Security Assessment

Gains.Farm V2

Apr 3rd, 2021

Summary

This report has been prepared for Gains.Farm v2 - GFarmTokenMumbai & GFarmNftSwap smart contracts, 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.

The security assessment resulted in 6 findings that ranged from major 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 given they are currently missing in the repository;
- Provide more comments per each function for readability, especially contracts are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	Gains.Farm V2
Description	
Platform	matic
Language	Solidity
Codebase	Compressed Package
Commits	1. GFarmTokenMumbai.sol: 2c94c010918f4732e5697c5a2efd11a31a45a38fa19c47cdfaeb0a623de766a5 2. GFarmNftSwap.sol 16966e230d12a3f645e2a9728a2e7303344ee33c63449bb7076a3cd4638a502a, 2c94c010918f4732e5697c5a2efd11a31a45a38fa19c47cdfaeb0a623de766a5

Audit Summary

Delivery Date	Apr 03, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

Vulnerability Summary

Total Issues	6
Critical	0
Major	0
Minor	1
Informational	5
Discussion	0

Audit Scope

ID	file	SHA256 Checksum
GFN	GFarmNftSwap.sol	16966e230d12a3f645e2a9728a2e7303344ee33c63449bb7076a3cd4638a502a
GFT	GFarmTokenMumbai.sol	2c94c010918f4732e5697c5a2efd11a31a45a38fa19c47cdfaeb0a623de766a5

Centralization

The client adopted a role-based access control mechanism to restrict accesses to the following sensitive functions in contract `GFarmTokenMumbai.sol`:

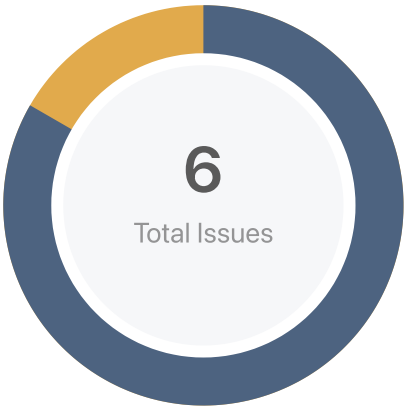
- `initiateGrantRequest()`
- `cancelGrantRequest()`
- `executeGrantRequest()`
- `mint()`
- `burn()`
- `deposit()`
- `withdraw()`

All the roles except `DEFAULT_ADMIN_ROLE` can only be granted by the address of `DEFAULT_ADMIN_ROLE` through functions `initiateGrantRequest()`, `cancelGrantRequest()` and `executeGrantRequest()`, where `grantRole()` is disabled to prevent any other possible way to grant roles. The client additionally added a layer of Timelock above the role-based access control mechanism with 1.5 days delay for all invocations of the abovementioned functions, which provided a notification window about any pending actions of the role grant to the community.

Considering sensitive functions such as `mint()`, `burn()`, `deposit()`, `withdraw()` can only be accessed by specific roles, Timelock assisted role-control system that is implemented in the contract `GFarmTokenMumbai.sol` can be regarded as the client's effort on reducing the centralization concerns from the community.

Besides, `gov` role is adopted in GFarmNftSwap.sol to set the implementations of GFarmBridgeableNftInterface. They can only be set once by the contract deployer along with the setting of implementation of `GFarmNftInterface` when deploying the `GFarmNftSwap.sol`. With the restriction of Timelock, the project community can have a certain time window to know the implementations of `GFarmBridgeableNftInterface` and `GFarmNftInterface`.

Findings



Critical	0 (0.00%)
Major	0 (0.00%)
Minor	1 (16.67%)
Informational	5 (83.33%)
Discussion	0 (0.00%)

ID	Title	Category	Severity	Status
GFN-1	Missing Error Message	Logical Issue	Informational	Resolved
GFN-2	Missing Emits Events	Coding Style	Informational	Resolved
GFN-3	Code Simplify	Coding Style	Informational	Resolved
GFN-4	Missing Emits Events	Coding Style	Informational	Resolved
GFN-5	Code Simplify	Coding Style	Informational	Resolved
GFN-6	Lack of Input Validation	Logical Issue	Minor	Resolved

GFN-1 | Missing Error Message

Category	Severity	Location	Status
Logical Issue	Informational	GFarmNftSwap.sol: 32	Resolved

Description

The require statement is missing a clear error message

Recommendation

We recommend adding error message for better error tracking, and actionable by the users

Alleviation

The client heeded the advice and added the error message in the newer version GFarmNftSwap.sol with hash `2c94c010918f4732e5697c5a2efd11a31a45a38fa19c47cdfaeb0a623de766a5`.

GFN-2 | Missing Emits Events

Category	Severity	Location	Status
Coding Style	Informational	GFarmNftSwap.sol: 55	Resolved

Description

Function getBridgeableNft which will invoke the swap. Missing event makes it difficult to track off-chain parameter changes. An event should be emitted for significant transactions like this.

Recommendation

We recommend emitting an event to log the swap feature in getBridgeableNft.

Alleviation

The client heeded the advice and added emit event to the swap feature in `getBridgeableNft()` in the newer version GFarmNftSwap.sol with hash `2c94c010918f4732e5697c5a2efd11a31a45a38fa19c47cdfaeb0a623de766a5`.

GFN-3 | Code Simplify

Category	Severity	Location	Status
Coding Style	Informational	GFarmNftSwap.sol: 65~72	Resolved

Description

The aforementioned lines can be reused the same code from function `bridgeableNft()`

Recommendation

We recommend following updates:

1. change the function visibility of `bridgeableNft()` or create a reusable internal function
2. replace the aforementioned line by `bridgeableNft()` to increase the reusability of code.

Alleviation

The client heeded the advice and simplified the code by reusing the code of `bridgeableNft()` in the newer version GFarmNftSwap.sol with hash `2c94c010918f4732e5697c5a2efd11a31a45a38fa19c47cdfaeb0a623de766a5`.

GFN-4 | Missing Emits Events

Category	Severity	Location	Status
Coding Style	Informational	GFarmNftSwap.sol: 82	Resolved

Description

Function `getNft()` which will invoke the swap bridgeable nft for unbridgeable nft . Missing event makes it difficult to track off-chain parameter changes. An event should be emitted for significant transactions like this.

Recommendation

We recommend emitting an event to log the swap bridgeable nft for unbridgeable nft in `getNft()`.

Alleviation

The client heeded the advice and added the events to the `getNft()` function in the newer version GFarmNftSwap.sol with hash `2c94c010918f4732e5697c5a2efd11a31a45a38fa19c47cdfaeb0a623de766a5`.

GFN-5 | Code Simplify

Category	Severity	Location	Status
Coding Style	Informational	GFarmNftSwap.sol: 92~96	Resolved

Description

The aforementioned lines can be reused the same code from function `getNft()`

Recommendation

We recommend following updates:

1. change the function visibility of `getNft()` or create a reusable internal function
2. replace the aforementioned line by `getNft()` to increase the reusability of code.

Alleviation

The client heeded the advice and simplified the code by reusing the code of `getNft()` in the newer version GFarmNftSwap.sol with hash `2c94c010918f4732e5697c5a2efd11a31a45a38fa19c47cdfaeb0a623de766a5`.

GFN-6 | Lack of Input Validation

Category	Severity	Location	Status
Logical Issue	Minor	GFarmNftSwap.sol: 47, 55, 59, 72, 77, 79, 86, 93	Resolved

Description

Based on the comments, the uint value of nftType must be in range from 1 to 5. If the nftType is out of the specific range, the token will be minted to address(0) with any code that is similar to the one at L55

Recommendation

We advise the client to add the following check at the beginning of the function `getBridgeableNft()`, getBridgeableNfts()`, getNft()`,getNfts()`

```
require(nftType > 0 && nftType < 6, "nftType must be in range of 1 to 5");
```

Alleviation

The client heeded the advice and added the modifier `correctNftType(uint nftType)` to validate the input of `nftType` in the newer version GFarmNftSwap.sol with hash `2c94c010918f4732e5697c5a2efd11a31a45a38fa19c47cdfaeb0a623de766a5`.

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 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 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.

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 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 without CertiK's prior written consent.

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.

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.