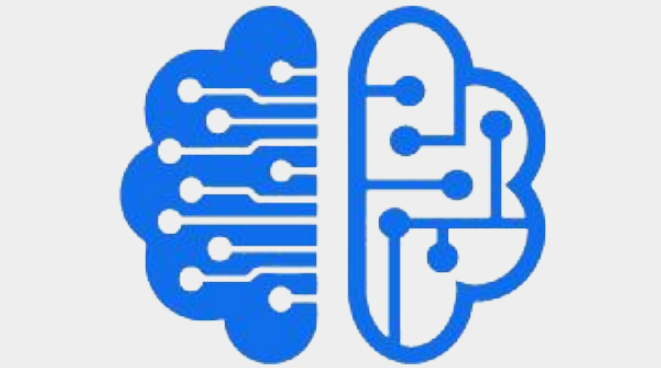

NOBRAiN

# AUDIT SERVICE

This is a limited report on our findings based on our analysis, in accordance with good industry practice as at the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report.

While we have done our best in conducting our analysis and producing this report, it is important to note that you should not should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions.
We go into more detail on this in the below disclaimer below – please make sure to read it in full.

## DISCLAIMER:

By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or prin and/or printed by you.
This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and NOBRAIN and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (NOBRAIN) owe no duty of care towards you or any other person, nor does NOBRAIN make any warranty or representation to any person on the accuracy or completeness of the report.
The report is provided "as is", without a is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and AUDIT NOBRAIN hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report.
Except and only to the extent that it is prohibited by law, NOBRAIN hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim a claim against NOBRAIN, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income,
profits, goodwill, data, contracts, use of money, or business interruption, and
whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report.
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.

## AUDIT DETAILS:

**Audited project:** 4INT
**Contract address:** 0x5ceebb0947d58fabde2fc026ffe4b33ccfe1ba8b
**Languages:** Solidity (Smart contract)
**Platforms and Tools:** Remix IDE, Truffle, Truffle Team, Ganache, Solhint, VScode, Mythril, Contract Library
**Total supply:** 100,000,000
**Token ticker:** 4INT
**Decimals:** 9
**Compiler Version:** v0.8.7+commit.e28d00a7
**Contract Deployer Address:** 0x23ee70e03337cd90faa0ba7057ad8c0b797911f4
**Optimization Enabled:** Yes with 200 runs
**Client contacts:** Forint Finance team
**Blockchain:** Polygon
**Project website:** https://www.forintfinance.com
**The audit items and results:**
(Other unknown security vulnerabilities are not included in the audit responsibility scope)
**Audit Result:** Passed
**Audit Date:** January 03, 2022
**Audit Team:** NOBRAiN
https://nobrain.site/

## INFORMATION

This Audit Report mainly focuses on the overall security of 4INT Smart Contract.
With this report, we have tried to ensure the reliability and correctness of their smart contract by complete and rigorous assessment of their system's architecture and the smart contract codebase.

## AUDITING APPROACH AND METHODOLOGIES APPLIED

The NOBRAIN team has performed rigorous testing of the project starting with analyzing the code design patterns in which we reviewed the smart contract architecture to ensure it is structured and safe use of third-party smart contracts and libraries.
Our team then performed a formal line by line inspection of the Smart Contract to find any potential issue like race conditions, transaction-ordering dependence, timestamp dependence, and denial of service attacks.
In the Unit In the Unit testing Phase, we coded/conducted custom unit tests written for each function in the contract to verify that each function works as expected.
In Automated Testing, we tested the Smart Contract with our in-house developed tools to identify vulnerabilities and security flaws.
The code was tested in collaboration of our multiple team members and this included:

- Testing the functionality of the Smart Contract to determine proper logic has been followed
throughout the whole process.
- Analyzing the complexity of the code in depth and detailed, manual review of the code, line by line.
- Deploying the code on testnet using multiple clients to run live tests.
- Analyzing failure preparations to check how the Smart Contract performs in case of any bugs
and vulnerabilities.
- Checking whether all the libraries used in the code are on the latest version.
- Analyzing the security of the on-chain data.

## AUDIT GOALS

The focus of the audit was to verify that the Smart Contract System is secure, resilient and working according to the specifications.
The audit activities can be grouped in the following three categories:

- Security
  Identifying security related issues within each contract and the system of contract.
- Sound Architecture
  Evaluation of the architecture of this system through the lens of established smart contract best practices and general software best practices.
- Code Correctness and Quality
  A full review of the contract source code. The primary areas of focus include:
  - Accuracy
  - Readability
  - Sections of code with high complexity
  - Quantity and quality of test coverage

## ISSUE CATEGORIES

Every issue in this report was assigned a severity level from the following:

- High level severity issues
  Issues on this level are critical to the smart contract's performance/functionality and should be fixed before moving to a live environment.
- Medium level severity issues
  Issues on this level could potentially bring problems and should eventually be fixed.
- Low level severity issues
  Issues on this l Issues on this level are minor details and warnings that can remain unfixed but would be better fixed at some point in the future

## NUMBER OF ISSUES PER SEVERITY

| Critical | HIgh | Medium | Low | Note |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |

## ISSUES CHECKING STATUS

| N° | Issue description | Checking status |
|---|---|---|
| 1 | Compiler warnings | Passed |
| 2 | Race conditions and Reentrancy. Cross-function race conditions | Passed |
| 3 | Possible delays in data delivery | Passed |
| 4 | Oracle calls | Passed |
| 5 | Front Running | Passed |
| 6 | Timestamp dependence | Passed |
| 7 | Integer Overflow and Underflow | Passed |
| 8 | DoS with Revert | Passed |
| 9 | DoS with block gas limit | Passed |
| 10 | Methods execution permissions | Passed |
| 11 | Economy model | Passed |
| 12 | The impact of the exchange rate on the logic | Passed |
| 13 | Private user data leaks | Passed |
| 14 | Malicious Event log | Passed |
| 15 | Scoping and declarations | Passed |
| 16 | Uninitialized storage pointers | Passed |
| 17 | Arithmetic accuracy | Passed |
| 18 | Design logic | Passed |
| 19 | Cross-functions race conditions | Passed |
| 20 | Safe Zeppelin module | Passed |
| 21 | Fallback function security | Passed |

## MANUAL AUDIT

For this section the code was tested/read line by line by our developers. We also used Remix IDE's JavaScript VM and Kovan networks to test the contract functionality.

## AUTOMATED AUDIT

- Remix Compiler Warnings
  It throws warnings by Solidity's compiler. If it encounters any errors the contract cannot be cannot be compiled and deployed. No issues found.

## OWNER PRIVILEGES

- Owner can transfer ownership of the contract to a new account
- Owner can renounce ownership
- Owner can burn tokens
- Owner can transfer coins other than the token in question from the contract

## CONCLUSION

Smart contracts do not contain any high severity issues!*

*Please check the disclaimer above and note, the audit makes no statements or warranties on business model, investment attractiveness or code sustainability. The report is provided for the only contract mentioned in the report and does not include any other potential contracts deployed by Owner.

7.

## TOKEN

The Forint Finance Token is a utility token fuelling the whole Forint Finance Ecosystem. Allows users to receive a discount on the Forint Finance Fee for Swappy and Dexy, as well as allowing them to have priority access to the IDOs launched in IDOL.

7.1

## SMART CONTRACT

### Polygon Blockchain

```
// SPDX-License-Identifier: MIT


pragma solidity ^0.8.0;

interface NewIERC20 {
    function transfer(address, uint) external returns (bool);
}

interface OldIERC20 {
    function transfer(address, uint) external;
}

interface IERC20 {
    function totalSupply() external view returns (uint256);
    function balanceOf(address account) external view returns (uint256);
    function transfer(address recipient, uint256 amount) external returns (bool);
    function allowance(address owner, address spender) external view returns (uint256);
    function approve(address spender, uint256 amount) external returns (bool);
    function transferFrom(
        address sender,
        address recipient,
        uint256 amount
    ) external returns (bool);
    event Transfer(address indexed from, address indexed to, uint256 value);
    event Approval(address indexed owner, address indexed spender, uint256 value);
}

interface IERC20Metadata is IERC20 {
    function name() external view returns (string memory);
    function symbol() external view returns (string memory);
    function decimals() external view returns (uint8);
}

abstract contract Context {
    function _msgSender() internal view virtual returns (address) {
        return msg.sender;
    }
    function _msgData() internal view virtual returns (bytes calldata) {
        return msg.data;
    }
}
```

```
abstract contract Ownable is Context {
    address private _owner;
        event OwnershipTransferred(address indexed previousOwner, address indexed
newOwner);
    constructor() {
        _transferOwnership(_msgSender());}
    function owner() public view virtual returns (address) {
        return _owner;}
    modifier onlyOwner() {
        require(owner() == _msgSender(), "Ownable: caller is not the owner");
        _;}
    function renounceOwnership() public virtual onlyOwner {
        _transferOwnership(address(0));}
    function transferOwnership(address newOwner) public virtual onlyOwner {
        require(newOwner != address(0), "Ownable: new owner is the zero address");
        _transferOwnership(newOwner);}
    function _transferOwnership(address newOwner) internal virtual {
        address oldOwner = _owner;
        _owner = newOwner;
        emit OwnershipTransferred(oldOwner, newOwner);}
}


contract ERC20 is Context, IERC20, IERC20Metadata {
    mapping(address => uint256) private _balances;
    mapping(address => mapping(address => uint256)) private _allowances;
    uint256 private _totalSupply;
    string private _name;
    string private _symbol;
    constructor(string memory name_, string memory symbol_) {
        _name = name_;
        _symbol = symbol_;
    }
    function name() public view virtual override returns (string memory) {
        return _name;
    }
    function symbol() public view virtual override returns (string memory) {
        return _symbol;
    }
    function decimals() public view virtual override returns (uint8) {
        return 9;
    }
    function totalSupply() public view virtual override returns (uint256) {
        return _totalSupply;
    }
    function balanceOf(address account) public view virtual override returns (uint256) {
        return _balances[account];
    }
      function transfer(address recipient, uint256 amount) public virtual override returns
(bool) {
        _transfer(_msgSender(), recipient, amount);
        return true;
    }
       function allowance(address owner, address spender) public view virtual override
returns (uint256) {
        return _allowances[owner][spender];
    }
```

```
    function approve(address spender, uint256 amount) public virtual override returns (bool) {
        _approve(_msgSender(), spender, amount);
        return true;
    }
    function transferFrom(
        address sender,
        address recipient,
        uint256 amount
    ) public virtual override returns (bool) {
        _transfer(sender, recipient, amount);
        uint256 currentAllowance = _allowances[sender][_msgSender()];
        require(currentAllowance >= amount, "ERC20: transfer amount exceeds allowance");
        unchecked {
            _approve(sender, _msgSender(), currentAllowance - amount);
        }
        return true;
    }
    function increaseAllowance(address spender, uint256 addedValue) public virtual returns (bool) {
        _approve(_msgSender(), spender, _allowances[_msgSender()][spender] + addedValue);
        return true;
    }
    function decreaseAllowance(address spender, uint256 subtractedValue) public virtual returns (bool) {
        uint256 currentAllowance = _allowances[_msgSender()][spender];
        require(currentAllowance >= subtractedValue, "ERC20: decreased allowance below zero");
        unchecked {
            _approve(_msgSender(), spender, currentAllowance - subtractedValue);
        }
        return true;
    }
    function _transfer(
        address sender,
        address recipient,
        uint256 amount
    ) internal virtual {
        require(sender != address(0), "ERC20: transfer from the zero address");
        require(recipient != address(0), "ERC20: transfer to the zero address");
        _beforeTokenTransfer(sender, recipient, amount);
        uint256 senderBalance = _balances[sender];
        require(senderBalance >= amount, "ERC20: transfer amount exceeds balance");
        unchecked {
            _balances[sender] = senderBalance - amount;
        }
        _balances[recipient] += amount;
        emit Transfer(sender, recipient, amount);
        _afterTokenTransfer(sender, recipient, amount);
    }
    function _mint(address account, uint256 amount) internal virtual {
        require(account != address(0), "ERC20: mint to the zero address");
        _beforeTokenTransfer(address(0), account, amount);
        _totalSupply += amount;
        _balances[account] += amount;
        emit Transfer(address(0), account, amount);
        _afterTokenTransfer(address(0), account, amount);
    }
```

```
    function _burn(address account, uint256 amount) internal virtual {
        require(account != address(0), "ERC20: burn from the zero address");
        _beforeTokenTransfer(account, address(0), amount);
        uint256 accountBalance = _balances[account];
        require(accountBalance >= amount, "ERC20: burn amount exceeds balance");
        unchecked {
            _balances[account] = accountBalance - amount;
        }
        _totalSupply -= amount;
        emit Transfer(account, address(0), amount);
        _afterTokenTransfer(account, address(0), amount);
    }
    function _approve(
        address owner,
        address spender,
        uint256 amount
    ) internal virtual {
        require(owner != address(0), "ERC20: approve from the zero address");
        require(spender != address(0), "ERC20: approve to the zero address");
        _allowances[owner][spender] = amount;
        emit Approval(owner, spender, amount);
    }
    function _beforeTokenTransfer(
        address from,
        address to,
        uint256 amount
    ) internal virtual {}
    function _afterTokenTransfer(
        address from,
        address to,
        uint256 amount
    ) internal virtual {}
}

abstract contract ERC20Burnable is Context, ERC20 {
    function burnFrom(address account, uint256 amount) public virtual {
        uint256 currentAllowance = allowance(account, _msgSender());
        require(currentAllowance >= amount, "ERC20: burn amount exceeds allowance");
        unchecked {
            _approve(account, _msgSender(), currentAllowance - amount);
        }
        _burn(account, amount);
    }
}

abstract contract ReentrancyGuard {
    uint256 private constant _NOT_ENTERED = 1;
    uint256 private constant _ENTERED = 2;
    uint256 private _status;
    constructor() {
        _status = _NOT_ENTERED;
    }
    modifier nonReentrant() {
        require(_status != _ENTERED, "ReentrancyGuard: reentrant call");
        _status = _ENTERED;
        _;
        _status = _NOT_ENTERED;
    }
}
```

```
contract Forint is ERC20Burnable, Ownable, ReentrancyGuard {
    constructor(
        string memory name,
        string memory symbol,
        uint256 initialSupply,
        address owner
    ) ERC20(name, symbol) {
        _mint(owner, initialSupply);
    }
    function transferAnyNewERC20Token(address _tokenAddr, address _to, uint _amount)
public onlyOwner {
            require(NewIERC20(_tokenAddr).transfer(_to, _amount), "Could not transfer out
tokens!");
    }
     function transferAnyOldERC20Token(address _tokenAddr, address _to, uint _amount)
public onlyOwner {
        OldIERC20(_tokenAddr).transfer(_to, _amount);
    }
}
```

## IMPLEMENTED EVENTS

event Transfer(address indexed from, address indexed to, uint256 value)
event Approval(address indexed owner, address indexed spender, uint256 value)
event OwnershipTransferred(address indexed previousOwner, address indexed newOwner)

## IMPLEMENTED FUNCTIONS

function transfer(address, uint)
function totalSupply()
function balanceOf(address account)
function transfer(address recipient, uint256 amount)
function allowance(address owner, address spender)
function approve(address spender, uint256 amount)
function transferFrom(address sender, address recipient, uint256 amount)
function name()
function symbol()
function decimals()
function _msgSender()
function _msgData()
function owner()
function renounceOwnership()
function transferOwnership(address newOwner)
function _mint(address account, uint256 amount)
function _burn(address account, uint256 amount)
function _approve(address owner,address spender, uint256 amount )
function _beforeTokenTransfer(address from, address to, uint256 amount)
function _afterTokenTransfer(address from, address to, uint256 amount)

## WEBSITE AUDIT

| | |
|---|---|
| Address | https://forintfinance.com/ |
| Domain registration | 1 years |
| Domain ID | 2642284972_DOMAIN_COM-VRSN |
| Web server | Contabo |
| Server is located in | GB |
| Server response time | 0.41 sec |
| SSL certificate | Yes |
| JavaScripts errors | Not found |
| Typos, or grammatical errors | Not found |
| Issues with code and style | Not found |
| Malware | Not found |
| Injected spam | Not found |
| Internal server errors | Not found |
| Popups | Not found |
| Blocking files | Not found |
| Mobile Friendly | Yes |
| Compress CSS files | Optimized |
| Compress JS files | Optimized |
| Image compression | Optimized |
| Visible content | Optimized |
| Social Media/Contacts | Yes |
| Roadmap | Yes |