# TuttiFrutti

## TuttiFruttiFinance

### Security Assessment

April 16th, 2021

**Audited By**:

Adrian Hetman @ CertiK

adrian.hetman@certik.org

**Reviewed By**:

Alex Papageorgiou @ CertiK

alex.papageorgiou@certik.org

## Disclaimer

CertiK reports are not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security review.

CertiK Reports do not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

CertiK Reports should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

CertiK Reports represent an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

## What is a CertiK report?

- A document describing in detail an in depth analysis of a particular piece(s) of source code provided to CertiK by a Client.
- An organized collection of testing results, analysis and inferences made about the structure, implementation and overall best practices of a particular piece of source code.
- Representation that a Client of CertiK has completed a round of auditing with the intention to increase the quality of the company/product's IT infrastructure and or source code.

# Overview

## Project Summary

| | |
|---|---|
| **Project Name** | TuttiFrutti - TuttiFruttiFinance |
| **Description** | Fork of SushiSwap |
| **Platform** | Ethereum; Solidity, Yul |
| **Codebase** | GitHub Repository |
| **Commits** | 1. 7eb9be2afe3a790b5ceb9211690dc3c3ab6b8eab<br>2. f7f037fcf6cf6cb0204a966c0ff0683c58760936 |

## Audit Summary

| | |
|---|---|
| **Delivery Date** | April 16th, 2021 |
| **Method of Audit** | Static Analysis, Manual Review |
| **Consultants Engaged** | 1 |
| **Timeline** | April 13th, 2021 - April 16th, 2021 |

## Vulnerability Summary

| | |
|---|---|
| **Total Issues** | 20 |
| ● **Total Critical** | 1 |
| ● **Total Major** | 0 |
| ● **Total Medium** | 1 |
| ● **Total Minor** | 7 |
| ● **Total Informational** | 11 |

## Executive Summary

We were tasked with auditing the Tutti Frutti Finance, which is using modified contracts of PancakeSwap and Synthetix. master.sol is modified version of MasterChef.sol and retirement-fund.sol is modified version of StakingRewards.sol from Synthetix.

First issue is with the naming conventions of the contracts and corresponding files. Filenames doesn't have the same name as the contract name, file names are also duplicated which can lead to wrongly importing the file to the contract and If a codebase has two contracts the similar names, the compilation artifacts will not contain one of the contracts with the duplicate name. We would strongly advise to follow solidity style guide and name contracts and file names accordingly to its function, purpose, and follow style guides. This issue is present across all of the codebase.

Second issue with code quality is lack of modularisation of the interfaces and contracts. In cases of Uniswap interfaces, bep20 contract, every used interface and contract is in one file.  Things like this should be separated to allow for better code quality and readability.
Third Issue is left-over contracts that are not used or utilised in the main/supporting contracts, like `Uniswap-v2.sol` , `token-timelock` , `timelock.sol` , `token/bep20.sol` , `poo.sol` and `address.sol` . We are advising to remove such contracts and leave only those who are indeed needed.

Major issue is with left-over code that leads to vulnerabilities and potential exploitation from the owner side. In the `master.sol` contract there is a function `salvage()` that enables an owner to transfer any tokens from the contract to himself. We strongly advise to remove this function as they do not add any needed functionality to the codebase and in case of lost access to the contract, malicious actor could steal user funds.

One important thing to note is the fact the code in question is already deployed on BSC and any of the changes and fixes were done only on the repository.

After the TuttiFrutti team reviewed CertiK's Preliminary Report they have opted to deploy "controller.sol" contract which should be set as the owner of the deployed master contract on BSC. `controller.sol` doesn't have access to the vulnerable function like `salvage()` and can only renounce it's ownership. Below is transaction history showcasing the deployment and ownership transfer to the controller contract.

Controller contract deploy:

https://bscscan.com/tx/0xecce2b94544f69cebe81f020568a98f4046631e0b2b098735266b7728c21a25e

Ownership transfer to controller on master:

https://bscscan.com/tx/0x39da6d5d0109476fe349dd26e3f45b541c1f62f523ed082b9e9bf550600d6167

Set owner of controller to multisig safe:

https://bscscan.com/tx/0x0e40406af2fd84ab4246fcb576d8ad152aafd13a95f33815575401b3be7e8157

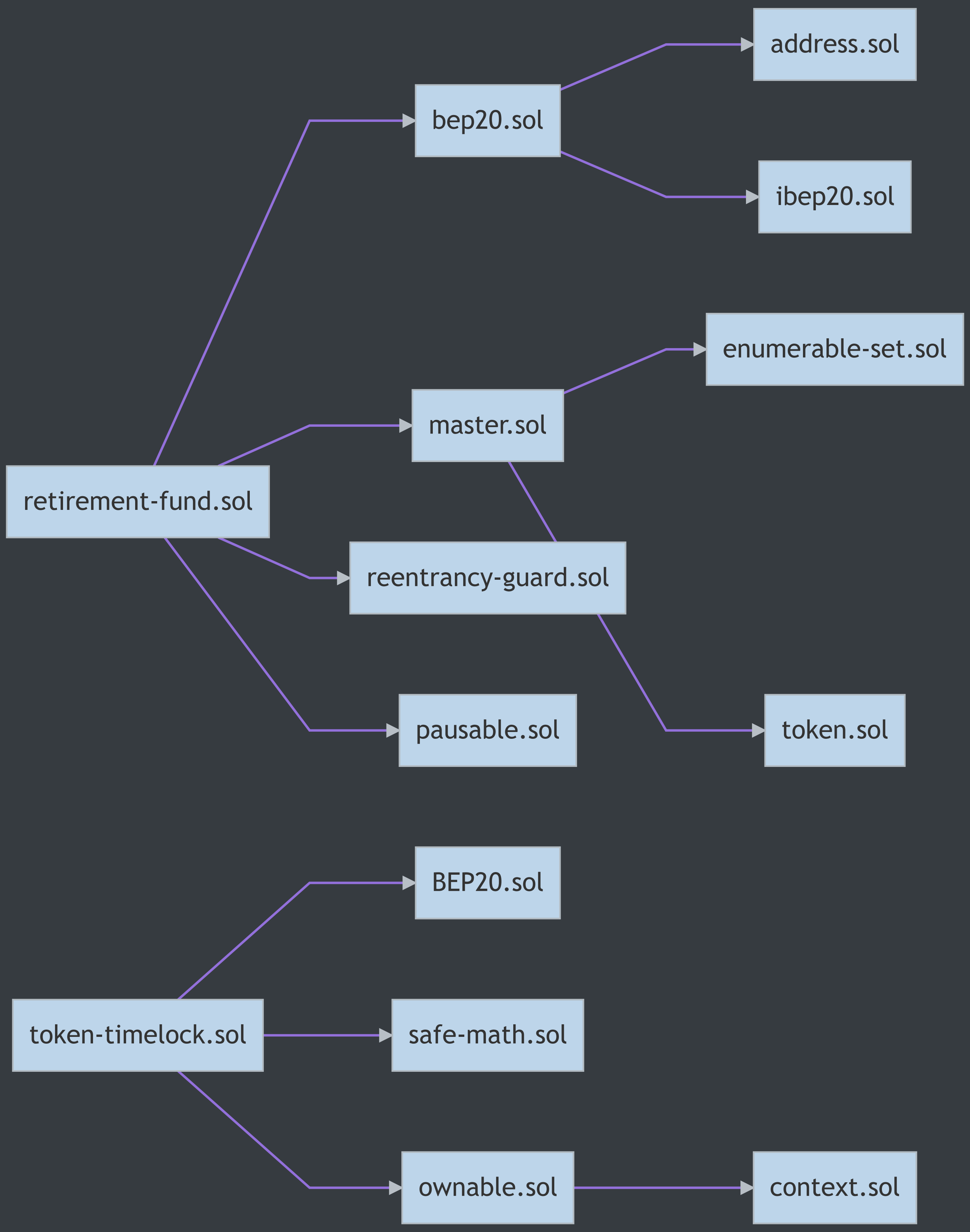
## System Analysis

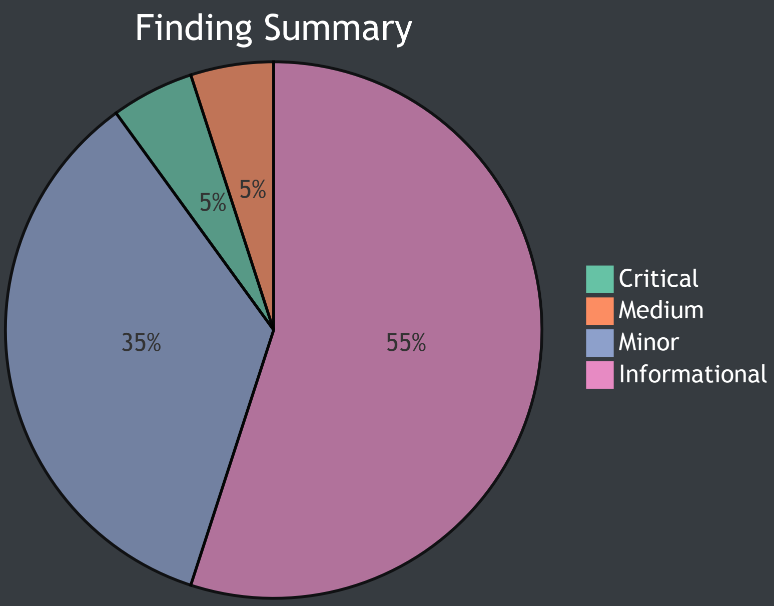
We have found many usage of `onlyOwner` modifier usage in the `master.sol`. Many contract parameters can be changed by the owner at will. In case of lost access to the private key of an account or mishandling security of private keys, an attacker could benefit from that and replace key parameters. We advise that a governance system or multi-signature wallet is utilized instead of a single account in this case.

## Files In Scope

| ID | Contract | Location |
|---|---|---|
| MAT | master.sol | contracts/master.sol |
| RET | retirement-fund.sol | contracts/retirement-fund.sol |
| TON | token-timelock.sol | contracts/token-timelock.sol |
| TOK | token.sol | contracts/token.sol |
| MAS | master.sol | contracts/interfaces/master.sol |
| POO | pool.sol | contracts/interfaces/pool.sol |
| TIM | timelock.sol | contracts/interfaces/timelock.sol |
| TOE | token-timelock.sol | contracts/interfaces/token-timelock.sol |
| UNI | uniswap-v2.sol | contracts/interfaces/uniswap-v2.sol |
| ADD | address.sol | contracts/libs/address.sol |
| BEP | bep20.sol | contracts/libs/bep20.sol |
| CON | context.sol | contracts/libs/context.sol |
| ENU | enumerable-set.sol | contracts/libs/enumerable-set.sol |
| IBE | ibep20.sol | contracts/libs/ibep20.sol |
| OWN | ownable.sol | contracts/libs/ownable.sol |
| PAU | pausable.sol | contracts/libs/pausable.sol |
| REE | reentrancy-guard.sol | contracts/libs/reentrancy-guard.sol |
| SAF | safe-math.sol | contracts/libs/safe-math.sol |
| BE2 | bep20.sol | contracts/token/bep20.sol |

# File Dependency Graph




Finding Summary
5%
5%
35%
55%
Critical
Medium
Minor
Informational

## Manual Review Findings

| ID | Title | Type | Severity | Resolved |
|---|---|---|---|---|
| MAT-01 | Possible drain of all funds by the owner | Volatile Code | 🔴 Critical | ✓ |
| MAT-02 | Centralization concern | Control Flow | 🟡 Medium | ✓ |
| MAT-03 | Checks-effect-pattern not applied | Volatile Code | 🔵 Minor | ✓ |
| MAT-04 | Requisite Value of ERC-20 \`transferFrom()\` / \`transfer()\` Call | Logical Issue | 🔵 Minor | ✓ |
| MAT-05 | External contract addresses should be set once. | Volatile Code | 🔵 Minor | ✓ |
| MAT-06 | uint256 variables cannot be negative | Gas Optimization | 🟢 Informational | ✓ |
| MAT-07 | Lack of emitted Events | Coding Style | 🟢 Informational | ✓ |
| RET-01 | Unnecessary split of math operation | Coding Style | 🟢 Informational | ✓ |
| TON-01 | Only beneficiary get's all the tokens | Volatile Code | 🔵 Minor | ✓ |
| TON-02 | Name of the contract is misleading | Coding Style | 🟢 Informational | ✓ |
| TOK-01 | Owner gets unfair advantage | Volatile Code | 🔵 Minor | ✓ |
| MAS-01 | Interface didn't had all functions | Compiler Error | 🔵 Minor | ⓘ |
| POO-01 | Redundant contract | Dead Code | 🟢 Informational | ✓ |
| UNI-01 | Each interface should be in seperate file | Coding Style | 🟢 Informational | ✓ |

| | | | | |
|---|---|---|---|---|
| UNI-02 | Redundant contract | Dead Code | 🟢 Informational | ✓ |
| BEP-01 | Each interface should be in seperate file | Coding Style | 🟢 Informational | ⓘ |
| BE2-01 | Un-initialized variable | Volatile Code | 🔵 Minor | ✓ |
| BE2-02 | Whitelist functionality should be used in `_beforeTokenTransfer` | Coding Style | 🟢 Informational | ✓ |
| BE2-03 | Redundant Statements | Dead Code | 🟢 Informational | ✓ |
| BE2-04 | Redundant contract | Dead Code | 🟢 Informational | ✓ |

## MAT-01: Possible drain of all funds by the owner

| Type | Severity | Location |
|---|---|---|
| Volatile Code | ● Critical | master.sol L547-L553 |

### Description:

Owner of the contract can use `salvage()` function to drain all tokens that are locked in the contract.

### Recommendation:

This function needs to be removed as it open a critical vulnerability in the contract. A malicious owner or in a case of loss access to the owner account, a malicious actor could drain all of the funds.
In case of needed to do an emergency withdrawal, appropriate function already exist in the contract.

### Alleviation:

The salvage function has been removed from the code and the team introduced the "controller.sol" contract which should be set as the owner of the deployed master contract which is live on BSC.

## MAT-02: Centralization concern

| Type | Severity | Location |
|---|---|---|
| Control Flow | ● Medium | master.sol L510-L579 |

### Description:

Owner has too much power over most important addresses used in the contract. In case of lost access to the private key of an account or mishandling security of private keys, an attacker could benefit from that and exploit the protocol.

### Recommendation:

Mentioned functions should be called by governance or be handled by multi-sig wallet.

### Alleviation:

The owner of the newly introduced controller contract will be set to a multi-sig contract provided by Gnosis Safe on Binance Smart Chain (address: 0xCfe31BC7D0250883be891bFfF78f97A0e14E5b96).

## MAT-03: Checks-effect-pattern not applied

| Type | Severity | Location |
|---|---|---|
| Volatile Code | ● Minor | master.sol L368-L372, L415-L417 |

### Description:

State variables are changed after `pool.lpToken.safeTransfer()` function call.

### Recommendation:

It is recommended to follow checks-effects-interactions pattern for cases like this.
It shields public functions from re-entrancy attacks. It's always a good practie to follow this pattern. `checks-effects-interaction` pattern also applies to ERC20 tokens as they can inform the recipient of a transfer in certain implementations.

### Alleviation:

Issue has been resolved. State variables are written before `pool.lpToken.safeTransfer()`

## MAT-04: Requisite Value of ERC-20 `transferFrom()` / `transfer()` Call

| Type | Severity | Location |
|---|---|---|
| Logical Issue | 🔵 Minor | master.sol L502, L504 |

### Description:

While the ERC-20 implementation does necessitate that the `transferFrom()` / `transfer()` function returns a `bool` variable yielding `true`, many token implementations do not return anything i.e. Tether (USDT) leading to unexpected halts in code execution.

### Recommendation:

We advise that the `SafeERC20.sol` library is utilized by OpenZeppelin to ensure that the `transferFrom()` / `transfer()` function is safely invoked in all circumstances.

### Alleviation:

Issue has been resolved. `safeTransfer` is utlized instead of `transfer()`

## MAT-05: External contract addresses should be set once.

| Type | Severity | Location |
|---|---|---|
| Volatile Code | ● Minor | master.sol L533-L543 |

### Description:

Crucial addresses that are important for the overall protocol stability and functionality should be only set once to avoid swaping address to the malicious one and risking draining funds form the user.

### Recommendation:

We would recommend to set linked addresses only once. As mentioned in the previous issue, in case of lost access to the private key of an owner account, an attacker could swap the addresses and exploit protocol.
For the chance to upgrade the contract we would advise to implement proxy pattern from OpenZeppelin for an upgradable contracts if there is a concern of a pottential issues in implementation of said addresses.

### Alleviation:

Issue partially resolved. See client's comment:
"The treasury & rewards contract can only be changed by the current treasury and/or rewards contract. Therefore it is the responsibility of the party which is in control of these keys to act accordingly and secure their access. We will suggest that the parties use a multi-sig solution.
The fund address has been changed so it can only be set once, since it will never change after that."
Rest of the set functions will be controller by the `controller.sol` contract which will be set to a multi-sig contract provided by Gnosis Safe on Binance Smart Chain (address: 0xCfe31BC7D0250883be891bFfF78f97A0e14E5b96).

## MAT-06: uint256 variables cannot be negative

| Type | Severity | Location |
|---|---|---|
| Gas Optimization | ● Informational | master.sol L378 |

### Description:

Due to nature of uint256 type following check `require(user.claimAmount<=0)` should only check if it's equal 0 as uint256 cannot be negative number.

### Recommendation:

We would recommend updating the linked requite statement to only check if claimAmount is equal to 0.

### Alleviation:

Issue has been resolved. Recommendation applied.

## MAT-07: Lack of emitted Events

| Type | Severity | Location |
|---|---|---|
| Coding Style | ● Informational | master.sol L510-L579 |

### Description:

During execution of linked function there is a lack of emitted events that would notify the front end or listeners about occured changes of important states.

### Recommendation:

We would recommend adding and emitting appropriate events to the linked functions.

### Alleviation:

Issue has been resolved. Events are now emitted.

## RET-01: Unnecessary split of math operation

| Type | Severity | Location |
|---|---|---|
| Coding Style | ● Informational | retirement-fund.sol L149, L151 |

### Description:

Linked statements could be combined. There is no reason to split them into two statements.

### Recommendation:

We would recommend joining the two statements into one.

### Alleviation:

Issue has been resolved. Recommendation was applied.

## TON-01: Only beneficiary get's all the tokens

| Type | Severity | Location |
|---|---|---|
| Volatile Code | ● Minor | token-timelock.sol L74 |

### Description:

Only single address, beneficiary would benefit from this contract and its functionality.

### Recommendation:

We would advise to add more beneficiaries as the vesting could benefit more users.

### Alleviation:

Issue has been resolved. Instead of one beneficiary bein set, now multiple of beneficiaries can be set in the contract.

## TON-02: Name of the contract is misleading

| Type | Severity | Location |
|---|---|---|
| Coding Style | ● Informational | token-timelock.sol L15 |

### Description:

Contract's logic is very similar to simple vesting mechanism and Timelock is mostly associated with Governance module.

### Recommendation:

We would advise to change the name of the file and the contract to better showcase the intended functionality of the code.

### Alleviation:

Issue has beeb resolved. Contract name has been changed to vesting-schedule.sol

## TOK-01: Owner gets unfair advantage

| Type | Severity | Location |
|---|---|---|
| Volatile Code | ● Minor | token.sol L12 |

### Description:

With minting this amount of tokens to himself, an owner have unfair advantage in the system he needs to monitor.

### Recommendation:

We would advise not to mint staking tokens to a single address. Instead of having a single address, multi-sig or Governance system in place would help mitigating this issue in case of malicious owner or lost access to the owner keys.

### Alleviation:

Issue has been resolved and now token mint is done to several addresses.
Client's comment:
"Token mint is now more customizable and adherent to the project whitepaper immediately on initialization."

## MAS-01: Interface didn't had all functions

| Type | Severity | Location |
|---|---|---|
| Compiler Error | ● Minor | master.sol L4-L8 |

### Description:

IMaster interface didn't had all of the functions declared. It is missing `function available(address) external view returns (uint256);`.

### Recommendation:

We would advise to add `function available(address) external view returns (uint256);` to the interface to allow the project to compile and contracts whhich rely on the interface to work properly.

### Alleviation:

The TuttiFrutti - TuttiFruttiFinance development team has acknowledged this exhibit but decided to not apply its remediation in the current version of the codebase due to time constraints.

## POO-01: Redundant contract

| Type | Severity | Location |
|---|---|---|
| Dead Code | ● Informational | pool.sol General |

### Description:

The linked file do not affect the functionality of the codebase and appear to be either leftovers from test code or older functionality.

### Recommendation:

We advise that they are removed to better prepare the code for production environments.

### Alleviation:

Issue has been resolved. Contract is removed.

## UNI-01: Each interface should be in seperate file

| Type | Severity | Location |
|---|---|---|
| Coding Style | ● Informational | uniswap-v2.sol L4, L81, L188 |

### Description:

Each interface should be in seperate file. Not always it's needed to import everything into a contract. Having a code modularity helps keeps things organized.

### Recommendation:

We would advise to split each interface into seperate file.

### Alleviation:

Issue has been alleviated as the contract as a whole has been removed due to suggestion in UNI-02M.

## UNI-02: Redundant contract

| Type | Severity | Location |
|---|---|---|
| Dead Code | ● Informational | uniswap-v2.sol General |

### Description:

The linked file do not affect the functionality of the codebase and appear to be either leftovers from test code or older functionality.

### Recommendation:

We advise that they are removed to better prepare the code for production environments.

### Alleviation:

Issue has been resolved. Contras has been removed.

## BEP-01: Each interface should be in seperate file

| Type | Severity | Location |
|---|---|---|
| Coding Style | ● Informational | bep20.sol L10, L87, L248, L532 |

### Description:

Each interface/contract should be in seperate file. Not always it's needed to import everything into a contract. Having a code modularity helps keeps things organized.

### Recommendation:

We would advise to split each interface/contract into seperate file.

### Alleviation:

The TuttiFrutti - TuttiFruttiFinance development team has acknowledged this exhibit but decided to not apply its remediation in the current version of the codebase due to time constraints.

## BE2-01: Un-initialized variable

| Type | Severity | Location |
|---|---|---|
| Volatile Code | ● Minor | bep20.sol L40 |

### Description:

Linked variable isn't initialized anywhere and is being used in other parts of the contract e.g. _transfer() function.

### Recommendation:

We would advise to add functions that can add to the mapping and remove whitelisted addresses.

### Alleviation:

Issue has been alleviated as the contract has been removed due to recommendation on BE2-04M

## BE2-02: Whitelist functionality should be used in `_beforeTokenTransfer`

| Type | Severity | Location |
|---|---|---|
| Coding Style | ● Informational | bep20.sol L245-L251 |

### Description:

Added functionality of whitelisted addresses should be utilized inside _beforeTokenTransfer and revert if additional requirements aren't met.

### Recommendation:

We would advise to move linked code block to the _beforeTokenTransfer function.

### Alleviation:

Issue has been alleviated as the contract has been removed as suggested in BE2-04M

## BE2-03: Redundant Statements

| Type | Severity | Location |
|---|---|---|
| Dead Code | ● Informational | bep20.sol L248 |

### Description:

The linked statements do not affect the functionality of the codebase and appear to be either leftovers from test code or older functionality.

### Recommendation:

We advise that they are removed to better prepare the code for production environments.

### Alleviation:

Issue has been alleviated as the contract has been removed as suggested in BE2-04M

## BE2-04: Redundant contract

| Type | Severity | Location |
|---|---|---|
| Dead Code | ● Informational | bep20.sol General |

### Description:

The linked file do not affect the functionality of the codebase and appear to be either leftovers from test code or older functionality.

### Recommendation:

We advise that they are removed to better prepare the code for production environments.

### Alleviation:

Issue has been resolved as contract has been removed.

## Appendix

### Finding Categories

#### Gas Optimization

Gas Optimization findings refer to exhibits that do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

#### Logical Issue

Logical Issue findings are exhibits that detail a fault in the logic of the linked code, such as an incorrect notion on how `block.timestamp` works.

#### Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

#### Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

#### Coding Style

Coding Style findings usually do not affect the generated byte-code and comment on how to make the codebase more legible and as a result easily maintainable.

#### Compiler Error

Compiler Error findings refer to an error in the structure of the code that renders it impossible to compile using the specified version of the project.

## Dead Code

Code that otherwise does not affect the functionality of the codebase and can be safely omitted.