InterFi NETWORK

SMART CONTRACT AUDIT

interfinetwork

hello@interfi.network

https://interfi.network

PREPARED FOR

NOVOOS CONTRACTS

INTRODUCTION

Auditing Firm	InterFi Network
Client Firm	Novoos Contracts
Methodology	Automated Analysis, Manual Code Review
Language	Solidity
Novoos Jackpot Contract	0x79b54054Ccb529A91DE74DD0521564335280EBb4
Wrapped Novoos Contract	0x64D9681f99faC3EC59Ca503B899724bF251C1375
Novoos Rewards Contract	0x5bC484Bedff95Ae43CFf47365485875604730Cc1
Novoos Capital Contract	0xa5601d0e5712Fb990D9dA2c12a13CD7136d6e13B
Blockchain	Binance Smart Chain
Centralization	Active ownership
Commit	5a551f187455fb8a430bad3aae84ab9884e22322
Website	https://novoos.net/ | https://novoos.net/en/novoos-official-links/
Telegram	https://t.me/novoosecosystem/
Twitter	https://twitter.com/Novotoken/
Discord	https://discord.gg/DMQTP4fjEe/
Report Date	March 15, 2023

ℹ️ Verify the authenticity of this report on our website: https://www.github.com/interfinetwork

EXECUTIVE SUMMARY

InterFi has performed the automated and manual analysis of solidity codes. Solidity codes were reviewed for common contract vulnerabilities and centralized exploits. Here's a quick audit summary:

Status	Critical 🔴	Major 🟠	Medium 🟡	Minor 🟢	Unknown 🟤
Open	0	0	0	2	1
Acknowledged	0	0	1	1	0
Resolved	1	1	1	2	1

ℹ️ Please note that smart contracts deployed on blockchains aren't resistant to exploits, vulnerabilities and/or hacks. Blockchain and cryptography assets utilize new and emerging technologies. These technologies present a high level of ongoing risks. For a detailed understanding of risk severity, source code vulnerability, and audit limitations, kindly review the audit report thoroughly.

ℹ️ Please note that centralization privileges regardless of their inherited risk status - constitute an elevated impact on smart contract safety and security.

TABLE OF CONTENTS

TABLE OF CONTENTS 4

SCOPE OF WORK 5

AUDIT METHODOLOGY 7

RISK CATEGORIES 9

CENTRALIZED PRIVILEGES 10

AUTOMATED ANALYSIS 11

INHERITANCE GRAPH 19

MANUAL REVIEW 20

DISCLAIMERS 33

ABOUT INTERFI NETWORK 36

SCOPE OF WORK

InterFi was consulted by Novoos Contracts to conduct the smart contract audit of their solidity source codes. The audit scope of work is strictly limited to mentioned solidity file(s) only:

- NovoosJackpot.sol
- WrappedNovoos.sol
- NovoosRewards.sol
- NovoosCapital.sol

ℹ️ If source codes are not deployed on the main net, they can be modified or altered before main-net deployment. Verify the contract's deployment status below:

Public Contract Link	
https://bscscan.com/address/0x79b54054ccb529a91de74dd0521564335280ebb4#code	
Contract Name	NovoosJackpot
Compiler Version	0.8.17
License	MIT

Public Contract Link	
https://bscscan.com/address/0x64d9681f99fac3ec59ca503b899724bf251c1375#code	
Contract Name	WrappedNovoos
Compiler Version	0.8.4
License	MIT

Public Contract Link	
https://bscscan.com/address/0x5bc484bedff95ae43cff47365485875604730cc1#code	
Contract Name	NovoosRewards
Compiler Version	0.8.17
License	MIT

Public Contract Link	
https://bscscan.com/address/0xa5601d0e5712Fb990D9dA2c12a13CD7136d6e13B#code	
Contract Name	NovoosCapital
Compiler Version	0.8.4
License	MIT

AUDIT METHODOLOGY

Smart contract audits are conducted using a set of standards and procedures. Mutual collaboration is essential to performing an effective smart contract audit. Here's a brief overview of InterFi's auditing process and methodology:

CONNECT

- The onboarding team gathers source codes, and specifications to make sure we understand the size, and scope of the smart contract audit.

AUDIT

- Automated analysis is performed to identify common contract vulnerabilities. We may use the following third-party frameworks and dependencies to perform the automated analysis:
  - Remix IDE Developer Tool
  - Open Zeppelin Code Analyzer
  - SWC Vulnerabilities Registry
  - DEX Dependencies, e.g., Pancakeswap, Uniswap
- Simulations are performed to identify centralized exploits causing contract and/or trade locks.
- A manual line-by-line analysis is performed to identify contract issues and centralized privileges. We may inspect below mentioned common contract vulnerabilities, and centralized exploits:

Centralized Exploits	○ Token Supply Manipulation ○ Access Control and Authorization ○ Assets Manipulation ○ Ownership Control ○ Liquidity Access ○ Stop and Pause Trading ○ Ownable Library Verification

Common Contract Vulnerabilities	○ Integer Overflow ○ Lack of Arbitrary limits ○ Incorrect Inheritance Order ○ Typographical Errors ○ Requirement Violation ○ Gas Optimization ○ Coding Style Violations ○ Re-entrancy ○ Third-Party Dependencies ○ Potential Sandwich Attacks ○ Irrelevant Codes ○ Divide before multiply ○ Conformance to Solidity Naming Guides ○ Compiler Specific Warnings ○ Language Specific Warnings

REPORT

- The auditing team provides a preliminary report specifying all the checks which have been performed and the findings thereof.
- The client's development team reviews the report and makes amendments to solidity codes.
- The auditing team provides the final comprehensive report with open and unresolved issues.

PUBLISH

- The client may use the audit report internally or disclose it publicly.

ℹ️ It is important to note that there is no pass or fail in the audit, it is recommended to view the audit as an unbiased assessment of the safety of solidity codes.

RISK CATEGORIES

Smart contracts are generally designed to hold, approve, and transfer tokens. This makes them very tempting attack targets. A successful external attack may allow the external attacker to directly exploit. A successful centralization-related exploit may allow the privileged role to directly exploit. All risks which are identified in the audit report are categorized here for the reader to review:

Risk Type	Definition
Critical 🔴	These risks could be exploited easily and can lead to asset loss, data loss, asset, or data manipulation. They should be fixed right away.
Major 🟠	These risks are hard to exploit but very important to fix, they carry an elevated risk of smart contract manipulation, which can lead to high-risk severity.
Medium 🟡	These risks 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-risk re-entrancy-related vulnerabilities should be fixed to deter exploits.
Minor 🟢	These risks do not pose a considerable risk to the contract or those who interact with it. They are code-style violations and deviations from standard practices. They should be highlighted and fixed nonetheless.
Unknown 🟤	These risks pose uncertain severity to the contract or those who interact with it. They should be fixed immediately to mitigate the risk uncertainty.

All statuses which are identified in the audit report are categorized here for the reader to review:

Status Type	Definition
Open	Risks are open.
Acknowledged	Risks are acknowledged, but not fixed.
Resolved	Risks are acknowledged and fixed.

CENTRALIZED PRIVILEGES

Centralization risk is the most common cause of cryptography asset loss. When a smart contract has a privileged role, the risk related to centralization is elevated.

There are some well-intended reasons have privileged roles, such as:

- Privileged roles can be granted the power to `pause()` the contract in case of an external attack.
- Privileged roles can use functions like, `include()`, and `exclude()` to add or remove wallets from fees, swap checks, and transaction limits. This is useful to run a presale and to list on an exchange.

Authorizing privileged roles to externally-owned-account (EOA) is dangerous. Lately, centralization-related losses are increasing in frequency and magnitude.

- The client can lower centralization-related risks by implementing below mentioned practices:
- Privileged role's private key must be carefully secured to avoid any potential hack.
- Privileged role should be shared by multi-signature (multi-sig) wallets.
- Authorized privilege can be locked in a contract, user voting, or community DAO can be introduced to unlock the privilege.
- Renouncing the contract ownership, and privileged roles.
- Remove functions with elevated centralization risk.

ℹ️ Understand the project's initial asset distribution. Assets in the liquidity pair should be locked. Assets outside the liquidity pair should be locked with a release schedule.

AUTOMATED ANALYSIS

Symbol	Definition
🛑	Function modifies state
💸	Function is payable
🔒	Function is internal
🔐	Function is private
!	Function is important

Novoos Jackpot

AutomationBase	Implementation			
└	preventExecution	Internal 🔒			
AutomationCompatible	Implementation	AutomationBase, AutomationCompatibleInterface		
AutomationCompatibleInterface	Interface			
└	checkUpkeep	External !	🛑	NO!
└	performUpkeep	External !	🛑	NO!
Initializable	Implementation			
└	_disableInitializers	Internal 🔒	🛑		
└	_getInitializedVersion	Internal 🔒			
└	_isInitializing	Internal 🔒			
AddressUpgradeable	Library			
└	isContract	Internal 🔒			

| └ | sendValue | Internal 🔒 | 🛑 | | |

| └ | functionCall | Internal 🔒 | 🛑 | | |

| └ | functionCall | Internal 🔒 | 🛑 | | |

| └ | functionCallWithValue | Internal 🔒 | 🛑 | | |

| └ | functionCallWithValue | Internal 🔒 | 🛑 | | |

| └ | functionStaticCall | Internal 🔒 | | | |

| └ | functionStaticCall | Internal 🔒 | | | |

| └ | verifyCallResultFromTarget | Internal 🔒 | | | |

| └ | verifyCallResult | Internal 🔒 | | | |

| └ | _revert | Private 🔐 | | | |

||||||

| **IUniswapV2Router02** | Interface | | | |

| └ | swapExactETHForTokensSupportingFeeOnTransferTokens | External ❗ | 💵 | NO❗ | |

| └ | swapExactTokensForETHSupportingFeeOnTransferTokens | External ❗ | 🛑 | NO❗ | |

| └ | swapExactTokensForTokensSupportingFeeOnTransferTokens | External ❗ | 🛑 | NO❗ | |

| └ | swapExactTokensForETH | External ❗ | 🛑 | NO❗ | |

| └ | getAmountsOut | 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❗ |

||||||

| **NovoosJackpot** | Implementation | Initializable, KeeperCompatibleInterface |||

| └ | <Fallback> | External ❗ | 💵 |NO❗ |

| └ | <Receive Ether> | External ❗ | 💵 |NO❗ |

| └ | initialize | Public ❗ | 🛑 | initializer |

| └ | userBuy | External ❗ | 🛑 | onlyToken |

| └ | deposit | External ❗ | 💵 | onlyToken |

| └ | maxAccumulated | Internal 🔒 | 🛑 | |

| └ | buyMainToken | Private 🔐 | 🛑 | |

| └ | sendOneHourReward | Public ❗ | 🛑 | owners |

| └ | sendBigBuyReward | Public ❗ | 🛑 | owners |

| └ | checkUpkeep | External ❗ | |NO❗ |

| └ | performUpkeep | External ❗ | 🛑 |NO❗ |

| └ | setBigBuyTime | Public ❗ | 🛑 | limited |

| └ | setJackpotLimit | Public ❗ | 🛑 | limited |

| └ | setRouterAndPairAddress | Public ❗ | 🛑 | limited |

| └ | setTokenAddress | Public ❗ | 🛑 | limited |

| └ | setBigBuyPercentage | Public ❗ | 🛑 | limited |

| └ | setBigBuyMarketingPercentage | Public ❗ | 🛑 | limited |

| └ | setBigBuyInsurancePercentage | Public ❗ | 🛑 | limited |

| └ | setOneHourPercentage | Public ❗ | 🛑 | limited |

| └ | setOneHourMarketingPercentage | Public ❗ | 🛑 | limited |

| └ | setOneHourInsurancePercentage | Public ❗ | 🛑 | limited |

| └ | setBuyBackPercentage | Public ❗ | 🛑 | limited |

| └ | setMarketingHitPercentage | Public ❗ | 🛑 | limited |

| └ | setDevelopmentHitPercentage | Public ❗ | 🛑 | limited |

| └ | setMarketingAddress | Public ❗ | 🛑 | limited |

| └ | setInsuranceAndRewardAddress | Public ❗ | 🛑 | limited |

| └ | setDevelopmentAddress | Public ❗ | 🛑 | limited |

| └ | setPairAddress | Public ❗ | 🛑 | limited |

| └ | balance | Public ❗ | | NO❗ |

| └ | transferOwnership | External ❗ | 🛑 | limited |

| └ | setVrfOwnership | External ❗ | 🛑 | limited |

Wrapped Novoos

| **IERC20** | Interface | |||

| └ | transfer | External ❗ | 🛑 | NO❗ |

| └ | approve | External ❗ | 🛑 | NO❗ |

| └ | transferFrom | External ❗ | 🛑 | NO❗ |

| └ | balanceOf | External ❗ | | NO❗ |

||||||

| **WNOVO** | Implementation | Initializable |||

| └ | initialize | Public ❗ | 🛑 | initializer |

| └ | deposit | Public ❗ | 🛑 | nonReentrant |

| └ | withdraw | Public ❗ | 🛑 | nonReentrant |

| └ | totalSupply | Public ❗ | | NO❗ |

| └ | approve | Public ❗ | 🛑 | NO❗ |

| └ | transfer | Public ❗ | 🛑 | NO❗ |

| └ | transferFrom | Public ❗ | 🛑 | NO❗ |

| └ | _nonReentrantBefore | Private 🔐 | 🛑 | |

| └ | _nonReentrantAfter | Private 🔐 | 🛑 | |

||||||

| **Initializable** | Implementation | |||

| └ | _disableInitializers | Internal 🔒 | 🛑 | |

| └ | _setInitializedVersion | Private 🔐 | 🛑 | |

|||||

| **AddressUpgradeable** | Library | |||

| └ | isContract | Internal 🔒 | | |

| └ | sendValue | Internal 🔒 | 🛑 | |

| └ | functionCall | Internal 🔒 | 🛑 | |

| └ | functionCall | Internal 🔒 | 🛑 | |

| └ | functionCallWithValue | Internal 🔒 | 🛑 | |

| └ | functionCallWithValue | Internal 🔒 | 🛑 | |

| └ | functionStaticCall | Internal 🔒 | | |

| └ | functionStaticCall | Internal 🔒 | | |

| └ | verifyCallResult | Internal 🔒 | | |

Rewards

| **SafeMath** | Library | |||

| └ | add | Internal 🔒 | | |

| └ | sub | Internal 🔒 | | |

| └ | sub | Internal 🔒 | | |

| └ | mul | Internal 🔒 | | |

| └ | div | Internal 🔒 | | |

| └ | div | Internal 🔒 | | |

|||||

| **SafeMathInt** | Library | |||

| └ | mul | Internal 🔒 | | |

| └ | div | Internal 🔒 | | |

| └ | sub | Internal 🔒 | | |

| └ | add | Internal 🔒 | | |

| └ | abs | Internal 🔒 | | |

|||||

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!
IDEXRouter	Interface			
└	factory	External !		NO!
└	WETH	External !		NO!
└	addLiquidity	External !	🛑	NO!
└	addLiquidityETH	External !	💵	NO!
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External !	🛑	NO!
└	swapExactETHForTokensSupportingFeeOnTransferTokens	External !	💵	NO!
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External !	🛑	NO!
└	getAmountsOut	External !		NO!
IDividendDistributor	Interface			
└	setDistributionCriteria	External !	🛑	NO!
└	setShare	External !	🛑	NO!
└	deposit	External !	💵	NO!
└	process	External !	🛑	NO!

NovoosRewards	Implementation	IDividendDistributor, Initializable		
└	<Fallback>	External ❗	💵	NO❗
└	<Receive Ether>	External ❗	💵	NO❗
└	initialize	Public ❗	🛑	initializer
└	setDistributionCriteria	External ❗	🛑	onlyOwner
└	setShare	External ❗	🛑	onlyToken
└	deposit	External ❗	💵	onlyToken
└	process	External ❗	🛑	onlyToken
└	shouldDistribute	Internal 🔒		
└	distributeDividend	Internal 🔒	🛑	
└	claimDividend	External ❗	🛑	NO❗
└	getUnpaidEarnings	Public ❗		NO❗
└	getCumulativeDividends	Internal 🔒		
└	addShareholder	Internal 🔒	🛑	
└	removeShareholder	Internal 🔒	🛑	
└	setBankAddress	External ❗	🛑	onlyOwner
└	transferOwnership	External ❗	🛑	onlyOwner
Initializable	Implementation			
└	_disableInitializers	Internal 🔒	🛑	
└	_setInitializedVersion	Private 🔐	🛑	
AddressUpgradeable	Library			
└	isContract	Internal 🔒		
└	sendValue	Internal 🔒	🛑	
└	functionCall	Internal 🔒	🛑	
└	functionCall	Internal 🔒	🛑	
└	functionCallWithValue	Internal 🔒	🛑	

| └ | functionCallWithValue | Internal 🔒 | 🛑 | | |

| └ | functionStaticCall | Internal 🔒 | | | |

| └ | functionStaticCall | Internal 🔒 | | | |

| └ | verifyCallResult | Internal 🔒 | | | |

Novoos Capital

| **IERC20** | Interface | | | |

| └ | transfer | External ❗ | 🛑 | NO❗ | |

| └ | balanceOf | External ❗ | | NO❗ | |

| | | | | | |

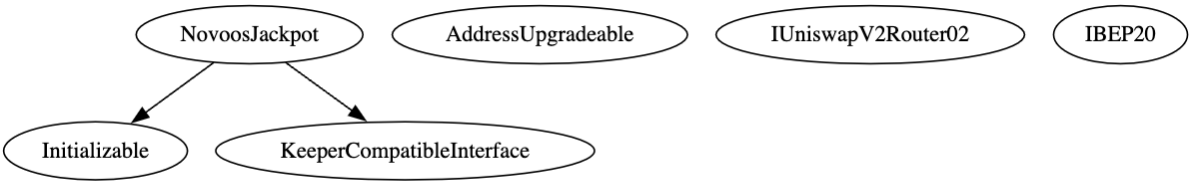
| **NovoosCapital** | Implementation | | | |

| └ | <Constructor> | Public ❗ | 🛑 | NO❗ | |

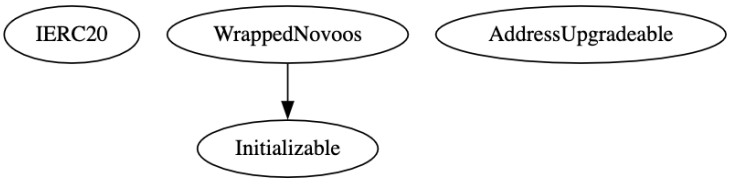
| └ | transfer | Public ❗ | 🛑 | NO❗ | |

INHERITANCE GRAPH

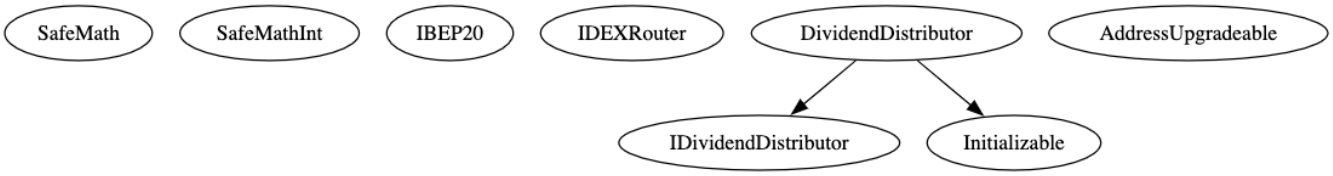
Novoos Jackpot



Wrapped Novoos



Rewards



Novoos Capital

MANUAL REVIEW

Identifier	Definition	Severity
CEN-01	Noteworthy centralized privileges of Novoos Contracts	Major 🟠

NovoosJackpot

`sendOneHourReward()`
`sendBigBuyReward()`
`setBigBuyTime()`
`setJackpotLimit()`
`setRouterAndPairAddress()`
`setTokenAddress()`
`setBigBuyPercentage()`
`setBigBuyMarketingPercentage()`
`setBigBuyInsurancePercentage()`
`setOneHourPercentage()`
`setOneHourMarketingPercentage()`
`setOneHourInsurancePercentage()`
`setBuyBackPercentage()`
`setMarketingHitPercentage()`
`setDevelopmentHitPercentage()`
`setMarketingAddress()`
`setInsuranceAndRewardAddress()`
`setDevelopmentAddress()`
`setPairAddress()`
`transferOwnership()`
`setVrfOwnership()`

WrappedNovoos

`Initialize()`

Rewards

`setDistributionCriteria()`
`setBankAddress()`

RECOMMENDATION

Deployer, contract owner, and privileged roles' private keys should be secured carefully. Please refer to PAGE-09 CENTRALIZED PRIVILEGES for a detailed understanding.

RESOLUTION

Centralization is needed and is publicly available to review. Here are statements provided by Novoos:

NovoosJackpot

When we are changing the jackpot timer, the reward percentage pay out etc. for marketing campaigns long term and it should only be done by the owner as it is.

Novoos jackpot, as an example for the jackpot limit and these privileges belong to the owner and one identity only: `sendOneHourReward()` `sendBigBuyReward()` `setBigBuyTime()` `setJackpotLimit()` `setRouterAndPairAddress()` `setTokenAddress()` `setBigBuyPercentage()` `setOneHourPercentage()` `setBuyBackPercentage()` `setBankAddress()` `setPairAddress()`

Campaigns will be run for the jackpots, for example the payouts, percentages can be changed for this where we can add more to the jackpot and market it as a contest periodically.

Plus, the jackpot and volume could be very high at times and we can increase the pot of the buyback of the $NOVO token threshold, or campaigns for higher percentage payouts to the hourly and 24 our jackpot etc.

The addresses have to be flexible; you can imagine our security setup with the cold wallets and other safety measures are top of the range, however, we should be able to change it, just in case the address is, in the circumstances taken over as an example which the chance of is lower than 99% against everyone else due to our setup which I shared the video of with you (Shamir Backups, hidden cold wallets etc.)

WrappedNovoos

`Initialize()` Initialize is the same as a constructor function for our upgradeable contract, it will work in the same way.

Rewards

`setDistributionCriteria()` This handles how many rewards should be distributed, minimum rewards distribution or minimum period time, again the distribution will be dynamic and with marketing campaigns ran where it will be higher at times to incentivize investors to buy more and hold etc.

`setBankAddress()` As stated above, the addresses have to be changeable, you can imagine our security setup with the cold wallets and other safety measures are top of the range, however, we should be able to change it, just in case the address is, in the circumstances taken over as an example which the chance of is lower than 99% against everyone else due to our setup which I shared the video of with you (Shamir Backups, hidden cold wallets etc.)

Identifier	Definition	Severity
CEN-06	Privileged role modifying pair and router	Minor

Privileged role can call `setRouterAndPairAddress()` and `setPairAddress()`

```
function setRouterAndPairAddress(
    address uniswapV2Router,
    address uniswapV2Pair
) public limited {
    _uniswapV2Router = uniswapV2Router;
    _uniswapV2Pair = uniswapV2Pair;
}

function setPairAddress(address _pairAddress) public limited {
    _uniswapV2Pair = _pairAddress;
}
```


RECOMMENDATION

The current trading pair, e.g., Pancakeswap pair should not be removed from automated market makers.

Identifier	Definition	Severity
CEN-09	Use of proxy and upgradeable contracts	Critical 🔴

Privileged role can initiate contract implementation. Contract upgradeability allows privileged roles to change current contract implementation.

RECOMMENDATION

Test and validate current contract thoroughly before deployment. Future contract upgradeability negatively elevates centralization risk.

RESOLUTION

According to Novoos, it has implemented upgradeable contracts for continuous seamless contract enhancements without any relaunches, and migrations.

Identifier	Definition	Severity
LOG-01	Lack of appropriate arbitrary boundaries	Medium 🟡

Below mentioned functions are set without any arbitrary boundaries.

```
setBigBuyTime()
setJackpotLimit()
setBigBuyPercentage()
setBigBuyMarketingPercentage()
setBigBuyInsurancePercentage()
setOneHourPercentage()
setOneHourMarketingPercentage()
setOneHourInsurancePercentage()
setBuyBackPercentage()
setMarketingHitPercentage()
setDevelopmentHitPercentage()
```


RECOMMENDATION

These functions should be provided appropriate upper and lower boundaries.

ACKNOWLEDGEMENT

Novoos team has acknowledged to this finding, and agreed to keep it as-is.

Identifier	Definition	Severity
LOG-03	Re-entrancy	Medium 🟡

In Wrapped Novoos contract, below mentioned functions are used without re-entrancy guard:

```
deposit()
withdraw()
```


RECOMMENDATION

Re-entrancy guard is used to prevent re-entrant calls. Learn more: https://consensys.github.io/smart-contract-best-practices/attacks/reentrancy/

RESOLUTION

Novoos team has added `nonReentrant` modifier to `deposit()` and `withdraw()`.

Identifier	Definition	Severity
COD-01	Authorization through `tx.origin`	Unknown

Using `tx.origin` for authorization could make the contract vulnerable as it refers to the original external account that started the transaction.

```
function preventExecution() internal view {
  if (tx.origin != address(0)) {
    revert OnlySimulatedBackend();
  }
}
```


RECOMMENDATION

Avoid authorizations via global variables wherever necessary.

Identifier	Definition	Severity
COD-02	Timestamp dependence via `block.timestamp` in Novoos Jackpot	Minor 🟢

Be aware that the timestamp of the block can be manipulated by a miner. When the contract uses the timestamp to seed a random number, the miner can actually post a timestamp within 15 seconds of the block being validated, effectively allowing the miner to precompute an option more favorable to their chances.

RECOMMENDATION

To maintain block integrity, follow 15 seconds rule, and scale time dependent events accordingly.

Identifier	Definition	Severity
COD-05	Missing zero address validation	Minor

Below mentioned functions are missing zero address input validation:

`setBankAddress()`
`setRouterAndPairAddress()`
`setTokenAddress()`
`setMarketingAddress()`
`setInsuranceAndRewardAddress()`
`setDevelopmentAddress()`
`setPairAddress()`
`transferOwnership()`
`setVrfOwnership()`

RECOMMENDATION

Validate if the modified address is `dead(0)` or not.

ACKNOWLEDGEMENT

Novoos team has acknowledged to this finding, and agreed to keep it as-is.

Identifier	Definition	Severity
COD-06	Unknown externally owned account and address	Minor 🟢

An externally owned account (EOA) has no code, and one can send messages from an externally owned account by creating and signing a transaction.

```
bankAddress=address(0xf2e14c21ed6B505b4E51Bb494481378eE5C9Ca85);
tokenAddress = address(0xD655981fB2F15498e3279EC34CAe2baC6c6744CD);
marketingAddress = payable(0xeEE5Eb161aBEa5afe61c6DE4b8231Db29acc3a7f);
insuranceAndRewardAddress = payable(
    0x29B4920AA795899aeFB0514BC273bb040F4AcDD4
developmentAddress = payable(
    0x4c3935Cf504FB36e65015B7Af5bE0963C8Fb0339
```


RECOMMENDATION

Private keys of externally owned accounts and ownership of externally owned addresses must be secured carefully.

PARTIAL RESOLUTION

According to Novoos team, EOAs are backed up and secured adequately.

Identifier	Definition	Severity
COD-10	Third Party Dependencies	Unknown

Smart contract is interacting with third party protocols e.g., Market Makers, Novoos token contract, Open Zeppelin tools. The scope of the audit treats third party entities as black boxes and assumes their functional correctness. However, in the real world, third parties can be compromised, and exploited. Moreover, upgrades in third parties can create severe impacts, e.g., increased transactional fees, deprecation of previous routers, etc.

RECOMMENDATION

Inspect third party dependencies regularly, and mitigate severe impacts whenever necessary.

RESOLUTION

Novoos team will inspect third party dependencies periodically.

Identifier	Definition	Severity
COM-01	Floating compiler status	Minor

Compilers should be fixed to the version that you're indenting to deploy your contracts with.

Rewards

```
pragma solidity >=0.8.4;
```

Novoos Capital:

```
pragma solidity >=0.8.2;
```

Novoos Jackpot:

```
pragma solidity >=0.8.2;
```

Wrapped Novoos:

```
pragma solidity >=0.8.4;
```


RECOMMENDATION

Pragma should be fixed to the version that you're indenting to deploy your contracts with.

RESOLUTION

According to Novoos team, contract will be deployed with appropriate stable compiler versions at the time of deployment.

DISCLAIMERS

InterFi Network provides the easy-to-understand audit of solidity source codes (commonly known as smart contracts).

The smart contract for this particular audit was analyzed for common contract vulnerabilities, and centralization exploits. This audit report makes no statements or warranties on the security of the code. This audit report does not provide any warranty or guarantee regarding the absolute bug-free nature of the smart contract analyzed, nor do they provide any indication of the client's business, business model or legal compliance. This audit report does not extend to the compiler layer, any other areas beyond the programming language, or other programming aspects that could present security risks. Cryptographic tokens are emergent technologies, they carry high levels of technical risks and uncertainty. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. This audit report could include false positives, false negatives, and other unpredictable results.

CONFIDENTIALITY

This report is subject to the terms and conditions (including without limitations, description of services, confidentiality, disclaimer and limitation of liability) outlined in the scope of the audit provided to the client. This report should not be transmitted, disclosed, referred to, or relied upon by any individual for any purpose without InterFi Network's prior written consent.

NO FINANCIAL ADVICE

This audit report does not indicate the endorsement of any particular project or team, nor guarantees its security. No third party should rely on the reports in any way, including to make any decisions to buy or sell a product, service or any other asset. The information provided in 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. This audit report should not be used in any way

to make decisions around investment or involvement. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort.

FOR AVOIDANCE OF DOUBT, SERVICES, INCLUDING ANY ASSOCIATED AUDIT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

TECHNICAL DISCLAIMER

ALL SERVICES, AUDIT REPORTS, SMART CONTRACT AUDITS, OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, INTERFI NETWORK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO SERVICES, AUDIT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, INTERFI NETWORK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM THE COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

WITHOUT LIMITING THE FOREGOING, INTERFI NETWORK MAKES NO WARRANTY OF ANY KIND THAT ALL SERVICES, AUDIT REPORTS, SMART CONTRACT AUDITS, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET THE CLIENT'S OR ANY OTHER INDIVIDUAL'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE.

TIMELINESS OF CONTENT

The content contained in this audit report is subject to change without any prior notice. InterFi Network does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following the publication.

LINKS TO OTHER WEBSITES

This audit report provides, through hypertext or other computer links, access to websites and social accounts operated by individuals other than InterFi Network. Such hyperlinks are provided for your reference and convenience only and are the exclusive responsibility of such websites' and social accounts' owners. You agree that InterFi Network is not responsible for the content or operation of such websites and social accounts and that InterFi Network shall have no liability to you or any other person or entity for the use of third-party websites and social accounts. You are solely responsible for determining the extent to which you may use any content at any other websites and social accounts to which you link from the report.

ABOUT INTERFI NETWORK

InterFi Network provides intelligent blockchain solutions. We provide solidity development, testing, and auditing services. We have developed 150+ solidity codes, audited 1000+ smart contracts, and analyzed 500,000+ code lines. We have worked on major public blockchains e.g., Ethereum, Binance, Cronos, Doge, Polygon, Avalanche, Metis, Fantom, Bitcoin Cash, Velas, Oasis, etc.

InterFi Network is built by engineers, developers, UI experts, and blockchain enthusiasts. Our team currently consists of 4 core members, and 6+ casual contributors.

Website: https://interfi.network

Email: hello@interfi.network

GitHub: https://github.com/interfinetwork

Telegram (Engineering): https://t.me/interfiaudits

Telegram (Onboarding): https://t.me/interfisupport

interfinetwork

hello@interfi.network

https://interfi.network

SMART CONTRACT AUDITS | SOLIDITY DEVELOPMENT AND TESTING

RELENTLESSLY SECURING PUBLIC AND PRIVATE BLOCKCHAINS