CERTIK

CertiK Audit Report for GAMA MINING

CertiK Audit Report for

GAMA MINING

Request Date: 2020-09-21

Revision Date: 2020-09-30

Platform Name: EVM

Contents

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 Verification Services Agreement between CertiK and GAMA(the "Company"), or the scope of services/verification, and terms and conditions provided to the Company in connection with the verification (collectively, 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.

About CertiK

CertiK is a technology-led blockchain security company founded by Computer Science professors from Yale University and Columbia University built to prove the security and correctness of smart contracts and blockchain protocols.

CertiK, in partnership with grants from IBM and the Ethereum Foundation, CertiK's mission of every audit is to apply different approaches and detection methods, ranging from manual, static, and dynamic analysis, to ensure that projects are checked against known attacks and potential vulnerabilities. CertiK leverages a team of seasoned engineers and security auditors to apply testing methodologies and assessments to each project, in turn creating a more secure and robust software system.

CertiK has served more than 100 clients with high quality auditing and consulting services, ranging from stablecoins such as Binance's BGBP and Paxos Gold to decentralized oracles

such as Band Protocol and Tellor. CertiK customizes its engineering tool kits, while applying cutting-edge research on smart contracts, for each client on its project to offer a high quality deliverable. For more information: https://certik.io.

Executive Summary

This report has been prepared for **GAMA** 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.

Testing Summary

SECURITY LEVEL

File	Vulnerabilities	Result
gama_mining.sol	3 Minor	Pass
gama_token.sol	0	Pass
Timelock.sol	0	Pass

Smart Contract Audit

This report has been prepared as a product of the Smart Contract Audit request by GAMA.

This audit was conducted to discover issues and vulnerabilities in the source code of GAMA's smart contract.

TYPE	Smart Contract
SOURCE CODE	mining project in gama_code.zip
PLATFORM	EVM
LANGUAGE	Solidity
REQUEST DATE	Sep 21, 2020
DELIVERY DATE	Sep 30, 2020
METHODS	A comprehensive examination has been performed using Dynamic Analysis, Static Analysis, and Manual Review.

Review Notes

Introduction

CertiK team was contracted by the **GAMA** team to audit the design and implementation of their smart contract of gama mining.

The audited source code link is:

- Source Code:
  mining project in gama_code.zip
  Source Code MD5 Checksum
  3a95a723e309e3279e1f1881f86445c1

The goal of this audit was to review the Solidity implementation for its business model, study potential security vulnerabilities, its general design and architecture, and uncover bugs that could compromise the software in production.

The findings of the initial audit have been conveyed to the team behind the contract implementations and the source code is expected to be re-evaluated before another round of auditing has been carried out.

Documentation

The sources of truth regarding the operation of the contracts in scope were lackluster and **are something we advise to be enriched to aid in the legibility of the codebase as well as project**. To help aid our understanding of each contract's functionality we referred to in-line comments and naming conventions.

These were considered the specification, and when discrepancies arose with the actual code behaviour, we consulted with the **GAMA** team or reported an issue.

Summary

Certain optimization steps that we pinpointed in the source code mostly referred to coding standards and inefficiencies, however **3 minor vulnerabilities were identified during our audit that solely concerns the specification**.

Certain discrepancies between the expected specification and the implementation of it were identified and were relayed to the team, however they pose no type of vulnerability and concern an optional code path that was unaccounted for.

Recommendations

Overall, the codebase of the contracts should be refactored to assimilate the findings of this report, enforce linters and / or coding styles as well as correct any spelling errors and mistakes that appear throughout the code **to achieve a high standard of code quality and security**.

Findings

Exhibit 1

TITLE	TYPE	SEVERITY	LOCATION
Unlocked Compiler Version Declaration	Language Specific Issue	Informational	gama_mining.sol, gama_token.sol, Timelock.sol

[INFORMATIONAL] Description:

The compiler version utilized throughout the project uses the "^" prefix specifier, denoting that a compiler version which is greater than the version will be used to compile the contracts. Recommend the compiler version should be consistent throughout the codebase.

Recommendations:

It is a general practice to instead lock the compiler at a specific version rather than allow a range of compiler versions to be utilized to avoid compiler-specific bugs and be able to identify ones more easily. We recommend locking the compiler at the lowest possible version that supports all the capabilities wished by the codebase. This will ensure that the project utilizes a compiler version that has been in use for the longest time and as such is less likely to contain yet-undiscovered bugs.

(GAMA - resolved) The issue is addressed in "gama_code_20200927.zip".
Source Code MD5 Checksum d75aacb7a8dec09d0742a9303e3ff3c4

Exhibit 2

TITLE	TYPE	SEVERITY	LOCATION
Declare Events	Optimization	Minor	gama_mining.sol: L213-L216

[MINOR] Description:

```
function safeGamaTransfer(address _to, uint256 _amount) internal {
      uint256 gamaBal = gama.balanceOf(address(this));
      if (_amount > gamaBal) {
         gama.transfer(_to, gamaBal);
      } else {
         gama.transfer(_to, _amount);
      }
   }
```

According to the “safeGamaTransfer”, its intention is to transfer the “_amount” from the gama token contract to the desired “_to” address. However, when the requested transfer “_amount” is greater than the “gamaBal”, it will just forward all the remaining balance to the “_to” address.

Recommendations:

Consider declaring an event to emit the “_amount” and the “gamaBal” when the pool does not have enough GAMAs due to the rounding error.

Exhibit 3

TITLE	TYPE	SEVERITY	LOCATION
Array Index Out of Bound	Optimization	Informational	gama_mining.sol: L955,L993,L1016,L1034,L1049,L1063,L1097

[INFORMATIONAL] Description:

PoolInfo storage pool = poolInfo[_pid];

If _pid is invalid, the above sentence will throw an error.

Recommendations:

Consider checking if “_pid” is valid as following:

require(_pid<poolInfo.length, “pool doesn’t exist!”);

Exhibit 4

TITLE	TYPE	SEVERITY	LOCATION
Declare Events	Optimization	Informational	gama_mining.sol : L912,L951, L1015

[INFORMATION]Description:

Several sensitive actions are defined without event declarations.

1. Constructor will set several attributes, and emit an event would be better.
2. The owner could change the allocPoint of the pool at any time. The allocPoint will influence the rewards for users. Add an emit event.
3. The update operation is pivotal, there should be an emit event before the end of the function.

Recommendations:

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

Exhibit 5

TITLE	TYPE	SEVERITY	LOCATION
Redundant address keyword	Optimization	Informational	gama_mining.sol : L1041,L1057,L1065,L1117,L1139

[INFORMATION]Description:

Msg.sender is an address already. Use address keyword wrap up is meaningless and thus will spend gas.

Recommendations:

Consider removing the address keyword for ”msg.sender”.

Exhibit 6

TITLE	TYPE	SEVERITY	LOCATION
Duplicated codes	Coding Style	Informational	gama_mining.sol: L9,L768,L796 gama_token.sol: L32, L13, L731

[INFORMATIONAL] Description:

Same contracts are declared in both gama_mining.sol and gama_token.sol. It's better to declare them in independent files and import the file where it is used.

Interface : IERC20

Abstract contract : Context

Contract: Ownable

Recommendations:

Consider declaring these contracts in independent files.

Exhibit 7

TITLE	TYPE	SEVERITY	LOCATION
Incorrect Naming Convention Utilization	Coding Style	Informational	gama_mining.sol: L933,L1085,L1096

[INFORMATIONAL] Description:

Solidity defines a naming convention that should be followed. In general, parameters should use mixedCase, refer to:

https://solidity.readthedocs.io/en/v0.6.0/style-guide.html#naming-conventions

Examples:

- Parameters like: _new_gama_wwner, _ctoken_amount

Additionally, there are some typos inside some parameters.

Examples:

- _new_gama_wwner , _poolTyep

Recommendations:

The recommendations outlined here are intended to improve the readability, and thus they are not rules, but rather guidelines to try and help convey the most information through the names of things.

Exhibit 8

TITLE	TYPE	SEVERITY	LOCATION
Proper Usage of “public” and “external” type	Coding Style	Informational	gama_mining.sol:L933,L951, L1033,L1048,L1085,L1091,L 1096

[INFORMATIONAL] Description:

“public” functions that are never called by the contract should be declared “external” . When the inputs are arrays “external” functions are more efficient than “public” functions.

Examples:

- Functions like : add() , set() , deposit() , withdraw() , transferGamaOwner() , approveCTokenUnderlyingAsset() , convertCtokenToGtokenAndDeposit()

Recommendations:

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

Exhibit 9

TITLE	TYPE	SEVERITY	LOCATION
Unused Library	Coding Style	Informational	gama_mining.sol: L537

[INFORMATIONAL] Description:

Library EnumerableSet is declared in gama_mining.sol but it is never used.

Recommendations:

Consider removing unused libraries.

(GAMA - resolved) The issue is addressed in “gama_code_20200927.zip”.
Source Code MD5 Checksum d75aacb7a8dec09d0742a9303e3ff3c4

Exhibit 10

TITLE	TYPE	SEVERITY	LOCATION
Unused Variable	Ineffectual Code	Informational	gama_mining.sol: L663

[INFORMATIONAL] Description:

An unused variable is declared. Remove or comment out the variable name.

Examples:

- Variable “initGamaToken”

Recommendations:

Consider removing the unused variable.

Exhibit 11

TITLE	TYPE	SEVERITY	LOCATION
Unlimited mint	Optimization	Minor	gama_mining.sol: L663

[MINOR] Description:

There is no cap for the GamaToken. As a liquidity mining project, it's better to set a finite value.

Recommendations:

Consider adding a cap value for the GamaToken, and setting the cap to a finite value, referring to below link:

https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC20/ERC20Capped.sol

(GAMA - confirmed) Minted token amount is restricted to the block height. Minting will be terminated when the block height meets some extent.

Exhibit 12

TITLE	TYPE	SEVERITY	LOCATION
Security risk of transferring underlying assets	Security	Minor	gama_mining.sol: L695

[MINOR] Description:

The mining contract can be attacked by maliciously manipulating the “add” function.
For example, deploying contracts like below and setting the contract addresses as the “_lpToken” and the “_cToken” when calling the “add” function.

```
contract hack1{
   function underlying() public view returns(address){
      return 0xWETH;
   }

   function hack() public{
      underlyingToken.transferFrom(miningAddress, hackAccount, amount);
   }
}
contract hack2{
   function underlying() public view returns(address){
      return 0xWETH;
   }
}
```

Then this contract “hack1” can migrate the underlying assets by calling the public function “transferFrom”.

Recommendations:

Although the “add” function can only be called by the owner, we still recommend to add emit events and timelock on the “add” function. And consider adding below checks:

require(_lpToken != cToken, "gToken and cToken address cannot be the same"）;

(GAMA - confirmed) After the deployment of the gama mining contract, will transfer the ownership to the Timelock contract. No plan to add GovernorAlpha contract to inherit Timelock feature so far. Plan to do after the deployment.
(GAMA - confirmed) Will add emit events for the “add” function.

Exhibit 13

TITLE	TYPE	SEVERITY	LOCATION
Missing require	Optimization	Informational	gama_mining.sol:L962

[INFORMATIONAL] Description:

Error message is returned when parameter “_from” is greater than “_to” in the function “getMultiplier”, it’s better to add a check with an accurate error message instead of mathematical errors.

Recommendations:

Consider adding a check between “_from” and “_to”:

require(_from<=_to,”_to cannot be smaller than _from”);