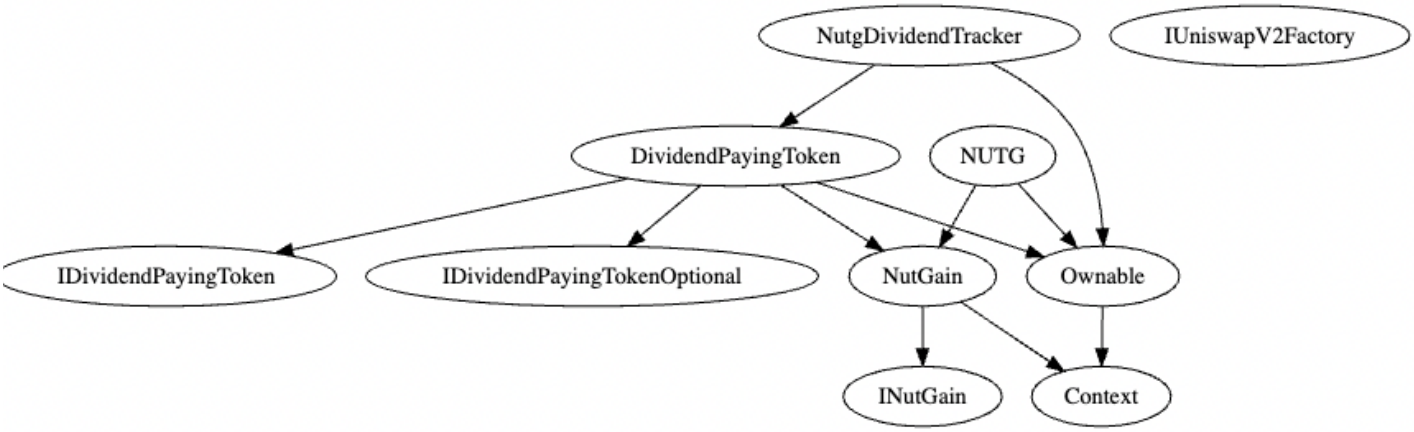
```
12564034  =>  transferDividends(address,address,DividendPayingToken,uint256)
39509351  =>  increaseAllowance(address,uint256)
43509138  =>  div(int256,int256)
67169090  =>  transferToWallet(address,uint256)
119df25f  =>  _msgSender()
8b49d47e  =>  _msgData()
8da5cb5b  =>  owner()
715018a6  =>  renounceOwnership()
f2fde38b  =>  transferOwnership(address)
dd467064  =>  lock(uint256)
18160ddd  =>  totalSupply()
70a08231  =>  balanceOf(address)
a9059cbb  =>  transfer(address,uint256)
dd62ed3e  =>  allowance(address,address)
095ea7b3  =>  approve(address,uint256)
23b872dd  =>  transferFrom(address,address,uint256)
06fdde03  =>  name()
95d89b41  =>  symbol()
313ce567  =>  decimals()
a457c2d7  =>  decreaseAllowance(address,uint256)
30e0789e  =>  _transfer(address,address,uint256)
4e6ec247  =>  _mint(address,uint256)
6161eb18  =>  _burn(address,uint256)
104e81ff  =>  _approve(address,address,uint256)
61e9edb2  =>  _setupDecimals(uint8)
cad3be83  =>  _beforeTokenTransfer(address,address,uint256)
91b89fba  =>  dividendOf(address)
6a474002  =>  withdrawDividend()
a8b9d240  =>  withdrawableDividendOf(address)
aafd847a  =>  withdrawnDividendOf(address)
27ce0147  =>  accumulativeDividendOf(address)
3243c791  =>  distributeDividends(uint256)
7e3e7fd2  =>  setDividendTokenAddress(address)
373de4aa  =>  _withdrawDividendOfUser(address)
ab86e0a6  =>  _setBalance(address,uint256)
017e7e58  =>  feeTo()
094b7415  =>  feeToSetter()
e6a43905  =>  getPair(address,address)
1e3dd18b  =>  allPairs(uint256)
574f2ba3  =>  allPairsLength()
c9c65396  =>  createPair(address,address)
f46901ed  =>  setFeeTo(address)
a2e74af6  =>  setFeeToSetter(address)
3644e515  =>  DOMAIN_SEPARATOR()
30adf81f  =>  PERMIT_TYPEHASH()
```


```
7ecebe00  =>  nonces(address)
d505accf  =>  permit(address,address,uint256,uint256,uint8,bytes32,bytes32)
ba9a7a56  =>  MINIMUM_LIQUIDITY()
c45a0155  =>  factory()
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)
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)
268d8e2e  =>  get(Map,address)
b45dad3d  =>  getIndexOfKey(Map,address)
```


```
7596720f  =>  getKeyAtIndex(Map,uint256)
b1b533f3  =>  size(Map)
6b06f325  =>  set(Map,address,uint256)
0eac8729  =>  remove(Map,address)
884557bf  =>  tryAdd(uint256,uint256)
a29962b1  =>  trySub(uint256,uint256)
6281efa4  =>  tryMul(uint256,uint256)
736ecb18  =>  tryDiv(uint256,uint256)
38dc0867  =>  tryMod(uint256,uint256)
771602f7  =>  add(uint256,uint256)
b67d77c5  =>  sub(uint256,uint256)
c8a4ac9c  =>  mul(uint256,uint256)
a391c15b  =>  div(uint256,uint256)
f43f523a  =>  mod(uint256,uint256)
e31bdc0a  =>  sub(uint256,uint256,string)
b745d336  =>  div(uint256,uint256,string)
71af23e8  =>  mod(uint256,uint256,string)
bbe93d91  =>  mul(int256,int256)
adefc37b  =>  sub(int256,int256)
a5f3c23b  =>  add(int256,int256)
744f7c7d  =>  toUint256Safe(int256)
e823b9bf  =>  toInt256Safe(uint256)
fdfbf73a  =>  whitelistPresale(address,address)
3395155e  =>  prepareForPartnerOrExchangeListing(address)
b3b5e043  =>  setMaxBuyTransaction(uint256)
ad8f3467  =>  setLpReceiver(address)
5c38ffe2  =>  setMaxSellTransaction(uint256)
0cf54a28  =>  updatenutgDividentToken(address)
f023f573  =>  updateDevelopmentWallet(address)
91d55f41  =>  setMaxWalletToken(uint256)
afa4f3b2  =>  setSwapTokensAtAmount(uint256)
fb7f634a  =>  setSellTransactionMultiplier(uint256)
07efbfdc  =>  afterPreSale()
e27ad5eb  =>  setTradingIsEnabled(bool)
cf8f19bf  =>  setBuyBackAndLiquifyEnabled(bool)
9f4d4de1  =>  setnutgDividendEnabled(bool)
8bfca5d5  =>  setDevelopmentEnabled(bool)
29da6105  =>  updatenutgDividendTracker(address)
25519cf2  =>  setBuyFee(uint256,uint256,uint256)
1d865c30  =>  setSellFee(uint256,uint256,uint256)
e7c89232  =>  setTransferFee(uint256,uint256,uint256)
65b8dbc0  =>  updateUniswapV2Router(address)
d63cad22  =>  setExcludeFromFees(address,bool)
b13f725d  =>  excludeFromDividend(address)
c492f046  =>  excludeMultipleAccountsFromFees(address[],bool)
9a7a23d6  =>  setAutomatedMarketMakerPair(address,bool)
a7f7b36f  =>  _setAutomatedMarketMakerPair(address,bool)
871c128d  =>  updateGasForProcessing(uint256)
3b364da8  =>  updateMinimumBalanceForDividends(uint256)
e98030c7  =>  updateClaimWait(uint256)
```


```
b1127452  =>  getEthClaimWait()
177ede8d  =>  getTotalnutgDividendsDistributed()
7b16cea0  =>  getIsExcludedFromFees(address)
6c59a207  =>  withdrawablenutgDividendOf(address)
9c96dc06  =>  nutgDividentTokenBalanceOf(address)
528a0d04  =>  getAccountnutgDividendsInfo(address)
315c2e47  =>  getAccountnutgDividendsInfoAtIndex(uint256)
700bb191  =>  processDividendTracker(uint256)
3b3dca76  =>  rand()
4e71d92d  =>  claim()
cac5a84b  =>  getLastnutgDividendProcessedIndex()
8cc644c4  =>  getNumberOfnutgDividentTokenHolders()
819a37d8  =>  getAllFees(bool,bool)
173865ad  =>  swapAndLiquify(uint256)
9cd441da  =>  addLiquidity(uint256,uint256)
e2981a58  =>  buyBackAndBurn(uint256)
56c3726b  =>  swapTokensForBNB(uint256)
e2545975  =>  swapTokensForDividendToken(uint256,address,address)
682ea2ff  =>  swapAndSendnutgDividends(uint256)
0dcb2e89  =>  updateMinimumTokenBalanceForDividends(uint256)
31e79db0  =>  excludeFromDividends(address)
f22708e3  =>  includeFromDividens(address)
e7841ec0  =>  getLastProcessedIndex()
09bbedde  =>  getNumberOfTokenHolders()
fbcbc0f1  =>  getAccount(address)
5183d6fd  =>  getAccountAtIndex(uint256)
77fdb837  =>  canAutoClaim(uint256)
e30443bc  =>  setBalance(address,uint256)
ffb2c479  =>  process(uint256)
bc4c4b37  =>  processAccount(address,bool)
```


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	NA	NA
Burn	executes transfers of a specified number of tokens to a burn address	NA	NA
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	NA	NA
Lock	stops or locks all function modules of the smart contract	Yes	**! Medium**

Function	Description	Tested	Verdict
Dividend	executes transfers of a specified dividend token to a specified address	Yes	**Passed**
Airdrop	executes transfers of a specified number of tokens to a specified address	NA	NA
Max Transaction	a non-whitelisted wallet can only transfer a specified number of tokens	Yes	**! Low**
Max Wallet	a non-whitelisted wallet can only hold a specified number of tokens	Yes	**Passed**
Cooldown Timer	functionality to limit the number of transactions that a wallet can make within a 24-hour span	NA	NA
Anti Bot	stops some or all bot wallets from interacting with the smart contract	NA	NA
Transfer Ownership	executes transfer of contract ownership to a specified wallet	Yes	**Passed**
Renounce Ownership	executes transfer of contract ownership to a dead address	Yes	**Passed**

Best Practices ✅

- Owner cannot lock or burn the user assets.
- Owner cannot mint tokens after initial contract creation/deployment.
- The smart contract utilizes "SafeMath" function to avoid common smart contract vulnerabilities.

```
string private _name = "NutGain";
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");
```


Note ⚠️

- Active smart contract owner: 0xd510a6fd23f1d909c5cb9e20481abb344b8209d7
- ***Be aware that active smart contract owner privileges constitute an elevated impact to smart contract's safety and security. Important owner only privileges are listed below:***

```
feeStore public buyFee;
feeStore public sellFee;
feeStore public transferFee;
uint256 public maxBuyTranscationAmount;
uint256 public maxSellTransactionAmount;
uint256 public swapTokensAtAmount;
uint256 public maxWalletToken;
```

```
uint256 public sellFeeIncreaseFactor = 150;
uint256 public gasForProcessing = 300000;
address public presaleAddress;
address public lpReceiver;
address public developmentWallet;
```

- Smart contract owner can ***lock*** the smart contract function modules.
- Smart contract owner can ***change the buy and sell fees***. This function module can be used to impose extraordinary transaction fees. No threshold set.
- Smart contract owner can ***change max buy and sell transaction limit %***. The smart contract owner can change the value to "zero". No threshold set.
- Smart contract owner can ***max wallet %***. The smart contract owner can change the value to "zero". No threshold set.

```
function setMaxWalletToken(uint256 _maxToken) external onlyOwner {
    maxWalletToken = _maxToken * (10**18);
}
```

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

{

"resource": " /NutGain.sol",

"owner": "_generated_diagnostic_collection_name_#0",

"severity": 8, (! Low Severity)

" Expected pragma, import directive or contract/interface/library definition",

"source": "solc",

}

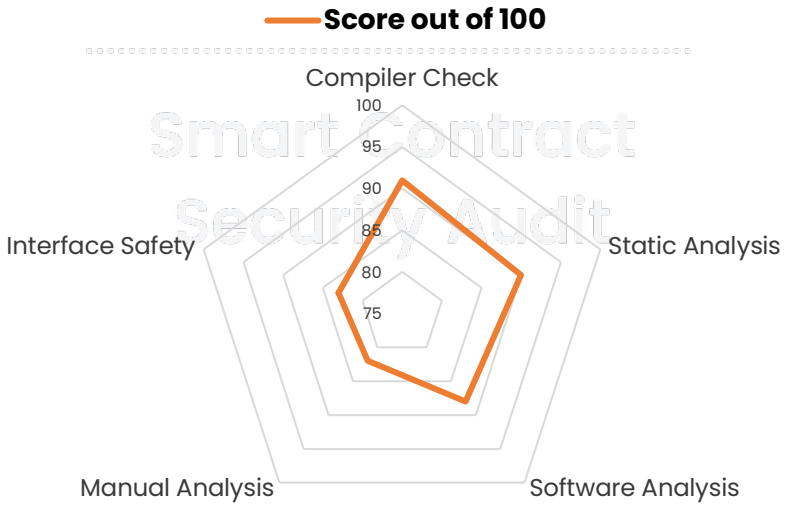
Smart Contract – SWC Attacks

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**

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	**Passed**
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)	**! Low**
SWC-136	Unencrypted Private Data On-Chain	**Passed**

Smart Contract - Risk Status & Radar Chart

Risk Severity	Status
! Critical	None critical severity issues identified
! High	None high severity issues identified
! Medium	1 medium severity issues identified
! Low	3 low severity issues identified
Verified	54 functions and instances verified and checked



Auditor's Verdict

InterFi 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.

- NutGain's smart contract source code has **MEDIUM RISK SEVERITY**
- NutGain's smart contract has an **ACTIVE OWNERSHIP**

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., InterFi.
- 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

InterFi 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 InterFi's prior written consent.**

InterFi 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, InterFi 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.

About InterFi Network

InterFi Network provides intelligent blockchain solutions. InterFi is developing an ecosystem that is seamless and responsive. Some of our services: Blockchain Security, Token Launchpad, NFT Marketplace, etc. **InterFi'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.**

InterFi 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. **InterFi provides manual, static, and automatic smart contract analysis, to ensure that project is checked against known attacks and potential vulnerabilities.**

To learn more, visit **https://interfi.network**

To view our audit portfolio, visit **https://github.com/interfinetwork**

To book an audit, message **https://t.me/interfiaudits**

@INTERFINETWORK

RELENTLESSLY SECURING THE PUBLIC BLOCKCHAIN | MADE IN CANADA