InterFi
NETWORK

# SMART CONTRACT AUDIT

interfinetwork

hello@interfi.network

https://interfi.network

PREPARED FOR

## CREATIVE WEALTH

INTERFI SMART CONTRACT AUDIT

# INTRODUCTION

| | |
|---|---|
| Auditing Firm | InterFi Network |
| Client Firm | Creative Wealth |
| Methodology | Automated Analysis, Manual Code Review |
| Language | Solidity |
| | |
| Contract | 0xB1095b653CD865a61EbCa425CA68BA7c3cF676Ff |
| Blockchain | Binance Smart Chain |
| Centralization | Active ownership |
| Commit | 2156d5bdd574df4d7577832b5f075ee5e8b920e9 |
| | |
| Website | https://creativewealth.finance/ |
| Report Date | February 19, 2023 |

ℹ 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 | Major 🟠 | Medium 🟡 | Minor 🟢 | Unknown 🟤 | Informational ⚪ |
|---|---|---|---|---|---|
| Open | 1 | 0 | 2 | 0 | 1 |
| Acknowledged | 0 | 0 | 2 | 1 | 0 |
| Resolved | 0 | 1* | 1 | 0 | 2 |
| | | | | | |
| Noteworthy Privileges | Set Password, Update BUSD Address, Update Min/Max User Investment, Toggle Deposit/Withdraw State, Withdraw BUSD | | | | |

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 Creative Wealth 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:

- trading_contract.sol

## DEPLOYMENT

| | |
|---|---|
| Public Contract Link | |
| https://bscscan.com/address/0xb1095b653cd865a61ebca425ca68ba7c3cf676ff#code | |
| | |
| Contract Name | Trading |
| Compiler Version | 0.8.17 |
| License | MIT |

ℹ 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:

## IMPORTANT SUMMARY

- Trading contract manages user investments and calculates the return on investment.
- Contract allows users to invest a minimum of 50 BUSD and a maximum of 500 BUSD, and the total maximum investment amount may not exceed 1,000,000 BUSD.
- Password is used for access via dapp. Owner can update password.

# 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<br>○ Access Control and Authorization<br>○ Assets Manipulation<br>○ Ownership Control<br>○ Liquidity Access<br>○ Stop and Pause Trading<br>○ Ownable Library Verification |

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

ℹ 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 |
|---|---|
| Major 🟠 | These risks can be exploited easily and can lead to asset loss, data loss, asset, or data manipulation. They should be fixed right away. |
| Medium 🟡 | These risks are very important to fix, they carry an elevated risk of smart contract exploitation, which can lead to major-risk severity. |
| 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. |
| Informational ⚪ | These risks do not pose any risk to the contract or those who interact with it. These are compiler version, optimization and gas-related issues. |

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.

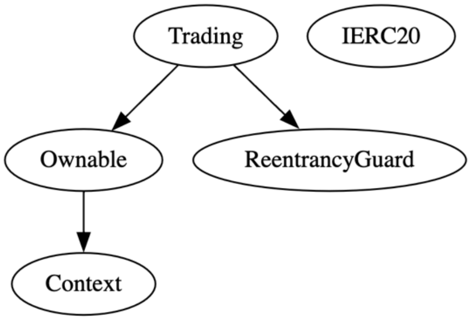
ℹ Understand the project's initial asset distribution. Assets in the liquidity pair should be locked. Assets outside the liquidity pair should be locked with a release schedule.

# AUTOMATED ANALYSIS

| Symbol | Definition |
|---|---|
| 🛑 | Function modifies state |
| 💵 | Function is payable |
| 🔒 | Function is internal |
| 🔐 | Function is private |
| ❗ | Function is important |

```
| **Context** | Implementation |  |||
| └ | _msgSender | Internal 🔒 |   | |
| └ | _msgData | Internal 🔒 |   | |
||||||
| **Ownable** | Implementation | Context |||
| └ | <Constructor> | Public ❗ | 🛑  |NO❗ |
| └ | owner | Public ❗ |   |NO❗ |
| └ | _checkOwner | Internal 🔒 |   | |
| └ | renounceOwnership | Public ❗ | 🛑  | onlyOwner |
| └ | transferOwnership | Public ❗ | 🛑  | onlyOwner |
| └ | _transferOwnership | Internal 🔒 | 🛑  | |
||||||
| **IERC20** | Interface |  |||
| └ | totalSupply | External ❗ |   |NO❗ |
| └ | balanceOf | External ❗ |   |NO❗ |
| └ | transfer | External ❗ | 🛑  |NO❗ |
| └ | allowance | External ❗ |   |NO❗ |
| └ | approve | External ❗ | 🛑  |NO❗ |
| └ | transferFrom | External ❗ | 🛑  |NO❗ |
||||||
| **ReentrancyGuard** | Implementation |  |||
| └ | <Constructor> | Public ❗ | 🛑  |NO❗ |
||||||
| **Trading** | Implementation | Ownable, ReentrancyGuard |||
| └ | <Constructor> | Public ❗ | 🛑  |NO❗ |
| └ | <Receive Ether> | External ❗ |  💵 |NO❗ |
| └ | <Fallback> | External ❗ |  💵 |NO❗ |
| └ | invest | External ❗ | 🛑  | nonReentrant |
| └ | monthlyWithdraw | External ❗ | 🛑  | nonReentrant |
| └ | updateROIWeekly | External ❗ | 🛑  | onlyOwner |
| └ | weeklyWithdraw | External ❗ | 🛑  | nonReentrant |
| └ | withdraw | External ❗ | 🛑  | onlyOwner |
| └ | fund | External ❗ | 🛑  | onlyOwner |
| └ | setPassword | External ❗ | 🛑  | onlyOwner |
| └ | toggleWithdrawOnAndOff | External ❗ | 🛑  | onlyOwner |
| └ | toggleDeposit | External ❗ | 🛑  | onlyOwner |
| └ | updateMaxInvestment | External ❗ | 🛑  | onlyOwner |
| └ | updateMinInvestment | External ❗ | 🛑  | onlyOwner |
| └ | updateMaxUserInvestment | External ❗ | 🛑  | onlyOwner |
| └ | updateBusdAddress | External ❗ | 🛑  | onlyOwner |
```

# INHERITANCE GRAPH



# MANUAL REVIEW

| Identifier | Definition | Severity |
|---|---|---|
| CEN-01 | Centralized privileges of Trading | Major ● |
| CEN-05 | Privileged role halting deposits and withdrawals | |
| CRE-00 | Privileged role withdrawing BUSD to any EOAs | |
| CRE-02 | Privileged role updating password | |

`onlyOwner` centralized privileges are listed below:

```
transferOwnership()
updateROIWeekly()
withdraw()
fund()
setPassword()
toggleWithdrawOnAndOff()
toggleDeposit()
updateMaxInvestment()
updateMinInvestment()
updateMaxUserInvestment()
updateBusdAddress()
```

## RECOMMENDATION

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

| Identifier | Definition | Severity |
|---|---|---|
| LOG-01 | Lack of appropriate arbitrary boundaries | Minor 🟢 |

Below mentioned functions are set without any arbitrary boundaries.

```
updateMaxInvestment()
updateMinInvestment()
updateMaxUserInvestment()
```

### RECOMMENDATION

These functions should be provided appropriate upper and lower boundaries.

### ACKNOWLEDGEMENT

Project team has acknowledged this finding and has kept the code as-is.

| Identifier | Definition | Severity |
|---|---|---|
| LOG-03 | Re-entrancy | Medium ● |

Since mentioned functions perform state change, it is recommended to add re-entrancy guard.

```
monthlyWithdraw()
invest()
weeklyWithdraw()
```

Moreover `monthlyWithdraw()` should verify mentioned instances:

- `if` statement that checks whether the remaining investment amount after withdrawal will be less than the minimum investment amount can be split this into two separate conditions: one to check if the remaining amount will be zero, and another to check if it will be less than the minimum investment amount.
- Verify if the `monthlyWithdraw()` was successfully called or not.

### RECOMMENDATION

Guard aforementioned functions against re-entrancy attacks. Re-entrancy attack happens when an attacker repeatedly calls a function within a contract before the previous invocation has been completed, in order to gain control of the flow of execution and potentially manipulate contract.

### PARTIAL RESOLUTION*

Aforementioned functions are protected against re-entrancy attack with `nonReentrant` modifier.

| Identifier | Definition | Severity |
|---|---|---|
| CRE-01 | Password use to access via dapp | Minor 🟢 |

Privileged role can change password to access via dapp.

```
function updatePassword(string memory _password) external onlyOwner {
    password = keccak256(abi.encodePacked(_password));
}
```

Please note, `keccak256` is one-way hash function, if the password is lost, it can’t be recovered. Additionally storing passwords in plain text, such as hash, can be vulnerable to attacks such as brute force or dictionary attacks. It is recommended to use a salt when hashing passwords to make it more difficult for attackers to predetermine hashes.

## RECOMMENDATION

Use secure hashing algorithms like *bcrypt* which has built-in features to prevent brute-force attacks.

Make sure that ownership of the contract isn’t compromised, an attacker could update the password and access contract functions.

## ACKNOWLEDGEMENT

Project team has added salt hashing to set password. This method is still not full-proof. It is not ideal to store password in the smart contract.

| Identifier | Definition | Severity |
|---|---|---|
| COD-01 | Authorization through `tx.origin` | Minor 🟢 |

Using `tx.origin` for authorization can make the contract vulnerable.

`isHuman` modifier can be useful in preventing certain types of attacks, as it requires that the caller is a human address rather than a smart contract. However, it's important to note that this check can be bypassed by an attacker who is able to create a transaction that originates from a human address but still interacts with the contract in a malicious way.

```
modifier isHuman() {
require(tx.origin == msg.sender, "sorry humans only");
_;
```

## RECOMMENDATION

Avoid authorizations via global variables wherever necessary.

| Identifier | Definition | Severity |
|---|---|---|
| COD-18 | Identical values in `investedAddresses` | Minor 🟢 |

If there are identical addresses in the `investedAddresses` array, it can cause potential issues with the calculations of the total amount invested by each investor.

## RECOMMENDATION

Make sure that the `investedAddresses` array only contains unique addresses.

| Identifier | Definition | Severity |
|---|---|---|
| COD-08 | Lack of fallback function | Minor ● |

Fallback functions are usually executed in one of the following cases: If a function identifier doesn't match any of the available functions in a smart contract. If there was no data supplied along with the function call.

## RECOMMENDATION

It is recommended to include a fallback function that either reverts the transaction or emits an event to inform the contract owner or users that an invalid function or transaction was attempted. This can help to prevent the contract from being misused or exploited.

## RESOLUTION

Creative Wealth team has added fallback function to the code.

| Identifier | Definition | Severity |
|---|---|---|
| COD-10 | Dependencies | Unknown ● |

Trading contract interacts with external applications e.g., https://dashboard.creativewealth.finance, Web3 self-custodial wallet connects, etc. The scope of the audit treats these entities as black boxes and assumes their functional correctness. However, in the real world, they can be compromised, and exploited.

### RECOMMENDATION

Inspect contract dependencies regularly, and mitigate severe impacts whenever necessary.

### ACKNOWLEDGEMENT

Creative Wealth team will inspect contract dependencies periodically.

| Identifier | Definition | Severity |
|---|---|---|
| VOL-02 | Typographical Error | Informational ● |

Typographical errors are found in:

`withdrawl`

## RECOMMENDATION

Fix typographical errors.

## RESOLUTION

Creative Wealth team has fixed aforementioned typographical error.

| Identifier | Definition | Severity |
|---|---|---|
| COM-01 | Floating compiler status | Informational ● |

Compiler is set to `^0.8.0`

## RECOMMENDATION

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

## RESOLUTION

Creative Wealth team has declared compiler version.

| Identifier | Definition | Severity |
|---|---|---|
| COM-04 | Potential resource exhaustion errors | Informational ● |

Array check may throw out of gas errors if it becomes too large:

```
investedAddresses
```

## RECOMMENDATION

Use a mapping to keep track of invested addresses, as it would make lookups faster and more efficient.

# 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