Code Security Assessment

GoldPesa.com

Feb 28th, 2022

Table of Contents

Summary

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

- 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.

Overview

Project Summary

Project Name	GoldPesa.com
Platform	Ethereum
Language	Solidity
Codebase	https://github.com/GoldPesa/goldpesa-core
Commit	d76b186bfeadbbdf5992f0831acb5742a2e9e835

Audit Summary

Delivery Date	Feb 28, 2022
Audit Methodology	Static Analysis, Manual Review

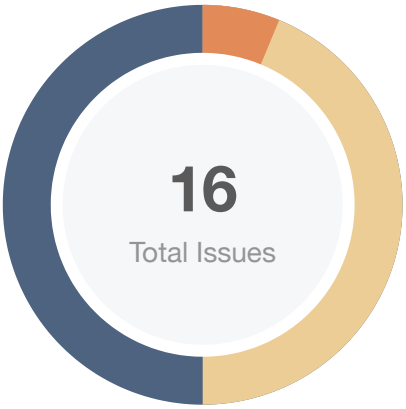
Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Mitigated	Resolved
Critical	0	0	0	0	0	0	0
Major	1	0	0	0	1	0	0
Medium	0	0	0	0	0	0	0
Minor	7	0	0	5	0	0	2
Informational	8	0	0	7	0	0	1
Discussion	0	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
GPO	GPO.sol	c07768a4496c1ed83312068fcee6f8ed084843fe60cc70ecb865486710a481ec

Findings



Critical	**0** (0.00%)
Major	**1** (6.25%)
Medium	**0** (0.00%)
Minor	**7** (43.75%)
Informational	**8** (50.00%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
GPO-01	Unlocked Compiler Version	Language Specific	Informational	Acknowledged
GPO-02	Function Visibility Optimization	Gas Optimization	Informational	Acknowledged
GPO-03	Missing Emit Events	Coding Style	Informational	Acknowledged
GPO-04	Variables That Could Be Declared As `constant`	Gas Optimization	Informational	Acknowledged
GPO-05	Centralization Risk in GPO.sol	**Centralization / Privilege**	**Major**	Partially Resolved
GPO-06	External Call Inside a Loop	Control Flow	Minor	Resolved
GPO-07	Missing Zero Address Validation	Control Flow	Minor	Acknowledged
GPO-08	Unimplemented Functionality	Inconsistency	Informational	Acknowledged
GPO-09	Unchecked Value of ERC-20 `transfer()` Call	Volatile Code	Minor	Acknowledged
GPO-10	Inconsistent Comments and Code	Logical Issue	Informational	Resolved
GPO-11	Missing Error Messages	Coding Style	Informational	Acknowledged
GPO-12	Undocumented Functionality	Inconsistency, Control Flow	Minor	Resolved
GPO-13	Lack Of Transparency On Fees Distribution	Data Flow	Informational	Acknowledged

ID	Title	Category	Severity	Status
GPO-14	Third Party Dependencies	Control Flow	Minor	Acknowledged
GPO-15	Lack of Amount Check Before Token Swap	Gas Optimization	Minor	Acknowledged
GPO-16	Potential Benign Reentrancy	Control Flow	Minor	Acknowledged

GPO-01 | Unlocked Compiler Version

Category	Severity	Location	Status
Language Specific	Informational	GPO.sol: 2	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 `v0.8.0` the contract should contain the following line:

```
pragma solidity 0.8.0;
```

Alleviation

`[GoldPesa.com]`: Noted and Understood.

GPO-02 | Function Visibility Optimization

Category	Severity	Location	Status
Gas Optimization	Informational	GPO.sol: 88, 235, 231, 220, 136, 132, 127, 123, 96, 92	Acknowledged

Description

`public` functions that are never called by the contract could be declared as `external`. `external` functions are more efficient than `public` functions and can save gas.

Recommendation

Consider using the external attribute for public functions that are never called within the contract.

Alleviation

`[GoldPesa.com]`: Noted and Understood.

GPO-03 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	Informational	GPO.sol: 88, 136, 132, 127, 123, 96, 92	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

`[GoldPesa.com]`: Noted and Understood.

GPO-04 | Variables That Could Be Declared As `constant`

Category	Severity	Location	Status
Gas Optimization	Informational	GPO.sol: 34, 36	Acknowledged

Description

The linked variables could be declared as `constant` since these state variables are never modified.

Recommendation

We recommend to declare these variables as `constant`.

Alleviation

`[GoldPesa.com]`: Noted and Understood.

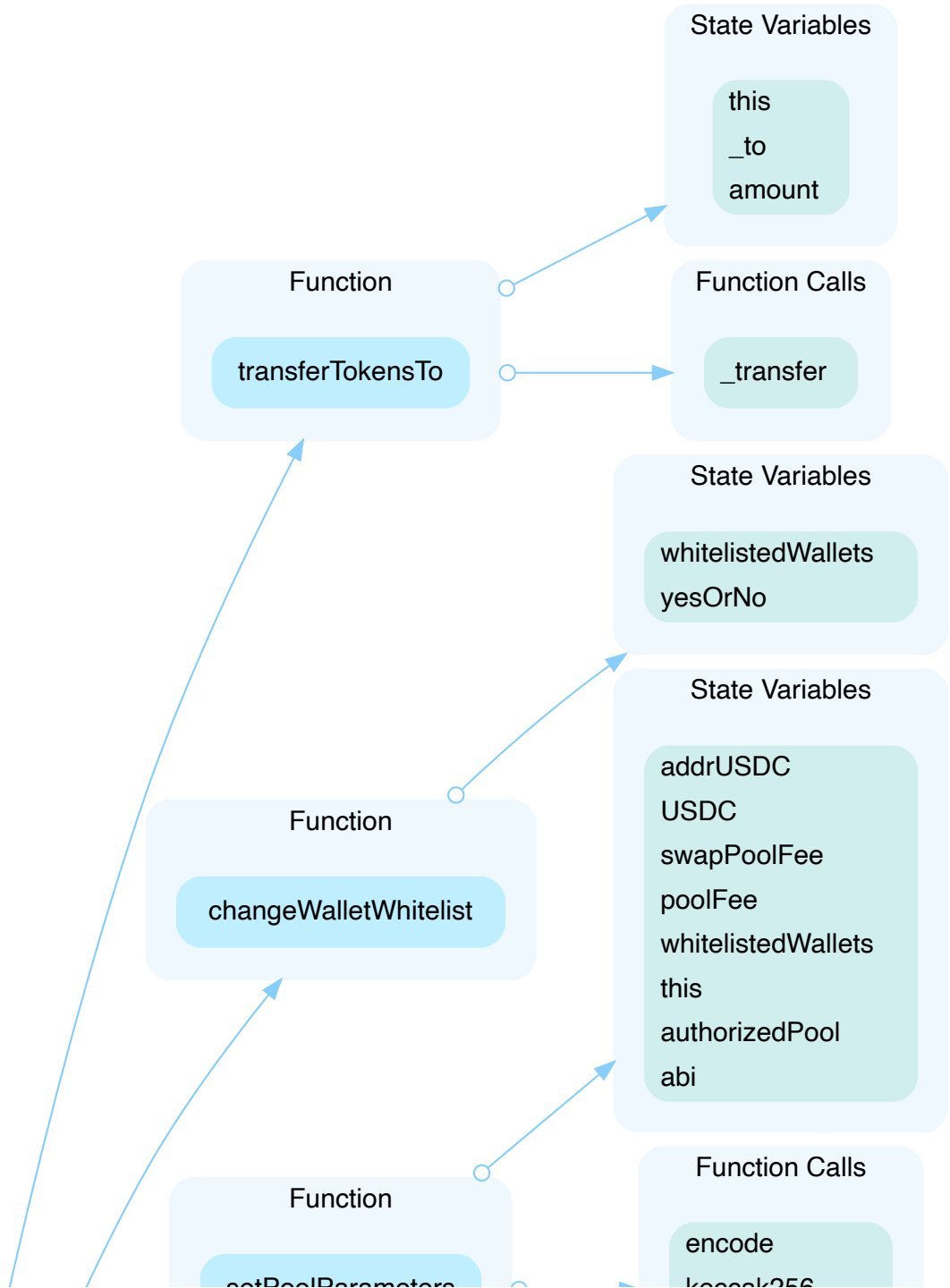
GPO-05 | Centralization Risk In GPO.sol

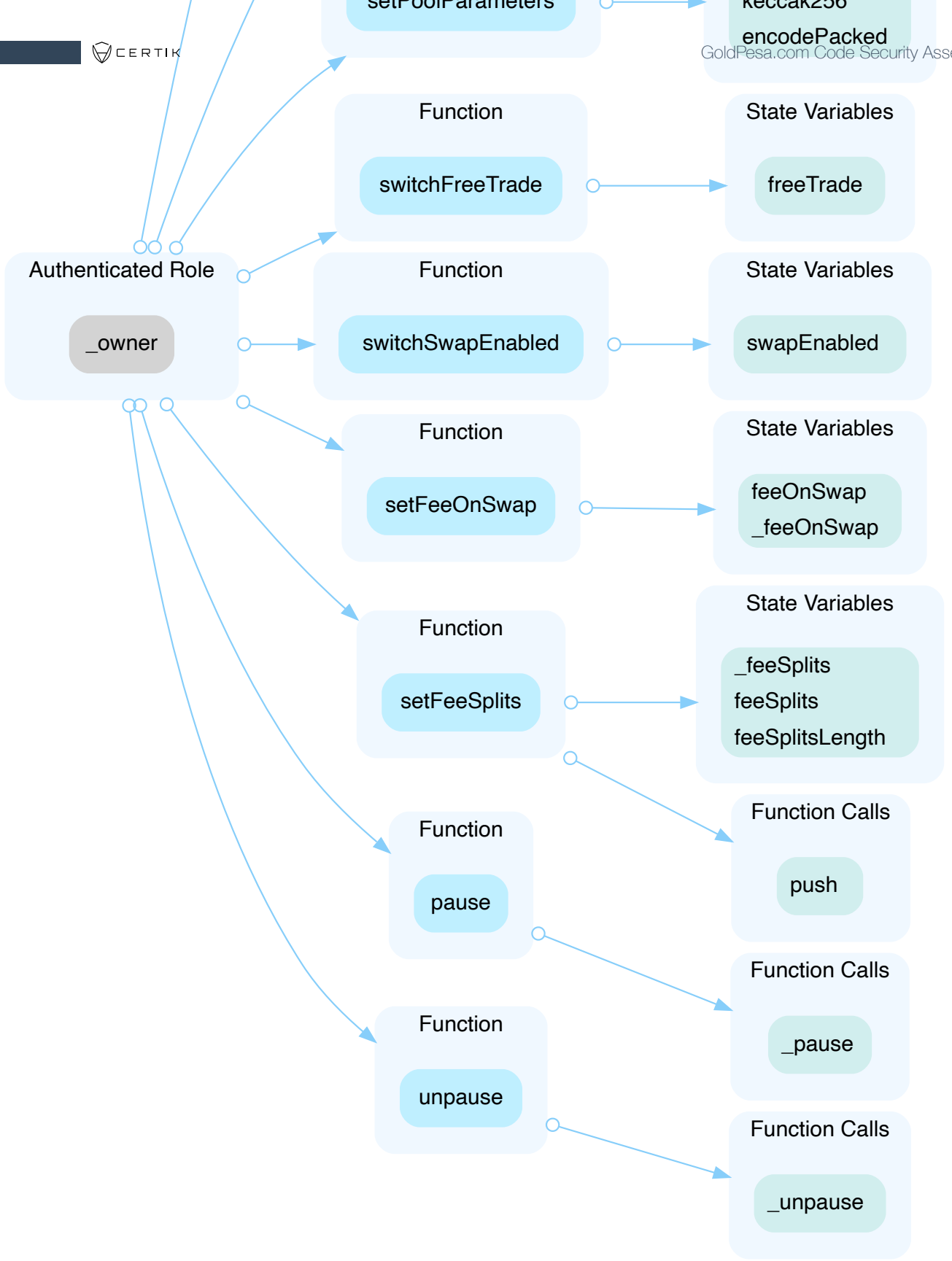
Category	Severity	Location	Status
Centralization / Privilege	Major	GPO.sol: 88, 235, 231, 136, 132, 127, 123, 96, 92	Partially Resolved

Description

In the contract, `GPO`, the role, `_owner`, has the authority over the functions shown in the diagram below.

Any compromise to the privileged account which has access to `_owner` may allow the hacker to take advantage of this and can pause the contract, switch on "Free Trade" or change pool parameters among other functionalities.





Recommendation

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 to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the different level in term of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

`[GoldPesa.com]`: The _Owner Role has been assigned to a multisig Gnosis Safe address 0x33E0EC5226D7175d024a7D6B066d0beE3F8aC9C0 2 of 5 signatures are required to execute transactions from this Gnosis safe and therefore this risk has already been mitigated.

GPO-06 | External Call Inside A Loop

Category	Severity	Location	Status
Control Flow	Minor	GPO.sol: 151	Resolved

Description

Function `distributeFee()` has external calls inside a loop. In case any of the calls fail, it can cause the entire loop to revert. It is advice to change the reward distribution strategy.

Recommendation

Favor pull over push strategy for external calls.

Alleviation

`[GoldPesa.com]`: Although not checking if the external calls revert is not good practice, this function can never fail. The aforementioned external calls are standard ERC20 token transfers from a trusted ERC20 token (in our case, USDC, but we made it so we can change it to another address in case we decide to go another way). Those tokens do not have any transfer requirements, other than the basic ones (checking the balance). We figured that even if one day, USDC or any other token we decide to use, blacklists the GPO smart contract, the swap on Uniswap would not even be working and our custom swap would revert before/after the distributeFee function is executed. Finally, these external calls are all made to the same smart contract address, so if one fails, all of them fail technically. Unless, obviously, the recipient wallets of the fee distribution are themselves blacklisted by the token (which will never happen obviously, but if we think hypothetically –), in which case we can change the distribution wallet addresses whenever that happens.

Also, just as clarification, the fee distribution feature is for us to take a certain percentage on any swaps and reinvest the money on marketing and our quant firm, so you can be sure that we will solve any problems that could arise if any.

All in all, this is why we would consider this issue to be minor, instead of medium.

GPO-07 | Missing Zero Address Validation

Category	Severity	Location	Status
Control Flow	Minor	GPO.sol: 97	Acknowledged

Description

The function `setPoolParameters()` assigns the `addrUSDC` variable to the value obtained from the parameter without checking if it is the Zero address.

Recommendation

Add a `require` statement verifying the USDC parameter before assigning.

Alleviation

`[GoldPesa.com]`: Noted and Understood.

GPO-08 | Unimplemented Functionality

Category	Severity	Location	Status
Inconsistency	Informational	GPO.sol: 143	Acknowledged

Description

The comment inside function `setFeeSplits()` refers to a temporary functionality of allowing 0 fee splits but it is the default behavior.

Recommendation

We advice to either remove the comment or prefer for a conditional flow that would trigger the temporary functionality.

Alleviation

`[GoldPesa.com]`: Noted and Understood.

GPO-09 | Unchecked Value Of ERC-20 `transfer()` Call

Category	Severity	Location	Status
Volatile Code	Minor	GPO.sol: 161, 156	Acknowledged

Description

`transfer()`/`transferFrom()` invocations do not check the return value of the function call which should yield a `true` result in case of proper ERC-20 implementation.

Recommendation

As many tokens do not follow the ERC-20 standard faithfully, they may not return a `bool` variable in this function's execution meaning that simply expecting it can cause incompatibility with these types of tokens. Instead, we advise that OpenZeppelin's `SafeERC20.sol` implementation is utilized for interacting with the `transfer()` and `transferFrom()` functions of ERC-20 tokens. The OZ implementation optionally checks for a return value rendering compatible with all ERC-20 token implementations.

Alleviation

`[GoldPesa.com]`: Same response as GPO-06: the practice could be considered bad, but we can be certain that this will never be an actual issue.

GPO-10 | Inconsistent Comments And Code

Category	Severity	Location	Status
Logical Issue	Informational	GPO.sol: 172	Resolved

Description

On the `_beforeTokenTransfer()` function, the `require` statement doesn't follow the comment above. If the user is on the whitelist, the `or` condition will suffice and it will ignore the `to != address(0x0)` condition, meaning that a user on the whitelist can burn tokens. Although this wouldn't take place as the `_transfer()` function of the ERC20 contract checks whether the `to` or `from` addresses are not the zero address before calling `_beforeTokenTransfer()`.

Recommendation

Change the `or` boolean expression to an `and` or make a separate `require` statement that checks if the recipient address is the zero address.

Alleviation

`[GoldPesa.com]`: The comment on line 171 is indeed misleading. We are allowing whitelisted wallets to burn the token, so none of the token holders cannot burn their own tokens but we (the whitelisted wallets) can, as we decide which wallets are whitelisted. In short, a whitelisted wallet has the right to hold more than 100k GPO, trade and manage liquidity freely on Uniswap and burn its tokens.

Now, regarding whether the burn function will work or not, the internal _burn function written in the ERC20.sol (from OpenZeppelin), called from the public burn function in ERC20Burnable, extended by ERC20PresetFixedSupply (which is GPO's base class), does call the _beforeTransferToken function, which makes our require statement valid and not redundant. Another way we could have done it is to just override the _burn function, or the burn function itself, but we have decided to keep everything in the beforeTokenTransfer function to keep a single place with all the "rules" of our token transfer (including minting and burning).

GPO-11 | Missing Error Messages

Category	Severity	Location	Status
Coding Style	Informational	GPO.sol: 224	Acknowledged

Description

The **require** can be used to check for conditions and throw an exception if the condition is not met. It is better to provide a string message containing details about the error that will be passed back to the caller.

Recommendation

We advice to add an error message displaying that the amount of tokens to be locked is bigger than the user's balance.

Alleviation

`[GoldPesa.com]`: Noted and Understood.

GPO-12 | Undocumented Functionality

Category	Severity	Location	Status
Inconsistency, Control Flow	Minor	GPO.sol: 142	Resolved

Description

Function `setFeeSplits()` recieves an array of FeeSplits struct and adds their `fee` field to the variable `grandTotal`, which is then forced to equal the value `100` on the `require` statement. Given that the data type of the `fee` field is `uint16`, this condition will make the max amount of participants who can get rewards of the fees to be 100. There are no comments of such functionality.

Recommendation

We advice to document this functionality if it was the intended behavior or to change the `fee` data type from `uint16` to `float` or `double` to allow more participation of the fees distribution. If the latter recommendation is to be follow, please ensure to check every usage of the variable on every math operation.

Alleviation

`[GoldPesa.com]`: It's important to note that our tokenomics does not have any fee redistribution to token holders. The setFeeSplits function is only set to distribute fees to the dev teams wallets, liquidity pool or other smart contract we as a company deploy. There will never be a scenario based on our tokenomics for more than 100 addresses.

GPO-13 | Lack Of Transparency On Fees Distribution

Category	Severity	Location	Status
Data Flow	Informational	GPO.sol: 160	Acknowledged

Description

On the `distributeFee()` function, every remnant the distribution is sent to the first address of the `feeSplits` array but given that the array could have been given using the `setFeeSplits()` function, the first address of the array could be anything, including the Zero Address. This is not very helpful for transparency as users would not know if some address may be preferred over others and thus, placing it first on the array.

Recommendation

In case of a clear functionality, for example sending the remnants to the Zero address, prefer to use `address(0)` instead of `feeSplits[0]`.

Alleviation

`[GoldPesa.com]`: Adding to point GPO-12 point above, the distributeFee function is only called anytime 1 of the 4 swap functions are called. When this swap takes place a fee is split based on the feeSplits array which will never contain any addresses belonging to our token holders as we do not have a fee redistribution in our tokenomics. Therefore, regardless of the first or last address in the feeSplits array or even if its address 0 it would be of no concern to our token holders. The feeSplits array is designed to have under 10 addresses which are either EOA, Liquidity pools or other smart contracts which are all controlled by the dev team in some way.

We are distributing the fees in USDC so they would never be sent to the zero address!

GPO-14 | Third Party Dependencies

Category	Severity	Location	Status
Control Flow	Minor	GPO.sol	Acknowledged

Description

The contract is serving as the underlying entity to interact with third party Uniswap protocols. The scope of the audit treats 3rd party entities as black boxes and assume their functional correctness. However, in the real world, 3rd parties can be compromised and this may lead to lost or stolen assets. In addition, upgrades of 3rd parties can possibly create severe impacts, such as increasing fees of 3rd parties, migrating to new LP pools, etc.

Recommendation

We understand that the business logic of GPO requires interaction with Uniswap. We encourage the team to constantly monitor the statuses of 3rd parties to mitigate the side effects when unexpected activities are observed.

Alleviation

`[GoldPesa.com]`: Noted and Understood.

GPO-15 | Lack Of Amount Check Before Token Swap

Category	Severity	Location	Status
Gas Optimization	Minor	GPO.sol: 343, 309, 274, 241	Acknowledged

Description

On functions `swapToExactOutput()`, `swapToExactInput()`, `swapFromExactOutput()` and `swapFromExactInput()` there is no validation of the variables containing the amounts of tokens to be swaped. If a user makes a swap with all the amounts to be swapped equal to zero, this would still be valid and would not only cause to waste gas on the swap but also on every transaction made by the `distributeFee()` function, which could lead to potential big loses.

Recommendation

Consider adding a validation for the amount to be greater than zero in all swap functions and on the `distributeFee()` function.

Alleviation

`[GoldPesa.com]`: This is noted and understood but I believe should be a minor issue and not medium. We call these 4 functions from the app.goldpesa.com DEX where we validate the amount to be greater than zero before calling any of the swap functions.

GPO-16 | Potential Benign Reentrancy

Category	Severity	Location	Status
Control Flow	Minor	GPO.sol: 343, 309, 274, 241	Acknowledged

Description

Functions `swapToExactOutput()`, `swapToExactInput()`, `swapFromExactOutput()` and `swapFromExactInput()` make an external call to the `addrUSDC` variable. Although the address should be pointing to the USDC Smart Contract address which should be trusted, it is assign via the `setPoolParameters()`.

There is no mechanisms on the aforementioned functions to avoid the external call make another call to the same functions. In this case, this would be the same as interacting with the functions more than once, which doesn't have a great impact for the contract.

Recommendation

We recommend using the check-effects-interactions pattern.

Alleviation

`[GoldPesa.com]`: Noted and Understood.

Appendix

Finding Categories

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.

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.

Data Flow

Data Flow findings describe faults in the way data is handled at rest and in memory, such as the result of a struct assignment operation affecting an in-memory struct rather than an in-storage one.

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

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.

About

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.