



CERTIK

Gains Farm V2 - NFT Exchange

Security Assessment

February 19th, 2021

For :

Gains Farm V2 - NFT Exchange





Disclaimer

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

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

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

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

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

What is a CertiK report?

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

Additionally, to bridge the trust gap between project owner and users, project owner needs to express a sincere attitude with the consideration of the project team’s anonymousness. The project owner has the responsibility to notify users with the following capability of the smart contract:

- Project owner can change `G0V` and therefore update `BID_FEE_P` and `HIGHEST_BID_FEE_P` , which will lead to changes of bidding fee (5% initially) and highest bidding fee(25% initially) respectively in **GFarmTradingV2.sol**

To improve the trustworthiness of the project, any dynamic runtime changes on the protocol should be notified to clients.



Overview

Project Summaries

Project Name	GainsFarm V2
Description	DeFi
Platform	Ethereum; Solidity
Codebase	Private Repository
Commit	GFarmNFTEExchange.sol: 65473452ad85e22472ea7e0f79c030ce28d99d238641283f6eb0f4a134d1cdfa GFarmTradingV2.sol: 2dbf3bc171cffb91a5d9a96daa776275f29fc6d9fd690a16289842412579c13a 6989e60a7fd2f53d36cc367ed09c6daa94932ab2f53454ca6997615ef38891f6 6e65d0dbcfb319f451475f9b944ff10c83e76ece1c4c3d9a10a2e3ec8088ed4c

Audit Summary

Delivery Date	Feb. 19th, 2021
Method of Audit	Static Analysis, Manual Review
Consultants Engaged	2
Timeline	Feb. 10, 2021 - Feb. 15, 2021, Feb. 17, 2021 - Feb. 19, 2021

Vulnerability Summary

Total Issues	17
Total Critical	0
Total Major	0
Total Minor	4
Total Informational	13



Executive Summary

This report has been prepared for **GFarmNFTEExchange.sol** and **GFarmTradingV2.sol** smart contract to discover issues and vulnerabilities in the source code of their Smart Contract as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Dynamic Analysis, 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.

To improve overall project quality, preserve the upgradability and the ability facing the on-chain emergency issue, the following functions are adopted in the codebase:

- `set_GOV()` to update address of `GOV` in smart contract `GFarmNFTEExchange.sol`.
- `set_BID_FEE_P()` to update value of `BID_FEE_P` in smart contract `GFarmNFTEExchange.sol`.
- `set_HIGHEST_BID_FEE_P()` to update value of `HIGHEST_BID_FEE_P` in smart contract `GFarmNFTEExchange.sol`.
- `setGovFund()` to update address of `govFund` in smart contract `GFarmTradingV2.sol`
- `setNode()` to update address of `oracleAddress`, value of `jobID` and value of `linkFee` in smart contract `GFarmTradingV2.sol`
- `setMinPosEth()` to update value of `minPosEth` in smart contract `GFarmTradingV2.sol`
- `setMaxPosTokenP()` to update value of `maxPosTokenP` in smart contract `GFarmTradingV2.sol`
- `setMaxPosTokenIncreaseP()` to update value of `maxPosTokenIncreaseP` in smart contract `GFarmTradingV2.sol`
- `setFees()` to update value of `govFeeP` and value of `devFeeP` in smart contract `GFarmTradingV2.sol`
- `setSpreadP()` to update value of `spreadP` in smart contract `GFarmTradingV2.sol`
- `setLiquidationTimelock()` to update value of `liquidationSuccessTimelock` and value of `liquidationFailTimelock` in smart contract `GFarmTradingV2.sol`

The advantage of the above functions in the codebase is that the owner reserves the ability to rescue the assets in this contract under unexpected cases. It is also worthy of note the potential drawbacks of these functions, where the treasury in this contract can be migrated to any addresses or affected due to changes of variables that are abovementioned.

To improve the trustworthiness of the project, any dynamic runtime updates in the project should be notified to the community. Any plan to invoke these functions should be also considered to move to the execution queue of Timelock contract, and also emit events.



File in Scope

ID	Contract	SHA-256 Checksum
GNE	GFarmNFTEExchange.sol	65473452ad85e22472ea7e0f79c030ce28d99d238641283f6eb0f4a134d1cdfa
GFT	GFarmTradingV2.sol	2dbf3bc171cffb91a5d9a96daa776275f29fc6d9fd690a16289842412579c13a



Finding

ID	Title	Type	Severity	Resolved
GNE-01	Missing Emit Events	Optimization	Minor	
GNE-02	Lack of input validation	Volatile Code	Minor	
GNE-03	Unlocked Compiler Versions	Language Specific	Informational	
GNE-04	Inconsistent Naming Convention	Coding Style	Informational	
GNE-05	Proper Usage of "public" and "external" type	Optimization	Informational	
GNE-06	Missing Error Message	Optimization	Informational	
GNE-07	Mathematical Operations Optimization	Optimization	Informational	
GNE-08	Function <code>claim</code> Access Control	Control Flow	Informational	
GNE-09	Function <code>claimBack</code> Access Control	Control Flow	Informational	
GFT-01	Missing Emit Events	Optimization	Informational	
GFT-02	Signed SafeMath not used	Optimization	Informational	
GFT-03	SafeMath not used	Optimization	Informational	
GFT-04	Variable Name Shadowing	Optimization	Informational	
GFT-05	Function <code>claimFees</code> Access Control	Control Flow	Minor	
GFT-06	Lack of input validation	Volatile Code	Minor	
GFT-07	Unlocked Compiler Versions	Language Specific	Informational	
GFT-08	Division before multiplication	Language Specific	Informational	



GNE-01: Missing Emit Events

Type	Severity	Location
Optimization	Minor	GFarmNFTEExchange.sol:L330, L333, L336, L446, L509, L530, L545

Description:

Functions, such as `set_GOV()`, `set_BID_FEE_PC()`, `set_HIGHEST_BID_FEE_PC()`, `claim()`, `harvest()`, `stake()`, `unstake()`, that affect the status of sensitive variables should be able to emit events as notifications to customers

Recommendation:

Consider adding events for sensitive actions, and emit it in the function like below.

```
1     event SET_GOV(address indexed user, address indexed _gov);
2
3     function set_GOV(address _gov) external onlyGov{
4         GOV = _gov;
5         emit SET_GOV(msg.sender, _gov);
6     }
```

Abbreviation:

[GainsFarm] : Client agrees that there could be events for these functions and fixed in the latest commit.



GNE-02: Lack of input validation

Type	Severity	Location
Volatile Code	Minor	GFarmNFTEExchange.sol:L313

Description:

The assigned value to `GOV` should be verified as non zero value to prevent being mistakenly assigned as `address(0)` in constructor of contract `GFarmNFTEExchange` and `set_GOV()` function. Violation of this may cause losing ownership of `GOV` authorization.

Recommendation:

Check that the address is not zero by adding following checks in the constructor of contract `GFarmNFTEExchange` and `set_GOV()` function.

```
1     require(_gov != address(0));
```

Abbreviation:

[GainsFarm] : Client agreed and fixed in the latest commit.



GNE-03: Unlocked Compiler Versions

Type	Severity	Location
Language Specific	Informational	GFarmNFTEExchange.sol:L89

Description:

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 versions of library `SafeMath` and interface `IUniswapV2Pair` are instead locked at the lowest version possible that the full project can be compiled at.

Abbreviation:

[GainsFarm] : Client may not consider this will be a problem in practice, but agreed to lock the compiler version



GNE-04: Inconsistent Naming Convention

Type	Severity	Location
Coding Style	Informational	<u>GFarmNFTEExchange.sol</u>

Description:

Multiple naming convention were adopted throughout the contract codebase, such as the names of function `set_GOV()` and function `removeBidding()`. Consistency of naming convention can significantly increase the readability of the codebase and therefore enhance the quality of the project.

Recommendation:

We advise that devloper adopts consistent naming convention, either `lower_case_with_underscores` or `CapitalizedWords / mixedCase`, throughout entire codebase.

Abbreviation:

[GainsFarm] : Client improved the coding style of codebase in the latest commit.



GNE-05: Proper Usage of “public” and “external” type

Type	Severity	Location
Optimization	Informational	GFarmNFTExchange.sol:L509

Description:

`public` functions that are never called by the contract could be declared `external`. When the inputs are arrays `external` functions are more efficient than "public" functions.

Examples:

- `harvest()`

Recommendation:

Consider using the `external` attribute for functions never called from the contract.

Abbreviation:

[GainsFarm] : Client agrees that the visibility of `harvest()` should be `external`



GNE-06: Missing Error Message

Type	Severity	Location
Optimization	Informational	GFarmNFTExchange.sol:L326

Description:

Missing error message in `require(msg.sender == GOV);` in modifier `onlyGov()`

Recommendation:

Consider adding error message in the `require` check.

Abbreviation:

[GainsFarm] : Client agrees that there could be an error message for `require` in modifier `onlyGov()`



GNE-07: Mathematical Operations Optimization

Type	Severity	Location
Optimization	Informational	GFarmNFTExchange.sol:L582

Description:

Discussion: What's the `1e5` representing here in the formula `reserveUSDC.mul(1e12).mul(1e5).div(reserveETH);`

According to the Ethereum yellowpaper, each operation will consume gas in the smart contract. Multiple `mul` operation can be combined in to single `mul`

Recommendation:

Combine `mul(1e12).mul(1e5)` into `mul(1e17)` in function `getEthPrice()` to significantly decrease the gas consumption.

Abbreviation:

[GainsFarm] : Client agrees that it can be combined in `mul(1e17)`



GNE-08: Function `claim` Access Control

Type	Severity	Location
Control Flow	Informational	GFarmNFTExchange.sol:L446

Description:

The function `claim()` better only access by `highestBidder` or `onlyGov`

Recommendation:

```
1 function claim(uint _nftID) external notContract listed(_nftID){
2     require(b.highestBidder == msg.sender, "Only the highest bidder can claim the listed NFT");
3     ...
4 }
```

Abbreviation:

[GainsFarm] : Client believes there's no reason anyone can't execute the tx. The tx will send the ETH the seller and the NFT to the bidder anyway.



GNE-09: Function `claimBack` Access Control

Type	Severity	Location
Control Flow	Informational	GFarmNFTExchange.sol:L478

Description:

The function `claimBack()` better only access by `seller` or `onlyGov`

Recommendation:

```
1 function claimBack(uint _nftID) external notContract listed(_nftID){
2     require(b.seller == msg.sender || msg.sender == _gov, "Only seller or _gov can claim back
   the listed NFT");
3     ...
4 }
```

Abbreviation:

[GainsFarm] : Client believes there's no reason anyone can't execute the tx. The tx will send the ETH to the seller and the NFT to the seller anyway.



GFT-01: Missing Emit Events

Type	Severity	Location
Optimization	Informational	GFarmTradingV2.sol

Description:

Function that affect the status of sensitive variables should be able to emit events as notifications to customers.

Examples:

- `setGovFund()`
- `setToken()`
- `setLp()`
- `setNft()`
- `setMinPosEth()`
- `setMaxPosTokenP()`
- `setMaxPosTokenIncreaseP()`
- `setFees()`
- `setSpreadP()`
- `setLiquidationTimelock()`
- `removePair()`
- `addPair()`
- `setNode()`
- `pause()`

Recommendation:

Consider adding events for sensitive actions, and emit each in the function like below.

```
1     event SetNodeEvent(address indexed user, address indexed _node);
2
3     function SetNodeEvent(address _node) external onlyGov{
4         require( _node != address(0) && _node != node, "Invalid address")
5         node = _node;
6         emit SetGovFund(msg.sender, _node);
7     }
```

Abbreviation:

[GainsFarm] : Client agrees that there could be events for these functions and updated in the last commit.



GFT-02: Signed SafeMath not used

Type	Severity	Location
Mathematical Operations	Minor	GFarmTradingV2.sol

Description:

Signed SafeMath from OpenZeppelin is not used on this two instances making it possible for overflow/underflow

Recommendation:

Considering use OpenZeppelin's Signed SafeMath library for all of the `int` operations.

Reference:

<https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/math/SignedSafeMath.sol>

Abbreviation:

[GainsFarm] : Client agrees that signed safe math library could be used. The gas optimization also an important vectors in the current contracts design. With all the consideration, client has been well thought the variables design given and cited the ranges of all variables in the contract.



GFT-03: SafeMath not used

Type	Severity	Location
Mathematical Operations	Minor	GFarmTradingV2.sol

Description:

SafeMath from OpenZeppelin is not used on this two instances making it possible for overflow/underflow

Recommendation:

Considering use OpenZeppelin's SafeMath library for all of the `uint` operations.

Reference:

<https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/math/SafeMath.sol>

Abbreviation:

[GainsFarm] : Client doesn't use safemath in some instances as it doesn't exist for other types of `uint` than `uint256`, and it's impossible in practice that it could overflow. With all the consideration, client has been well thought the variables design given and cited the ranges of all variables in the contract.



GFT-04: Variable Name Shadowing

Type	Severity	Location
Optimization	Informational	GFarmTradingV2.sol L50

Description:

The variable name of `oracle` is shadowing same name variable in `ChainlinkClient.sol`

Recommendation:

Rename the variable of `oracle` in `GFarmTradingV2.sol`

Abbreviation:

[GainsFarm] : Client updated name from `oracle` to `oracleAddress` in the latest commit.



GFT-05: Function `claimFees` Access Control

Type	Severity	Location
Control Flow	Minor	GFarmTradingV2.sol:L639

Description:

The function `claimFees()` can be called any addresses for minting fund to dev and gov addresses.

Recommendation:

```
1 function claimFees() external onlyGov{
2     ...
3 }
```

Abbreviation:

[GainsFarm] : Client believes any user except gov fund or dev fund addresses will waste gas to call function `claimFees()` , as it will simply transfer fees to the client.



GFT-06: Lack of input validation

Type	Severity	Location
Volatile Code	Minor	GFarmTradingV2.sol

Description:

The assigned value to `_GOV` , `_DEV` , `_gov` , `_oracle` should be verified as non zero value to prevent being mistakenly assigned as `address(0)` in constructor of contract `GFarmTradingV2` , `setGovFund()` and `setNode()` function. Violation of this may cause losing ownership of `govFund` and `devFund` authorization.

Recommendation:

Check that the address is not zero by adding following checks in the constructor of contract `GFarmTradingV2` , `setGovFund()` and `setNode()` function.

```
1     require(_gov != address(0));
```

Abbreviation:

[GainsFarm] : Client agrees that there should be a check to see if the new gov address isn't the `address(0)` and fixed in the latest commit.



GFT-07: Unlocked Compiler Versions

Type	Severity	Location
Language Specific	Informational	GFarmTradingV2.sol

Description:

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 versions of entire codebase locked at the lowest version possible that the full project can be compiled at.

Abbreviation:

[GainsFarm] : It is not a problem in practice because contract deployer specified a precise version in truffle to compile the contracts, but agreed to lock the compiler version



GFT-08: Division before multiplication

Type	Severity	Location
Language Specific	Informational	GFarmTradingV2.sol L300, L301

Description:

Mathematical operations in the aforementioned lines perform divisions before multiplications. Performing multiplication before division can sometimes avoid loss of precision.

Recommendation:

We recommend applying multiplications before divisions if integer overflow would not happen.

Abbreviation:

[GainsFarm] : Client agrees the finding item and fixed the issue in the last commit.

Appendix

Finding Categories

Gas Optimization

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

Mathematical Operations

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

Logical Issue

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

Control Flow

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

Volatile Code

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

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 instorage 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 and 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.

Magic Numbers

Magic Number findings refer to numeric literals that are expressed in the codebase in their raw format and should otherwise be specified as `constant` contract variables aiding in their legibility and maintainability.


Compiler Error


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


Dead Code

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

Icons explanation

 : Issue resolved

 : Issue not resolved / Acknowledged. The team will be fixing the issues in the own timeframe.

 : Issue partially resolved. Not all instances of an issue was resolved.