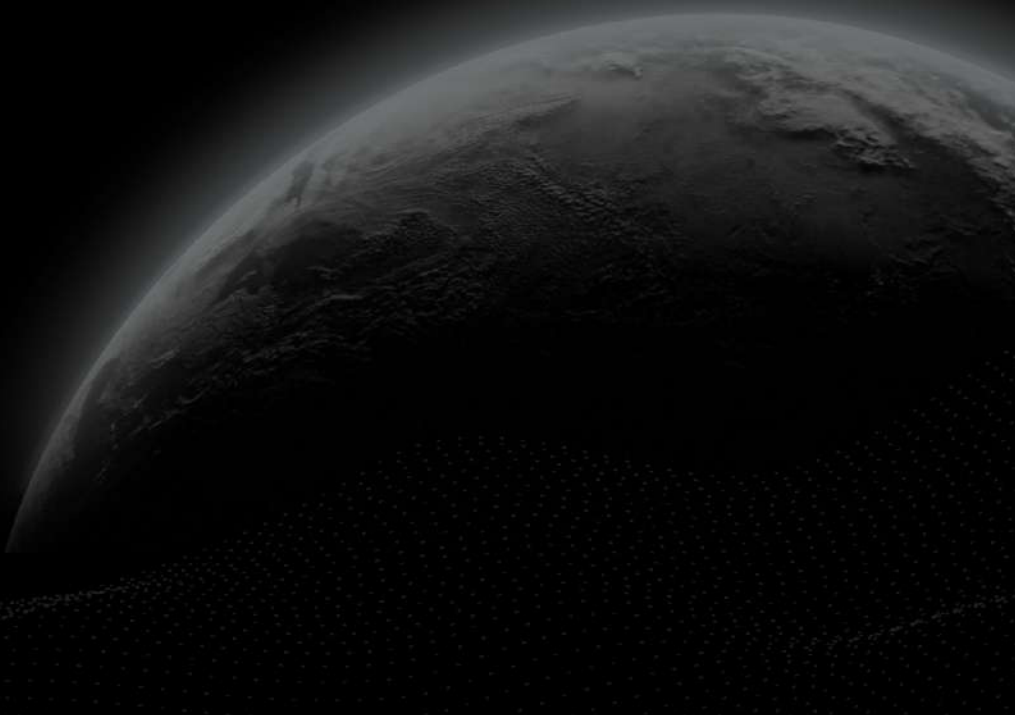




Security Assessment

**GrowFi**

CertiK Verified on Apr 10th, 2025





Certik Verified on Apr 10th, 2025

## GrowFi

The security assessment was prepared by Certik, the leader in Web3.0 security.

### Executive Summary

#### TYPES

DeFi

#### ECOSYSTEM

Base

#### METHODS

Manual Review, Static Analysis

#### LANGUAGE

Solidity

#### TIMELINE

Delivered on 03/19/2025

#### KEY COMPONENTS

N/A

#### CODEBASE

<https://basescan.org/address/0xd8bd12f002ae1ff834bb580a6f8a34585fb26dae>

### Vulnerability Summary



9

Total Findings

31

Resolved

0

Mitigated

0

Partially Resolved

9

Acknowledged

0

Declined

0

Unresolved



0

Critical

Critical risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.



0

Major

Major risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.



0

Medium

Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.



0

Minor

Minor risks can be any of the above, but on a smaller scale. They generally do not compromise the overall integrity of the project, but they may be less efficient than other solutions.



0

Informational

Informational errors are often recommendations to improve the style of the code or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

# TABLE OF CONTENTS | GROWFI

## I **Summary**

[Executive Summary](#)

[Vulnerability Summary](#)

[Codebase](#)

[Audit Scope](#)

[Approach & Methods](#)

## I **Findings**

[GRO-01: Floating Pragma](#)

[GRO-02: Uninitialized State Variables](#)

[GRO-03: Tautology or Contradiction](#)

[GRO-04: Unchecked Transfer](#)

[GRO-05: Uninitialized Local Variables](#)

[GRO-06: Missing Zero Address Validation](#)

[GRO-07: Incorrect Solidity Version](#)

[GRO-08: State Variables Should be Declared Constant](#)

[GRO-09: Public Functions Should be Declared External](#)

[GRO-13: No unchecked call responses found](#)

[GRO-14: No vulnerable self-destruct functions found No assertion vulnerabilities found](#)

[GRO-15: No old solidity code found](#)

[GRO-16: No external delegated calls found No external call dependency found](#)

[GRO-17: No vulnerable authentication calls found](#)

## I **Optimizations**

[GRO-11: Variables That Could Be Declared as Immutable](#)

[GRO-12: Function Should Be Declared External](#)

[GRO-10: Variable Could be Declared as `constant`](#)

## I **Appendix**

## I **Disclaimer**


# CODEBASE | GROWFI

## Repository

<https://basescan.org/address/0xd8bd12f002ae1ff834bb580a6f8a34585fb26dae>

# AUDIT SCOPE | GROWFI

1 file audited ● 1 file with Acknowledged findings

ID	File	SHA256 Checksum
● GRO	 GROWFI.sol	fbdc32d7c2e53e0674e07abf93790df22f1fe10 5d08f064e8260076a5c478c3b

## APPROACH & METHODS | GROWFI

This report has been prepared for Grow to discover issues and vulnerabilities in the source code of the Grow project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Manual Review and Static Analysis techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Testing the smart contracts against both common and uncommon attack vectors;
- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

# FINDINGS | GROWFI



9

Total Findings

0

Critical

0

Major

0

Medium

0

Minor

9

Informational

This report has been prepared to discover issues and vulnerabilities for Growfi. Through this audit, we have uncovered 9 issues ranging from different severity levels. Utilizing the techniques of Manual Review & Static Analysis to complement rigorous manual code reviews, we discovered the following findings:

ID	Title	Category	Severity	Status
GRO-01	Floating Pragma	Descentralization / Privilege	Informational	● Acknowledged
GRO-02	Uninitialized State Variables	Descentralization / Privilege	Informational	● Acknowledged
GRO-03	Tautology or Contradiction	Volatile Code	Informational	● Acknowledged
GRO-04	Unchecked Transfer	Language Specific	Informational	● Acknowledged
GRO-05	Uninitialized Local Variables	Logical Issue	Informational	● Acknowledged
GRO-06	Missing Zero Address Validation	Logical Issue	Informational	● Acknowledged
GRO-07	Incorrect Solidity Version	Coding Style	Informational	● Acknowledged
GRO-08	State Variables Should be Declared Constant	Language Specific	Informational	● Acknowledged
GRO-09	Public Functions Should be Declared External	Coding Style	Informational	● Acknowledged
GRO-13	No unchecked call responses found	Inconsistency	Informational	● Acknowledged
GRO-14	No vulnerable self-destruct functions found No assertion vulnerabilities found	Coding Style	Informational	● Acknowledged

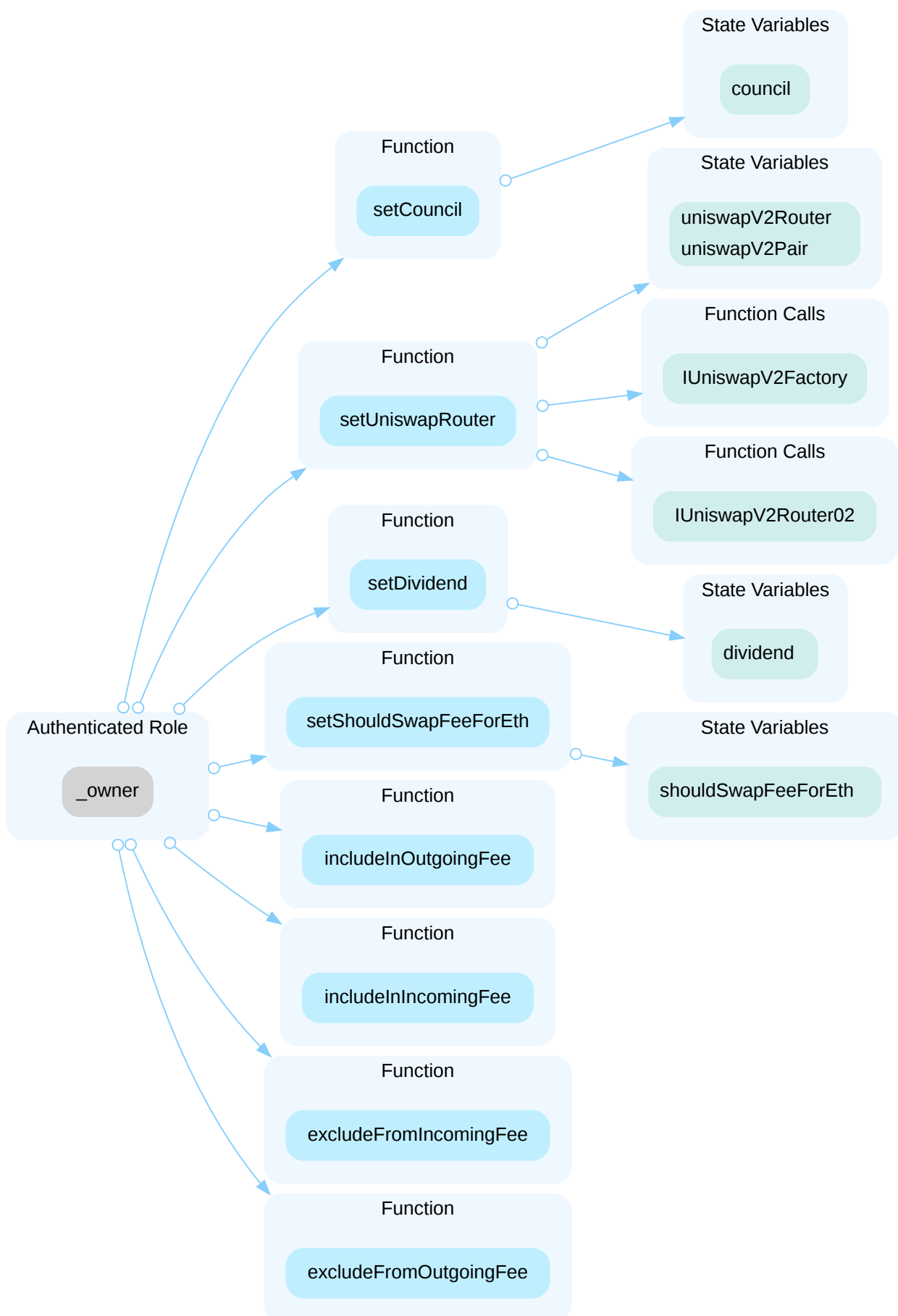
ID	Title	Category	Severity	Status
GRO-15	No old solidity code found	Coding Style	Informational	● Acknowledged
GRO-16	No external delegated calls found No external call dependency found	Volatile Code	Informational	● Acknowledged
GRO-17	No vulnerable authentication calls found	Coding Style	Informational	● Acknowledged

## GRO-01 | Floating Pragma

Category	Severity	Location	Status
Descentralization / Privilege	● Informational	GROWFI.sol: 414, 422, 509, 517, 524, 542, 549, 567, 581, 592	● Acknowledged

### Description

In the contract `GROWFI` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and modify different token settings such as the UniSwap (PancakeSwap) router, dividends and fees.



Furthermore, the role `_owner` has authority over the functions `renounceOwnership()` and `transferOwnership()`. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and renounce and transfer ownership at any time.

## Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multisignature wallets. Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

### Short Term:

Timelock and Multi sign ( $\frac{2}{3}$ ,  $\frac{3}{5}$ ) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;  
AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;  
AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

### Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;  
AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.  
AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

### Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles.  
OR

- Remove the risky functionality.

## **I Alleviation**

### **[Growfi Team]**

According to the protocol outlined in the lite papers, adjustments might need to be made to fee wallets. Ownership is required to execute the functions of the contract.

## GRO-02 | Uninitialized State Variables

Category	Severity	Location	Status
Descentralization / Privilege	● Informational	GROWFI.sol: 493	● Acknowledged

### Description

All of the `USDC` tokens are sent to the contract deployer when deploying the contract. This could be a centralization risk as the deployer can distribute `USDC` tokens without obtaining the consensus of the community.

### Recommendation

We recommend the team to be transparent regarding the initial token distribution process, and the team shall make enough efforts to restrict the access of the private key.

### Alleviation

#### [Growfi Team]

According to the protocol outlined in the lite papers, token is distributed on an ongoing basis. These distributions can be monitored on the blockchain observer.

## GRO-03 | Tautology or Contradiction

Category	Severity	Location	Status
Volatile Code	● Informational	GROWFI.sol: 472, 476~478	● Acknowledged

### Description

The contract is serving as the underlying entity to interact with one or more third party protocols. The scope of the audit treats third party entities as black boxes and assume their functional correctness. However, in the real world, third parties can be compromised and this may lead to lost or stolen assets. In addition, upgrades of third parties can possibly create severe impacts, such as increasing fees of third parties, migrating to new LP pools, etc.

The contract `GROWFI` interacts with third party contract with `IUniswapV2Router02` interface via `uniswapV2Router` :

```
472     IUniswapV2Router02 public uniswapV2Router;
```

### Recommendation

We understand that the business logic requires interaction with the third parties. We encourage the team to constantly monitor the statuses of third parties to mitigate the side effects when unexpected activities are observed.

### Alleviation

#### [Growfi Team]

Third party functionality is monitored regularly.

## GRO-04 | Unchecked Transfer

Category	Severity	Location	Status
Language Specific	● Informational	GROWFI.sol: 504	● Acknowledged

### Description

The contract has one payable function, but does not have a function to withdraw the funds.

### Recommendation

We recommend removing the `payable` attribute or adding a withdraw function.

### Alleviation

#### [Growfi Team]

As the contract is immutable, removing the payable function is not an option. Future D.E.B.T. projects will consider removing this option.

## GRO-05 | Uninitialized Local Variables

Category	Severity	Location	Status
Logical Issue	● Informational	GROWFI.sol: 702	● Acknowledged

### Description

The `approve` function could be used in an attack that allows a spender to transfer more tokens than the owner of the tokens ever wanted to allow the spender to transfer.

Here is a possible attack scenario:

Alice allows Bob to transfer N of Alice's tokens ( $N > 0$ ) by calling `approve()` method on `Grow` smart contract passing Bob's address and N as method arguments. After some time, Alice decides to change from N to M ( $M > 0$ ) the number of Alice's tokens Bob is allowed to transfer, so she calls `approve` method again, this time passing Bob's address and M as method arguments. Bob notices Alice's second transaction before it was mined and quickly sends another transaction that calls `transferFrom()` method to transfer N Alice's tokens somewhere. If Bob's transaction is executed before Alice's transaction, then Bob will successfully transfer N Alice's tokens and will gain the ability to transfer another M tokens. Before Alice noticed that something went wrong, Bob calls `transferFrom()` method again, this time to transfer M Alice's tokens.

So, Alice's attempt to change Bob's allowance from N to M ( $N > 0$  and  $M > 0$ ) made it possible for Bob to transfer  $N+M$  of Alice's tokens, while Alice never wanted to allow so many of her tokens to be transferred by Bob.

BASE API: An Attack Vector on Approve/TransferFrom Methods

### Recommendation

We recommend only using `safeIncreaseAllowance()` and `safeDecreaseAllowance()` instead of the `approve()` method.

### Alleviation

#### [Growfi Team]

As the contract is immutable, using the `safeIncreaseAllowance` function is not an option. Future D.E.B.T. projects will consider using `safeIncreaseAllowance` and `safeDecreaseAllowance`.

## GRO-06 | Missing Zero Address Validation

Category	Severity	Location	Status
Logical Issue	● Informational	GROWFI.sol: 958	● Acknowledged

### Description

A sandwich attack might happen when an attacker observes a transaction swapping tokens or adding liquidity without setting restrictions on slippage or minimum output amount. The attacker can manipulate the exchange rate by frontrunning (before the transaction being attacked) a transaction to purchase one of the assets and make profits by backrunning (after the transaction being attacked) a transaction to sell the asset.

The `swapExactTokensForETHSupportingFeeOnTransferTokens()` function is called without setting restrictions on slippage or minimum output amount, so transactions triggering this function are vulnerable to sandwich attacks, especially when the input amount is large.

### Recommendation

We recommend setting reasonable minimum output amounts, instead of 0, based on token prices when calling the aforementioned functions.

### Alleviation

#### [Growfi Team]

Disabling swaps on the contract will avoid the sandwich attack. This mechanism can be used accordingly.

## GRO-07 | Incorrect Solidity Version

Category	Severity	Location	Status
Coding Style	● Informational	GROWFI.sol: 852, 853	● Acknowledged

### Description

The `_transfer()` function uses the magic number `20` as the fee for the council and dividend.

### Recommendation

We recommend replacing the magic number `20` with `COUNCIL_FEE` and `DIVIDEND_FEE` respectively.

### Alleviation

#### [Growfi Team]

As the contract is immutable, removing the magic number is not an option. Future D.E.B.T. projects will consider using more variables.

## GRO-08 | State Variables Should be Declared Constant

Category	Severity	Location	Status
Language Specific	● Informational	GROWFI.sol: 3	● Acknowledged

### Description

The contract has unlocked compiler version. An unlocked compiler version in the source code of the contract permits the user to compile it at or above a particular version. This, in turn, leads to differences in the generated bytecode between compilations due to differing compiler version numbers. This can lead to an ambiguity when debugging as compiler specific bugs may occur in the codebase that would be hard to identify over a span of multiple compiler versions rather than a specific one.

### Recommendation

We advise that the compiler version is instead locked at the lowest version possible that the contract can be compiled at. For example, for version `v0.8.12` the contract should contain the following line:

```
pragma solidity 0.8.12;
```

### Alleviation

#### [Growfi Team]

As the contract is immutable, specifying the solidity version is not an option. Other D.E.B.T. projects use the recommended version at time of creation.

## GRO-09 | Public Functions Should be Declared External

Category	Severity	Location	Status
Coding Style	● Informational	GROWFI.sol: 517, 524, 542, 549	● Acknowledged

### Description

There should always be events emitted in the sensitive functions that are controlled by centralization roles.

### Recommendation

It is recommended emitting events for the sensitive functions that are controlled by centralization roles.

### Alleviation

#### [Growfi Team]

As the contract is immutable, adding comprehensive emitters is not an option. Future D.E.B.T. projects will consider expanding the use of emitters.

## GRO-13 | No unchecked call responses found

Category	Severity	Location	Status
Inconsistency	● Informational	GROWFI.sol: 578~581	● Acknowledged

### Description

The highlighted comment does not reflect what the function below is doing.

### Recommendation

Please ensure the consistency between the code logic and comments.

### Alleviation

#### [Growfi Team]

As the contract is immutable, correcting the comments is not an option. Future D.E.B.T. projects will tune in on comments.

## GRO-14

### No vulnerable self-destruct functions found

### No assertion vulnerabilities found

Category	Severity	Location	Status
Coding Style	● Informational	GROWFI.sol: 5~239, 476~478, 873, 884, 912, 920, 940, 942, 943	● Acknowledged

### Description

The `GROW` contract uses `Pancakeswap` for swapping and adding liquidity to `Pancakeswap` pool, but naming it `Uniswap`. Function `swapTokensForEth(uint256 tokenAmount)` swaps `YUMMY` token for `BASE` instead of `ETH`.

### Recommendation

Change "Uniswap" and "ETH" to "Pancakeswap" and "BASE" in the contract respectively to match the operating environment and avoid confusion.

### Alleviation

#### [Growfi Team]

As the contract is immutable, correcting the variables is not an option. Future D.E.B.T. projects will consider similar variable changes.

## GRO-15 | No old solidity code found

Category	Severity	Location	Status
Coding Style	● Informational	GROWFI.sol: 441	● Acknowledged

### Description

One or more declarations do not conform to the [Solidity style guide](#) with regards to its naming convention.

Particularly:

- `camelCase` : Should be applied to function names, argument names, local and state variable names, modifiers
- `UPPER_CASE` : Should be applied to `constant` variables
- `CapWords` : Should be applied to contract names, struct names, event names and enums

### Recommendation

We recommend adjusting those variable and function names to properly conform to Solidity's naming convention.

### Alleviation

#### [Growfi Team]

As the contract is immutable, correcting the variables is not an option. Future D.E.B.T. projects will consider similar variable changes.

## GRO-16

No external delegated calls found  
No external call dependency found

Category	Severity	Location	Status
Volatile Code	● Informational	GROWFI.sol: 345~439	● Acknowledged

### Description

It is highly recommended NOT to include OpenZeppelin library code directly in source code, but import the original library to minimize risk because even slight changes to the library code may lead to critical/major vulnerabilities/bugs. For Solidity version 0.8.x, the latest OpenZeppelin version 4.7.3 should be used.

### Recommendation

We advise the client to remove OpenZeppelin library code from source code and import the latest OpenZeppelin version.

### Alleviation

#### [Growfi Team]

By including a copy of the library code directly, external changes cannot affect the inner workings of the contract.

## GRO-17 | No vulnerable authentication calls found

Category	Severity	Location	Status
Coding Style	● Informational	GROWFI.sol: 489	● Acknowledged

### Description

Literals with many digits are difficult to read and review.

### Recommendation

We advise the client to use the scientific notation to improve readability.

### Alleviation

#### [Growfi Team]

As the contract is immutable, changing the literal is not an option. Future D.E.B.T. projects will consider scientific notation.

# OPTIMIZATIONS | GROWFI

ID	Title	Category	Severity	Status
GRO-11	Variables That Could Be Declared As Immutable	Gas Optimization	Optimization	● Acknowledged
GRO-12	Function Should Be Declared External	Gas Optimization	Optimization	● Acknowledged
GRO-10	Variable Could Be Declared As <code>constant</code>	Gas Optimization	Optimization	● Acknowledged

## GRO-11 | VARIABLES THAT COULD BE DECLARED AS IMMUTABLE

Category	Severity	Location	Status
Gas Optimization	● Optimization	GROWFI.sol: 463~465	● Acknowledged

### Description

The linked variables assigned in the constructor can be declared as `immutable`. Immutable state variables can be assigned during contract creation but will remain constant throughout the lifetime of a deployed contract. A big advantage of immutable variables is that reading them is significantly cheaper than reading from regular state variables since they will not be stored in storage.

### Recommendation

We recommend declaring these variables as immutable. Please note that the `immutable` keyword only works in Solidity version `v0.6.5` and up.

## GRO-12 | FUNCTION SHOULD BE DECLARED EXTERNAL

Category	Severity	Location	Status
Gas Optimization	● Optimization	GROWFI.sol: 414, 422	● Acknowledged

### Description

The functions which are never called internally within the contract should have external visibility for gas optimization.

### Recommendation

We advise to change the visibility of the aforementioned functions to `external`.

## GRO-10 | VARIABLE COULD BE DECLARED AS `constant`

Category	Severity	Location	Status
Gas Optimization	<span>●</span> Optimization	GROWFI.sol: 486~489	<span>●</span> Acknowledged

### Description

Variables `_name`, `_symbol`, `_decimals` and `_initialSupply` could be declared as `constant` since these state variables are hardcoded in the constructor and never to be changed.

### Recommendation

We recommend declaring the mentioned variables as `constant`.

## APPENDIX | GROWFI

### Finding Categories

Categories	Description
Centralization / Privilege	Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.
Gas Optimization	Gas Optimization findings 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 detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.
Volatile Code	Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.
Language Specific	Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.
Coding Style	Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.
Inconsistency	Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different require statements on the input variables than a setter function.

### Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

## DISCLAIMER | CERTIK

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK's prior written consent in each instance.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is 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 assessment. This report does 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.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing 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.

The assessment services provided by CertiK is subject to dependencies and under continuing development. 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. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR 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, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'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. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR

UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

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

# CertiK | Securing the Web3 World

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.



