InterFi NETWORK

SMART CONTRACT AUDIT

interfinetwork

hello@interfi.network

https://interfi.network

PREPARED FOR

BNBABYLON

INTRODUCTION

Auditing Firm	InterFi Network
Client Firm	BNBabylon
Methodology	Automated Analysis, Manual Code Review
Language	Solidity
Contract	0xbA1c2E3DbF96544A62Fe8A29Edf1baBe38C1c318
Blockchain	Binance Smart Chain
Centralization	Active ownership
Commit	4b99e780fd58facd20761b660cde55b23c6e7a69
Website	https://bnbabylon.com/
Report Date	August 16, 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	3	4	0
Acknowledged	0	0	0	1	1
Resolved	1	1	0	0	0

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

SCOPE OF WORK

InterFi was consulted by BNBabylon 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:

- BNBabylon.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/0xbA1c2E3DbF96544A62Fe8A29Edf1baBe38C1c318#code	
Contract Name	BNBabylon
Compiler Version	0.5.16
License	GPL-3.0-or-later

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

SafeMath	Library			
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	sub	Internal 🔒		
└	add	Internal 🔒		

Percent	Library			
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	sub	Internal 🔒		
└	add	Internal 🔒		

BNBAbylon	Implementation			
└	<Constructor>	Public !	🛑	NO!
└	<Fallback>	External !	💸	NO!
└	_setSponsor	Private 🔐	🛑	
└	_deposit	Private 🔐	🛑	
└	_reinvest	Private 🔐	🛑	

| └ | _refPayout | Private 🔐 | 🛑 | |

| └ | deposit | External ❗ | 💵 |NO❗ |

| └ | withdraw | External ❗ | 🛑 |NO❗ |

| └ | maxPayoutOf | Internal 🔒 | | |

| └ | payoutOf | External ❗ | |NO❗ |

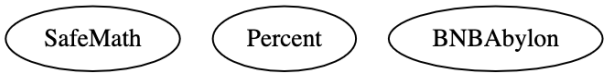
| └ | dailyBonus | Internal 🔒 | | |

| └ | getHoldBonus | Public ❗ | |NO❗ |

| └ | holdBonus | Internal 🔒 | | |

| └ | referralsCount | External ❗ | |NO❗ |

INHERITANCE GRAPH



MANUAL REVIEW

Identifier	Definition	Severity
CEN-01	Lack of centralized privileges and role-based access controls	Medium 🟡
LOG-01	Unrestricted operations	

Smart contract lacks clear ownership mechanism or access control to set administrative values. Access control is a fundamental security measure in smart contracts to prevent unauthorized state changes.

Smart contract sets `owner` and `promote` addresses. These addresses have privileges like collecting fees.

`_setSponsor()`, `_deposit()`, `_reinvest()`, and `_refPayout()` functions are marked as `private` and may potentially be centralized points of control or failure.

Mentioned functions aren't provided any clear access control restrictions:

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

RECOMMENDATION

Add access control restrictions wherever necessary. Ensure transparency with users regarding fees or benefits `owner` and `promote` addresses receive.

Identifier	Definition	
ABY-01	Contract uniqueness	

- Smart contract is written to achieve High Yield Investment Program (HYIP) logic.
- Smart contract implements a multi-level referral system where rewards are provided for multiple levels of referrals. Ensure that smart contract and its functionalities (like referral system) are compliant with the regulations of the jurisdictions it operates in.
- Smart contract enforces a one withdrawal per day policy. Users should be made aware of this.
- Smart contract has an automatic reinvestment feature. Users should be informed of this as it affects their withdrawals.

Identifier	Definition	Severity
LOG-03	Re-entrancy	Critical 🔴

Below mentioned functions are used without re-entrancy guard:

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


RECOMMENDATION

Use Checks Effects Interactions pattern when handing over the flow to an external entity and/or guard functions against re-entrancy attacks. Re-entrancy guard is used to prevent re-entrant calls.

RESOLUTION

BNBabylon team has added `nonReentrant` modifier to deter re-entrant calls to mentioned functions.

Identifier	Definition	Severity
LOG-04	Delegate call to untrusted contract	Medium 🟡

Using `.call()` is potentially dangerous because it allows the callee to execute arbitrary code within the context of calling contract.

```
_deposit()
withdraw()
```


RECOMMENDATION

Always be cautious of the ordering and potential for state changes.

Identifier	Definition	Severity
LOG-02	Potential front-running	Minor

Front-running risk exists because an attacker can observe the pending transactions in the `mempool` and submit a transaction with a higher gas price to have their transaction executed before the original one. Due to the transparent nature of blockchain transactions, malicious actors can observe pending transactions and try to front-run them. For instance, in `deposit`, `withdraw` functions, someone can observe a transaction and try to interact with the contract before that transaction is mined.

RECOMMENDATION

Functions responsible for transfer should be provided reasonable minimum output amounts, instead of zero.

ACKNOWLEDGEMENT

BNBabylon team argued that, in stake contract, owners with appropriate allowance of their own tokens can stake. Front-running these operations should not be a viable outcome.

Identifier	Definition	Severity
COD-06	Unknown externally owned account	Minor 🟢
LOG-04	Logical oversight	

- Hardcoded values are pushed in constructor for `PERCENT_REFERRAL` and `PLANS`.
- Smart contract has hardcoded addresses for `owner` and `promote`.

```
address payable private owner =
    address(0x5c00114237e2FC579B80176dca49dBfd639a2cC8);
address payable private promote =
    address(0x979707294108A6Ec6A5A304E3709ce7A223AB0B9);
```


NOTE

This is not a vulnerability itself per se, it's an oversight that can be reviewed.

COD-06 RECOMMENDATION

Private keys of externally owned accounts must be secured carefully.

Identifier	Definition	Severity
COD-02	Timestamp manipulation via `block.timestamp`	Minor

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

RECOMMENDATION

Ensure that minor deviations in timing aren't critical to the contract's operation.

Identifier	Definition	
COD-04	Potential flash loan scenario	

Smart contracts do not appear to have direct flash loan vulnerabilities. Flash loan attacks typically exploit functions that rely on external data, e.g., oracles or functions that can be gamed within a single transaction.

NOTE

Due to the interconnected nature of DeFi contracts, one contract's vulnerability can be exploited using another contract. Be cautious while interacting with third-party tokens.

Identifier	Definition	
COD-09	Lack of emergency stop mechanism	

Add functions like `pause()` and `emergencyWithdraw()` to halt critical functions or allow users to withdraw their tokens in certain scenarios.

Identifier	Definition	Severity
COD-10	Direct and indirect dependencies	Unknown

Smart contracts are interacting with addresses, contracts, protocols e.g., Staked Token Contracts, Promote and Owner Addresses, External Contracts and Interfaces, Web 3 Applications, Open Zeppelin tools, etc. The scope of the audit treats these entities as black boxes and assumes their functional correctness. However, in the real world, all of them can be compromised, and exploited. Moreover, upgrades in these entities can create severe impacts, e.g., increased transactional fees, deprecation of previous routers pairs, etc. It's crucial to be aware that staked token contract (or any entity with control of it) may have a high degree of power over the assets which are being locked.

RECOMMENDATION

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

ACKNOWLEDGEMENT

BNBabylon team will inspect third party dependencies to minimize downtime from third-party intervention.

Identifier	Definition	Severity
COD-12	Lack of event-driven architecture	Minor

Smart contract uses function calls to update state, which can make it difficult to track and analyze changes to the contract over time.

RECOMMENDATION

Use events to track state changes. Events improve transparency and provide a more granular view of contract activity.

Identifier	Definition	Severity
VOL-02	Use of percentage system	Medium 🟡

It is not recommended to use systems following percentage-based calculations. `Percent` is not a standard solidity library. Due to rounding and precision limits, there could be unintended behaviors.

```
contract BNBAbylon {
    using SafeMath for uint256;
    using Percent for Percent.percent;
```


RECOMMENDATION

`Percent` is not a standard solidity library. Stick to core Solidity data types and mathematics, e.g., use basis points instead of representing percentages as fractions.

Identifier	Definition	Severity
COM-01	Floating compiler status	Major 🟠
COM-02	Outdated compiler version	

Compiler is set to `^0.5.16`.

RECOMMENDATION

Pragma should be fixed to the version that you're indenting to deploy your contracts with. Use compiler version `0.8.12` and above.

RESOLUTION

BNBabylon team has deployed contract with `0.8.12`.

Identifier	Definition	Severity
COM-04	Potential resource exhaustion errors	Minor

Below mentioned loops may throw out of gas errors:

`holdBonusMultiplier`
`PERCENT_REFERRAL`

RECOMMENDATION

Set max bound for loops.

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