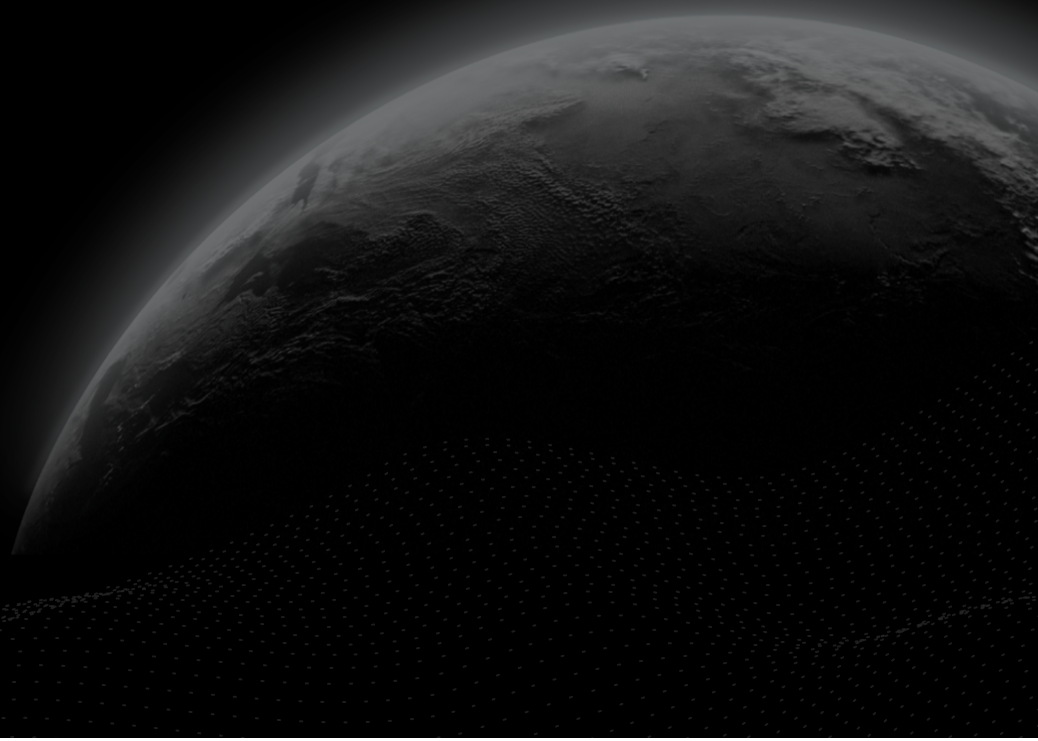
CERTIK

Security Assessment

LFGswap

CertiK Verified on Oct 21st, 2022



CertiK Verified on Oct 21st, 2022

LFGswap

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

Executive Summary

TYPES	ECOSYSTEM	METHODS
DEX	Ethereum (ETH)	Manual Review, Static Analysis
LANGUAGE	**TIMELINE**	**KEY COMPONENTS**
Solidity	Delivered on 10/21/2022	N/A

CODEBASE
https://github.com/LfgSwap/lfg-protocol
...View All

COMMITS
0f58905d83e8a6403c4965fe8057c5f1a2539ae2
...View All

Vulnerability Summary

15	0	2	0	13	0	0
Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined	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.
3 Major	2 Mitigated, 1 Acknowledged	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.

4	Minor	4 Acknowledged	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.
8	Informational	8 Acknowledged	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 | LFGSWAP

CODEBASE | LFGSWAP

Repository

https://github.com/LfgSwap/lfg-protocol

Commit

0f58905d83e8a6403c4965fe8057c5f1a2539ae2

AUDIT SCOPE | LFGSWAP

16 files audited ● 8 files with Acknowledged findings ● 8 files without findings

ID	File	SHA256 Checksum
LTL	contracts/LfgToken.sol	503a33cd667a3d8004fd24617e900b1d126e97bbc4725099f90d3f3b689490d0
LSF	contracts/core/LfgSwapFactory.sol	118a47015d773227d73bce1598429e10c3465fbf43e1387cd7c3d125ea6595b0
LSP	contracts/core/LfgSwapPair.sol	110bf023f6c1f7faed43929c7167f46e3d0392b9e24b9ed80f2cb0c9a3c03c21
LSR	contracts/core/LfgSwapRouter.sol	bc8b25d7ae4f265cc9b9b614a5c9b54fe72b5090fedf199fe64fc29bb02e4b1e
ISM	contracts/interface/ISwapMining.sol	16ca36a9273fce3591a017a877b44c5b9f5b92e4d49df6ef0710003f4eb9ac98
FPL	contracts/libraries/FixedPoint.sol	d47c279bdd9024bf0c7c59755fab09d3d6a8fad71e9d1154e30dc045e5643099
LSL	contracts/libraries/LfgSwapLibrary.sol	63b973eb8825e133072b626b4e39a9e4515a672ea6545eee3b9fae38e15f6e74
SML	contracts/libraries/SafaMath.sol	be7b55582bda6261ac326aeb5ab661672b45a498405610a2a2aaac370488a69b
IER	contracts/interface/IERC20LfgSwap.sol	f1673bb2169ebe76b7d71b1ac0895a1f3c48343097a2b75cd8950dbba2bcb413
ILS	contracts/interface/ILfgSwapFactory.sol	3f185777e8075233cc97a3db1b706a74b8d73ebef433eb32a321d94fa95a5ce3
ILP	contracts/interface/ILfgSwapPair.sol	5df3cebe46e7f6e5e7eb183e47a187e621c0378ef95543aaa202512502eb21e7
ILR	contracts/interface/ILfgSwapRouter.sol	7cd914e53d0b389ec7050c49da432a452267843a66e5cce7294b4c1703c19f52
ILT	contracts/interface/ILfgToken.sol	768881c2a7fe3b2ef5ceaafa9d764b3bfe0071576f835ba10579bfc90ea7da28

ID	File	SHA256 Checksum
IWE	contracts/interface/IWETH.sol	de16b4553228c2e904c505c54be313d0f64f4f99bba36b34adde2c2b1554e9ec
THL	contracts/libraries/TransferHelper.sol	4df6715ebc2d1b3f0aed22ff7376c13c9fa5fa1c490b0c531bf10949cace6f56
UQL	contracts/libraries/UQ112x112.sol	f7e1e2d0275a103f2332b12bfca703e65e2881b23c823658ab8878cbb7615a92

APPROACH & METHODS | LFGSWAP

This report has been prepared for LFGswap to discover issues and vulnerabilities in the source code of the LFGswap 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 | LFGSWAP

15	0	3	0	4	8
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for LFGswap. Through this audit, we have uncovered 15 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
LSF-01	**Centralization Risks In LfgSwapFactory.Sol**	**Centralization / Privilege**	**Major**	**Acknowledged**
LSF-02	Missing Zero Address Validation	Volatile Code	Minor	Acknowledged
LSP-01	Missing Input Validation	Volatile Code	Minor	Acknowledged
LSP-02	Divide By Zero	Logical Issue	Minor	Acknowledged
LTL-01	**Initial Token Distribution**	**Centralization / Privilege**	**Major**	**Mitigated**
LTL-02	**Centralization Risks In LfgToken.Sol**	**Centralization / Privilege**	**Major**	**Mitigated**
LTL-03	Potential Integer Underflow	Mathematical Operations	Minor	Acknowledged
FPL-01	Unused Library `FixedPoint`	Coding Style	Informational	Acknowledged
LSL-01	Commented Out Code	Coding Style	Informational	Acknowledged
LSP-03	Redundant Check	Logical Issue	Informational	Acknowledged

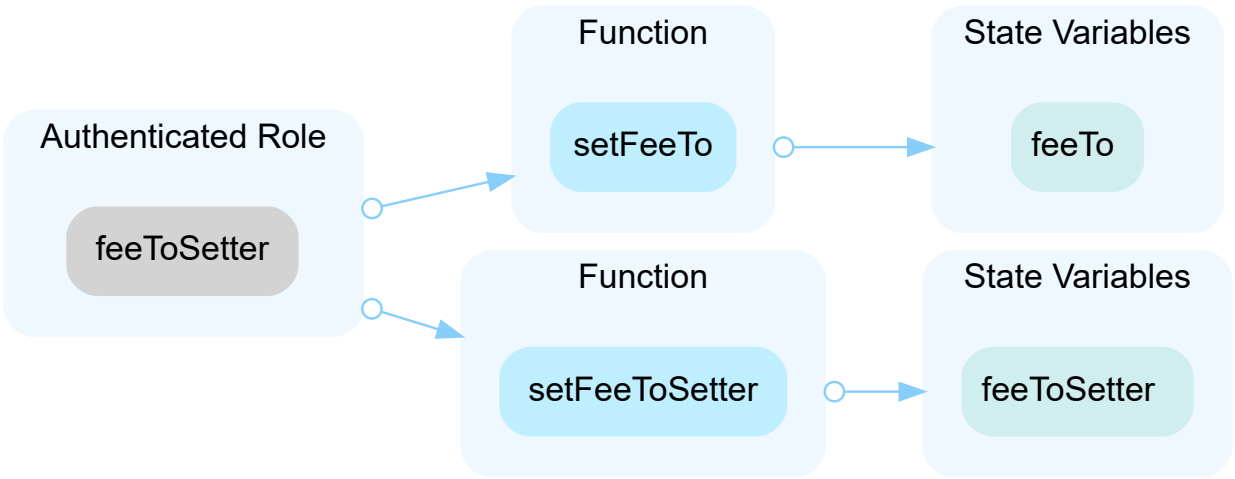
ID	Title	Category	Severity	Status
LSP-04	Typo In File Name `SafaMath.sol`	Coding Style	Informational	Acknowledged
LSP-05	Incorrect Comment	Coding Style, Inconsistency	Informational	Acknowledged
LSR-01	Unused Import File	Coding Style	Informational	Acknowledged
LSU-02	Redundant Interfaces `IMigrator` And `ISwapMining`	Coding Style	Informational	Acknowledged
LSU-03	Typos In `require()` Check Error Messages	Coding Style	Informational	Acknowledged

LSF-01 | CENTRALIZATION RISKS IN LFGSWAPFACTORY.SOL

Category	Severity	Location	Status
Centralization / Privilege	**Major**	**contracts/core/LfgSwapFactory.sol: 69, 75**	**Acknowledged**

Description

In the contract `LfgSwapFactory` the role `feeToSetter` has authority over the functions shown in the diagram below. Any compromise to the `feeToSetter` account may allow the hacker to take advantage of this authority and set `feeTo`.



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 (⅔, ⅗) 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.

Alleviation

`[LFGswap]`: We had called `setFeeToSetter('0x67deFBFa85289E9a767D29ef9682a9B80Bd11A38')`. `0x67de` is Multi sign (⅗) address created by wafebox which is a Mult Sign supporting ETHW chain.

`[CertiK]`: The Issue has been acknowledged. The client used a Multi sign wallet but didn't follow all the recommendations.

LSF-02 | MISSING ZERO ADDRESS VALIDATION

Category	Severity	Location	Status
Volatile Code	Minor	contracts/core/LfgSwapFactory.sol: 17, 71, 77	Acknowledged

Description

Addresses should be checked before assignment or external call to make sure they are not zero addresses.

Recommendation

We advise adding a zero-check for the passed-in address value to prevent unexpected errors.

LSP-01 | MISSING INPUT VALIDATION

Category	Severity	Location	Status
Volatile Code	Minor	contracts/core/LfgSwapPair.sol: 58, 64, 70~71	Acknowledged

Description

In function `_burn()`, `_approve()` and `_transfer()`, addresses should be checked before assigning to make sure they are not zero addresses.

Recommendation

We recommended adding `require()` statements to make sure the input parameters are not `address(0)`. Here is the list of input parameters to be checked:

- `_burn()`: `_from`
- `_approve()`: `owner` and `spender`
- `_transfer()`: `from` and `to`

LSP-02 | DIVIDE BY ZERO

Category	Severity	Location	Status
Logical Issue	Minor	contracts/core/LfgSwapPair.sol: 243~244	Acknowledged

Description

The call to `burn()` function will fail if the value of `_totalSupply` is 0.

Recommendation

We advise the client to add the following validation in the function `burn()`

```
1  require(totalSupply != 0, "The value of totalSupply must not be 0");
```

LTL-01 | INITIAL TOKEN DISTRIBUTION

Category	Severity	Location	Status
Centralization / Privilege	**Major**	**contracts/LfgToken.sol: 15**	**Mitigated**

Description

Tokens are sent to `msg.sender` when deploying the contract. This could be a centralization risk as the `msg.sender` can distribute 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

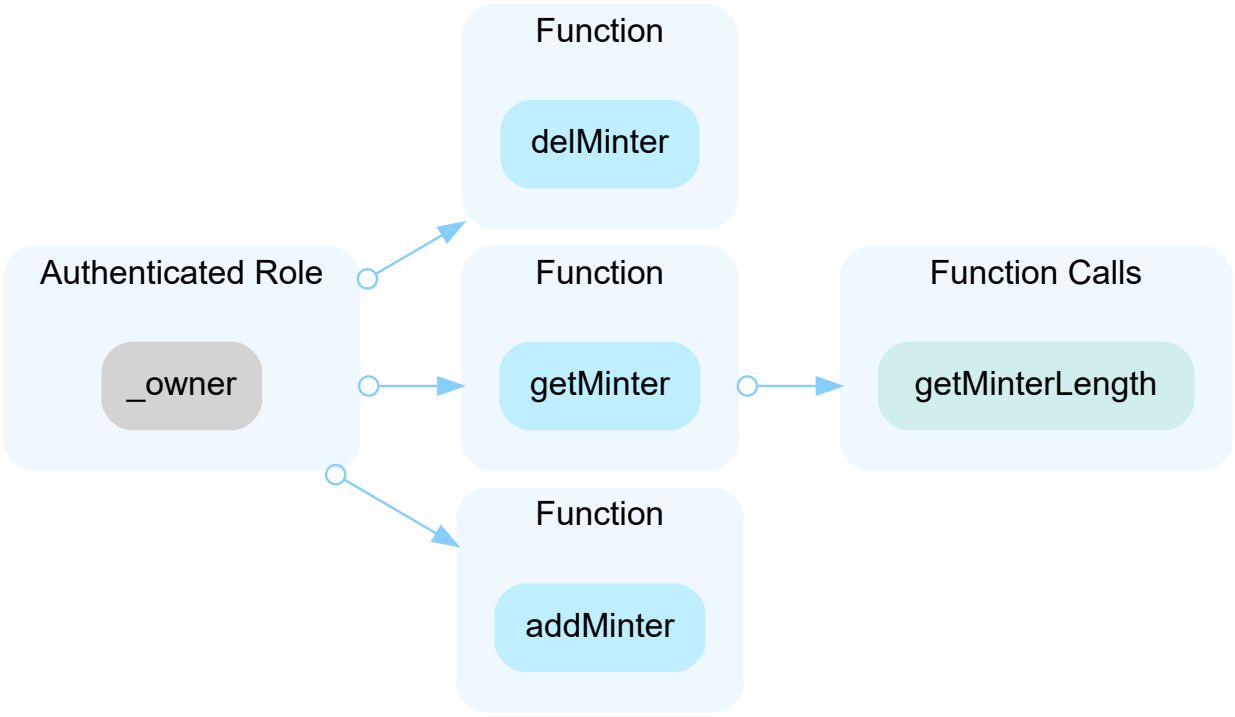
`[LFGswap]` : The preMint code scope has been disabled in the LfgToken public environment and had an open source. And we had put a commit at github to sync the code between browser and github.

LTL-02 | CENTRALIZATION RISKS IN LFGTOKEN.SOL

Category	Severity	Location	Status
Centralization / Privilege	**Major**	**contracts/LfgToken.sol: 19, 24, 29, 42**	**Mitigated**

Description

In the contract `LfgToken` 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 add or delete `Minter` role.



In the contract `LfgToken`, the role `_minters` has authority over the `mint()` function. Any compromise to the `_minters` account may allow a hacker to take advantage of this authority and mint new tokens.

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 (⅔, ⅗) 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.

Alleviation

`[LFGswap]`: Now the `delMinter()` at Tx and `renounceOwnership()` at Tx:0x817e flow the `Permanent Recommendation`. The effects of the operation also could be checked by call `owner()` and `getMinterLength()` of LfgToken.

LTL-03 | POTENTIAL INTEGER UNDERFLOW

Category	Severity	Location	Status
Mathematical Operations	Minor	contracts/LfgToken.sol: 44	Acknowledged

Description

In the function `LfgToken.getMinter()`, the input `_index` is checked to ensure the index is valid. However, it is possible that `getMinterLength() == 0` which might lead to integer underflow in the calculation:

```
43          require(_index <= getMinterLength() - 1, "JfToken: index out of bounds");
```

Recommendation

We advise the client to use the SafeMath library for all of the mathematical operations.

FPL-01 | UNUSED LIBRARY `FixedPoint`

Category	Severity	Location	Status
Coding Style	Informational	contracts/libraries/FixedPoint.sol: 1~60	Acknowledged

Description

The library `FixedPoint` is never used within the project and thus can be removed.

Recommendation

We advise removing the library `FixedPoint`.

LSL-01 | COMMENTED OUT CODE

Category	Severity	Location	Status
Coding Style	Informational	contracts/libraries/LfgSwapLibrary.sol: 23~28	Acknowledged

Description

Commented out code is redundant.

Recommendation

We recommend removing the commented out code.

LSP-03 | REDUNDANT CHECK

Category	Severity	Location	Status
Logical Issue	Informational	contracts/core/LfgSwapPair.sol: 259	Acknowledged

Description

In the `swap` function, the `require(amount0Out > 0 || amount1Out > 0, 'LfgSwap: INSUFFICIENT_OUTPUT_AMOUNT');` will always be satisfied due to the condition `if( amount0Out > 0 || amount1Out > 0 )`.

Recommendation

We recommend reviewing the logic of the function to avoid duplicate logic.

LSP-04 | TYPO IN FILE NAME `SafaMath.sol`

Category	Severity	Location	Status
Coding Style	Informational	contracts/core/LfgSwapPair.sol: 3	Acknowledged

Description

There is a typo in the file name `SafaMath.sol`.

Recommendation

We recommend changing the file name to `SafeMath.sol`.

LSP-05 | INCORRECT COMMENT

Category	Severity	Location	Status
Coding Style, Inconsistency	Informational	contracts/core/LfgSwapPair.sol: 187	Acknowledged

Description

Based on the coefficient `mul(2)` and based on the formula at 2.4 of https://uniswap.org/whitepaper.pdf, the correct fee is equivalent to 1/3th.

Recommendation

We recommend correcting the comment based on the project code.

LSR-01 | UNUSED IMPORT FILE

Category	Severity	Location	Status
Coding Style	Informational	contracts/core/LfgSwapRouter.sol: 3, 14	Acknowledged

Description

The following import file is not used in the audit files:

```
3  import "@openzeppelin/contracts/access/Ownable.sol";
```

Recommendation

Consider commenting this import out or removing it.

LSU-02 | REDUNDANT INTERFACES `IMigrator` AND `ISwapMining`

Category	Severity	Location	Status
Coding Style	● Informational	contracts/core/LfgSwapPair.sol: 8~11; contracts/interface/ISwapMining.sol: 1~7	● Acknowledged

Description

The `IMigrator` and `ISwapMining` interfaces are not used by other code.

Recommendation

We recommend removing the redundant declarations to better prepare the code for production environments.

LSU-03 | TYPOS IN `require()` CHECK ERROR MESSAGES

Category	Severity	Location	Status
Coding Style	Informational	contracts/LfgToken.sol: 25, 30, 43; contracts/core/LfgSwapPair.sol: 96, 105; contracts/libraries/LfgSwapLibrary.sol: 13, 15, 40, 41, 47, 48, 57, 58, 66, 77	Acknowledged

Description

There is a typos in `require` messages:

- `UniswapV2Library` expected `LfgSwapLibrary`;
- `UniswapV2` expected `LfgSwap`.
- `JfToken` expected `LfgToken`

Recommendation

We recommend correcting all typos.

OPTIMIZATIONS | LFGSWAP

ID	Title	Category	Severity	Status
LSB-01	Variables That Could Be Declared As Immutable	Gas Optimization	Optimization	Acknowledged
LSU-01	Function Should Be Declared External	Gas Optimization	Optimization	Acknowledged
SML-01	User-Defined Getters	Gas Optimization	Optimization	Acknowledged

LSB-01 | VARIABLES THAT COULD BE DECLARED AS IMMUTABLE

Category	Severity	Location	Status
Gas Optimization	Optimization	contracts/core/LfgSwapPair.sol: 117; contracts/core/LfgSwapRouter.sol: 17, 18	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.

LSU-01 | FUNCTION SHOULD BE DECLARED EXTERNAL

Category	Severity	Location	Status
Gas Optimization	Optimization	contracts/LfgToken.sol: 19, 24, 29, 42; contracts/core/LfgSwapFactory.sol: 37, 64; contracts/core/LfgSwapRouter.sol: 413, 417, 427, 437, 447; contracts/libraries/SafaMath.sol: 7, 11	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`.

SML-01 | USER-DEFINED GETTERS

Category	Severity	Location	Status
Gas Optimization	Optimization	contracts/libraries/SafaMath.sol: 7~9, 11~13	Acknowledged

Description

The linked functions are equivalent to the compiler-generated getter functions for the respective variables.

Recommendation

We advise that the linked variables are instead declared as `public` as compiler-generated getter functions are less prone to error and much more maintainable than manually written ones.

FORMAL VERIFICATION | LFGSWAP

Formal guarantees about the behavior of smart contracts can be obtained by reasoning about properties relating to the entire contract (e.g. contract invariants) or to specific functions of the contract. Once such properties are proven to be valid, they guarantee that the contract behaves as specified by the property. As part of this audit, we applied automated formal verification (symbolic model checking) to prove that well-known functions in the smart contracts adhere to their expected behavior.

Considered Functions And Scope

Verification of ERC-20 compliance

We verified properties of the public interface of those token contracts that implement the ERC-20 interface. This covers

- Functions `transfer` and `transferFrom` that are widely used for token transfers,
- functions `approve` and `allowance` that enable the owner of an account to delegate a certain subset of her tokens to another account (i.e. to grant an allowance), and
- the functions `balanceOf` and `totalSupply`, which are verified to correctly reflect the internal state of the contract.

The properties that were considered within the scope of this audit are as follows:

Property Name	Title
erc20-transfer-revert-zero	Function `transfer` Prevents Transfers to the Zero Address
erc20-transfer-succeed-normal	Function `transfer` Succeeds on Admissible Non-self Transfers
erc20-transfer-succeed-self	Function `transfer` Succeeds on Admissible Self Transfers
erc20-transfer-correct-amount	Function `transfer` Transfers the Correct Amount in Non-self Transfers
erc20-transfer-correct-amount-self	Function `transfer` Transfers the Correct Amount in Self Transfers
erc20-transfer-change-state	Function `transfer` Has No Unexpected State Changes
erc20-transfer-exceed-balance	Function `transfer` Fails if Requested Amount Exceeds Available Balance
erc20-transfer-recipient-overflow	Function `transfer` Prevents Overflows in the Recipient's Balance

Property Name	Title
erc20-transfer-never-return-false	Function `transfer` Never Returns `false`
erc20-transfer-false	If Function `transfer` Returns `false`, the Contract State Has Not Been Changed
erc20-transferfrom-revert-from-zero	Function `transferFrom` Fails for Transfers From the Zero Address
erc20-transferfrom-revert-to-zero	Function `transferFrom` Fails for Transfers To the Zero Address
erc20-transferfrom-succeed-normal	Function `transferFrom` Succeeds on Admissible Non-self Transfers
erc20-transferfrom-succeed-self	Function `transferFrom` Succeeds on Admissible Self Transfers
erc20-transferfrom-correct-amount	Function `transferFrom` Transfers the Correct Amount in Non-self Transfers
erc20-transferfrom-correct-amount-self	Function `transferFrom` Performs Self Transfers Correctly
erc20-transferfrom-change-state	Function `transferFrom` Has No Unexpected State Changes
erc20-transferfrom-fail-exceed-balance	Function `transferFrom` Fails if the Requested Amount Exceeds the Available Balance
erc20-transferfrom-correct-allowance	Function `transferFrom` Updated the Allowance Correctly
erc20-transferfrom-fail-exceed-allowance	Function `transferFrom` Fails if the Requested Amount Exceeds the Available Allowance
erc20-transferfrom-false	If Function `transferFrom` Returns `false`, the Contract's State Has Not Been Changed
erc20-totalsupply-succeed-always	Function `totalSupply` Always Succeeds
erc20-transferfrom-fail-recipient-overflow	Function `transferFrom` Prevents Overflows in the Recipient's Balance
erc20-transferfrom-never-return-false	Function `transferFrom` Never Returns `false`
erc20-balanceof-succeed-always	Function `balanceOf` Always Succeeds
erc20-totalsupply-correct-value	Function `totalSupply` Returns the Value of the Corresponding State Variable

Property Name	Title
erc20-totalsupply-change-state	Function `totalSupply` Does Not Change the Contract's State
erc20-balanceof-correct-value	Function `balanceOf` Returns the Correct Value
erc20-allowance-succeed-always	Function `allowance` Always Succeeds
erc20-balanceof-change-state	Function `balanceOf` Does Not Change the Contract's State
erc20-allowance-correct-value	Function `allowance` Returns Correct Value
erc20-allowance-change-state	Function `allowance` Does Not Change the Contract's State
erc20-approve-revert-zero	Function `approve` Prevents Giving Approvals For the Zero Address
erc20-approve-succeed-normal	Function `approve` Succeeds for Admissible Inputs
erc20-approve-correct-amount	Function `approve` Updates the Approval Mapping Correctly
erc20-approve-change-state	Function `approve` Has No Unexpected State Changes
erc20-approve-false	If Function `approve` Returns `false`, the Contract's State Has Not Been Changed
erc20-approve-never-return-false	Function `approve` Never Returns `false`

Verification Results

For the following contracts, model checking established that each of the 38 properties that were in scope of this audit (see scope) are valid:

Contract LfgToken (Source File contracts/LfgToken.sol)

Detailed results for function `transfer`

Property Name	Final Result	Remarks
erc20-transfer-revert-zero	True	
erc20-transfer-succeed-normal	True	
erc20-transfer-succeed-self	True	
erc20-transfer-correct-amount	True	
erc20-transfer-correct-amount-self	True	
erc20-transfer-change-state	True	
erc20-transfer-exceed-balance	True	
erc20-transfer-recipient-overflow	True	
erc20-transfer-never-return-false	True	
erc20-transfer-false	True	

Detailed results for function `transferFrom`

Property Name	Final Result	Remarks
erc20-transferfrom-revert-from-zero	True	
erc20-transferfrom-revert-to-zero	True	
erc20-transferfrom-succeed-normal	True	
erc20-transferfrom-succeed-self	True	
erc20-transferfrom-correct-amount	True	
erc20-transferfrom-correct-amount-self	True	
erc20-transferfrom-change-state	True	
erc20-transferfrom-fail-exceed-balance	True	
erc20-transferfrom-correct-allowance	True	
erc20-transferfrom-fail-exceed-allowance	True	
erc20-transferfrom-false	True	
erc20-transferfrom-fail-recipient-overflow	True	
erc20-transferfrom-never-return-false	True	

Detailed results for function `totalSupply`

Property Name	Final Result	Remarks
erc20-totalsupply-succeed-always	True	
erc20-totalsupply-correct-value	True	
erc20-totalsupply-change-state	True	

Detailed results for function `balanceOf`

Property Name	Final Result	Remarks
erc20-balanceof-succeed-always	True	
erc20-balanceof-correct-value	True	
erc20-balanceof-change-state	True	

Detailed results for function `allowance`

Property Name	Final Result	Remarks
erc20-allowance-succeed-always	True	
erc20-allowance-correct-value	True	
erc20-allowance-change-state	True	

Detailed results for function `approve`

Property Name	Final Result	Remarks
erc20-approve-revert-zero	True	
erc20-approve-succeed-normal	True	
erc20-approve-correct-amount	True	
erc20-approve-change-state	True	
erc20-approve-false	True	
erc20-approve-never-return-false	True	

In the remainder of this section, we list all contracts where model checking of at least one property was not successful. There are several reasons why this could happen:

- Model checking reports a counterexample that violates the property. Depending on the counterexample,this occurs if
    - The specification of the property is too generic and does not accurately capture the intended behavior of the smart contract. In that case, the counterexample does not indicate a problem in the underlying smart contract. We report such instances as being "inapplicable".

- The property is applicable to the smart contract. In that case, the counterexample showcases a problem in the smart contract and a correspond finding is reported separately in the Findings section of this report. In the following tables, we report such instances as "invalid". The distinction between spurious and actual counterexamples is done manually by the auditors.

- The model checking result is inconclusive. Such a result does not indicate a problem in the underlying smart contract. An inconclusive result may occur if
  - The model checking engine fails to construct a proof. This can happen if the logical deductions necessary are beyond the capabilities of the automated reasoning tool. It is a technical limitation of all proof engines and cannot be avoided in general.
  - The model checking engine runs out of time or memory and did not produce a result. This can happen if automatic abstraction techniques are ineffective or of the state space is too big.

Contract LfgSwapERC20 (Source File contracts/core/LfgSwapPair.sol)

Detailed results for function `transfer`

Property Name	Final Result	Remarks
erc20-transfer-succeed-normal	True	
erc20-transfer-succeed-self	True	
erc20-transfer-correct-amount	True	
erc20-transfer-revert-zero	False	
erc20-transfer-correct-amount-self	True	
erc20-transfer-change-state	True	
erc20-transfer-exceed-balance	True	
erc20-transfer-recipient-overflow	True	
erc20-transfer-false	True	
erc20-transfer-never-return-false	True	

Detailed results for function `transferFrom`

Property Name	Final Result	Remarks
erc20-transferfrom-succeed-normal	True	
erc20-transferfrom-succeed-self	True	
erc20-transferfrom-revert-from-zero	False	
erc20-transferfrom-correct-amount	True	
erc20-transferfrom-correct-amount-self	True	
erc20-transferfrom-revert-to-zero	False	
erc20-transferfrom-correct-allowance	True	
erc20-transferfrom-change-state	True	
erc20-transferfrom-fail-exceed-balance	True	
erc20-transferfrom-fail-exceed-allowance	True	
erc20-transferfrom-fail-recipient-overflow	True	
erc20-transferfrom-false	True	
erc20-transferfrom-never-return-false	True	

Detailed results for function `totalSupply`

Property Name	Final Result	Remarks
erc20-totalsupply-succeed-always	True	
erc20-totalsupply-correct-value	True	
erc20-totalsupply-change-state	True	

Detailed results for function `balanceOf`

Property Name	Final Result	Remarks
erc20-balanceof-succeed-always	True	
erc20-balanceof-correct-value	True	
erc20-balanceof-change-state	True	

Detailed results for function `allowance`

Property Name	Final Result	Remarks
erc20-allowance-succeed-always	True	
erc20-allowance-correct-value	True	
erc20-allowance-change-state	True	

Detailed results for function `approve`

Property Name	Final Result	Remarks
erc20-approve-succeed-normal	True	
erc20-approve-change-state	True	
erc20-approve-correct-amount	True	
erc20-approve-revert-zero	False	
erc20-approve-never-return-false	True	
erc20-approve-false	True	

Contract LfgSwapPair (Source File contracts/core/LfgSwapPair.sol)

Detailed results for function `transfer`

Property Name	Final Result	Remarks
erc20-transfer-succeed-normal	True	
erc20-transfer-succeed-self	True	
erc20-transfer-correct-amount	True	
erc20-transfer-revert-zero	False	
erc20-transfer-correct-amount-self	True	
erc20-transfer-exceed-balance	True	
erc20-transfer-change-state	True	
erc20-transfer-recipient-overflow	True	
erc20-transfer-false	True	
erc20-transfer-never-return-false	True	

Detailed results for function `transferFrom`

Property Name	Final Result	Remarks
erc20-transferfrom-succeed-normal	True	
erc20-transferfrom-succeed-self	True	
erc20-transferfrom-revert-from-zero	False	
erc20-transferfrom-revert-to-zero	False	
erc20-transferfrom-correct-amount	True	
erc20-transferfrom-correct-amount-self	True	
erc20-transferfrom-fail-exceed-balance	True	
erc20-transferfrom-correct-allowance	True	
erc20-transferfrom-change-state	True	
erc20-transferfrom-fail-exceed-allowance	True	
erc20-transferfrom-fail-recipient-overflow	True	
erc20-transferfrom-false	True	
erc20-transferfrom-never-return-false	True	

Detailed results for function `totalSupply`

Property Name	Final Result	Remarks
erc20-totalsupply-succeed-always	True	
erc20-totalsupply-correct-value	True	
erc20-totalsupply-change-state	True	

Detailed results for function `balanceOf`

Property Name	Final Result	Remarks
erc20-balanceof-succeed-always	True	
erc20-balanceof-correct-value	True	
erc20-balanceof-change-state	True	

Detailed results for function `allowance`

Property Name	Final Result	Remarks
erc20-allowance-succeed-always	True	
erc20-allowance-correct-value	True	
erc20-allowance-change-state	True	

Detailed results for function `approve`

Property Name	Final Result	Remarks
erc20-approve-succeed-normal	True	
erc20-approve-change-state	True	
erc20-approve-correct-amount	True	
erc20-approve-revert-zero	False	
erc20-approve-false	True	
erc20-approve-never-return-false	True	

APPENDIX | LFGSWAP

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.
Mathematical Operations	Mathematical Operation findings relate to mishandling of math formulas, such as overflows, incorrect operations etc.
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.
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.