QuillAudits

Audit Report
August, 2021

GAIN

Contents

Scope of Audit	01
Techniques and Methods	02
Issue Categories	03
Issues Found – Code Review/Manual Testing	04
Automated Testing	13
Disclaimer	20
Summary	21

Scope of Audit

The scope of this audit was to analyze and document the GainPool smart contract codebase for quality, security, and correctness.

Checked Vulnerabilities

We have scanned the smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that we considered:

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- Unsafe type inference
- Implicit visibility level

Techniques and Methods

Throughout the audit of smart contract, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

Structural Analysis
In this step we have analyzed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.
SmartCheck.

Static Analysis
Static Analysis of Smart Contracts was done to identify contract vulnerabilities. In this step a series of automated tools are used to test security of smart contracts.

Code Review / Manual Analysis
Manual Analysis or review of code was done to identify new vulnerability or verify the vulnerabilities found during the static analysis. Contracts were completely manually analyzed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of automated analysis were manually verified.

Gas Consumption
In this step we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit
Remix IDE, Truffle, Truffle Team, Ganache, Solhint, Mythril, Slither, SmartCheck.

Issue Categories

Every issue in this report has been assigned with a severity level. There are four levels of severity and each of them has been explained below.

High severity issues

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality and we recommend these issues to be fixed before moving to a live environment.

Medium level severity issues

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems and they should still be fixed.

Low level severity issues

Low level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are severity four issues which indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Number of issues per severity

Type	High	Medium	Low	Informational
Open	0	0	0	0
Acknowledged	0	0	0	3
Closed	0	0	1	6

Introduction

During the period of **July 20, 2021 to August 16, 2021** - QuillAudits Team performed a security audit for GainPool smart contracts.
The code for the audit was provided by GainPool and their hashes are mentioned below:
Github Link: https://github.com/GainPool/gainpool
Git Branch: master
Commit Hash: 5822F4C0161DE5E539B410404117F0C3
Updated Commit Hash: 104100cc15514ae0323ebbbec6c4fffda392bab9
Updated Commit Hash (v3):
6333753b8409340ea2b1c4d7c0ed65ba4d0b1d56

File 1: GainToken.sol
Smart Contract Online Code: https://github.com/GainPool/gainpool/blob/main/contracts/GainToken.sol
MD5 hash: E1B5FF933CCB483808867233898363E0
Updated MD5 hash: ADD735038CFD809A9DD70C47E7849266

File 2: Staking.sol
Smart Contract Online Code: https://github.com/GainPool/gainpool/blob/main/contracts/Staking.sol
MD5 hash: A7AE23A7EE9D205B96811CF0B2CBE698
Updated MD5 hash: A812F4579FD55C8290677B72BE8AC873
Updated MD5 hash(v3): 4C819624F376D9E736E29085E82F2C52

Issues Found – Code Review / Manual Testing

A. Contract - GainToken

High severity issues

No issues were found.

Medium severity issues

No issues were found.

Low level severity issues

No issues were found.

Informational

1. **Use the latest solidity version**

   pragma solidity 0.5.16;

   Description
   Using the latest solidity will prevent any compiler-level bugs.

   Remediation
   Please use 0.8.6, which is the latest version.

   Status: Fixed

2. **Warning: SPDX license identifier**

   Description
   SPDX license identifier not provided in the source file, if version changed to >= 0.6.7.

   Remediation
   SPDX-License-Identifier , if version changed to >= 0.6.7.

   Status: Fixed

B. Contract - Staking

High severity issues

No issues were found.

Medium severity issues

No issues were found.

Low level severity issues

1. Use the latest solidity version

Line	Code

148	// distribute rewards
	function distributeRewards() public onlyOwner {
	for (uint256 s = 0; s < stakeholders.length; s += 1) {
	address stakeholder = stakeholders[s];
	uint256 reward = calculateReward(stakeholder);
	rewards[stakeholder] = rewards[stakeholder].add(reward);
	}
	}

99	function isStakeholder(address _address) public view returns (bool, uint256)
	{
	for (uint256 s = 0; s < stakeholders.length; s += 1) {
	if (_address == stakeholders[s]) return (true, s);
	}
	return (false, 0);
	}

Description

In distributeRewards, isStakeholder functions for loop do not have a stakeholders length limit, which costs more gas.

Remediation

The upper limit should be limited for loops.

Status: Fixed

Informational

2. Use the latest solidity version

pragma solidity 0.5.16;

Description
Using the latest solidity will prevent any compiler-level bugs.

Remediation
Please use 0.8.6, which is the latest version.

Status: Fixed

3. Warning: SPDX license identifier

Description
SPDX license identifier not provided in the source file, if version changed to >= 0.6.7.

Remediation
SPDX-License-Identifier , if version changed to >= 0.6.7.

Status: Fixed

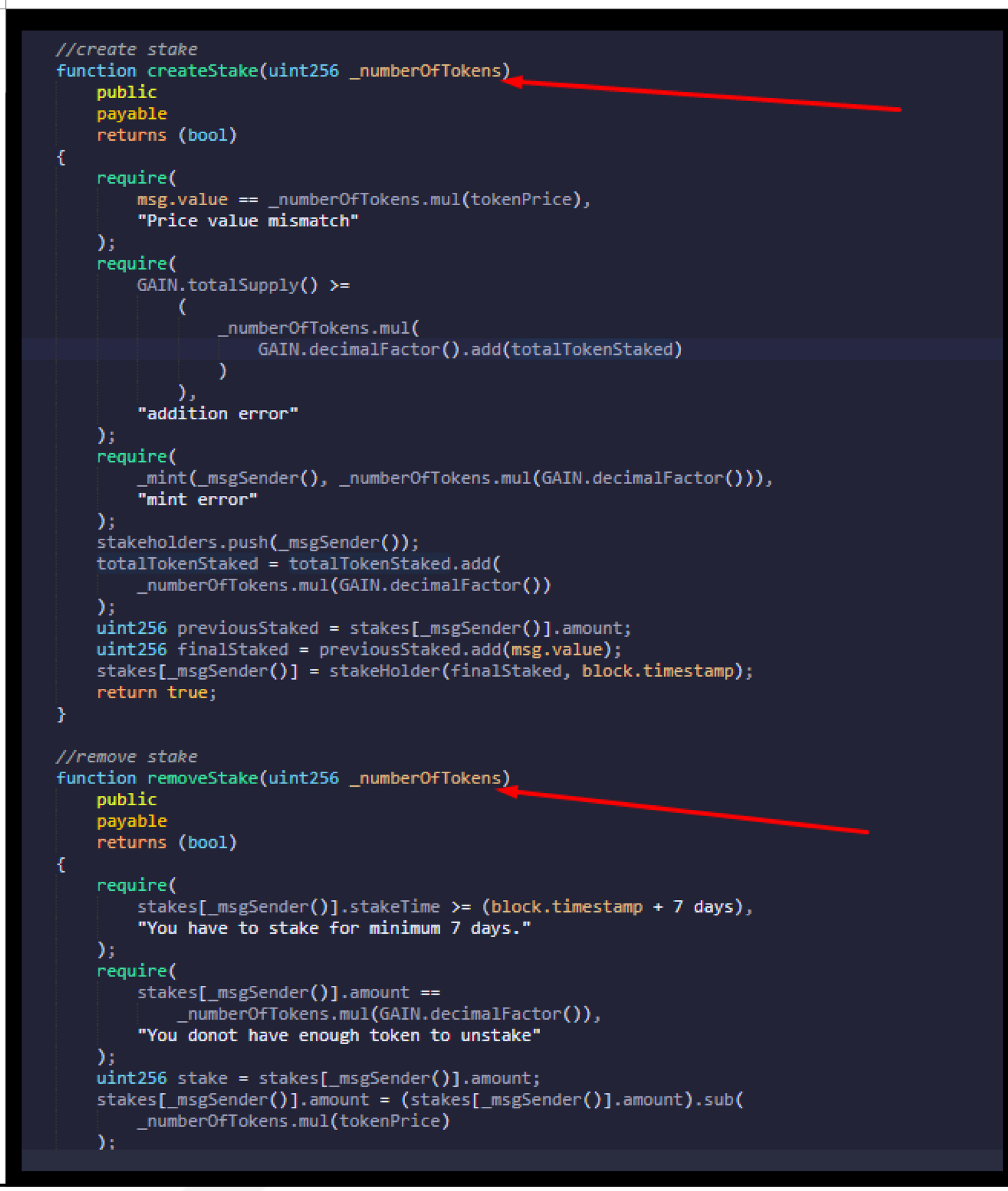
4. Critical operation lacks event log

Description
Missing event log for withdrawReward, createStake,removeStake, removeStakeholder.

Remediation
Please write an event log for listed events.

Status: Acknowledged by the Auditee

5. Function input parameters lack of check

Line	Code
33	

```
//create stake
function createStake(uint256 _numberOfTokens)
    public
    payable
    returns (bool)
{
    require(
        msg.value == _numberOfTokens.mul(tokenPrice),
        "Price value mismatch"
    );
    require(
        GAIN.totalSupply() >=
            (
                _numberOfTokens.mul(
                    GAIN.decimalFactor().add(totalTokenStaked)
                )
            ),
        "addition error"
    );
    require(
        _mint(_msgSender(), _numberOfTokens.mul(GAIN.decimalFactor())),
        "mint error"
    );
    stakeholders.push(_msgSender());
    totalTokenStaked = totalTokenStaked.add(
        _numberOfTokens.mul(GAIN.decimalFactor())
    );
    uint256 previousStaked = stakes[_msgSender()].amount;
    uint256 finalStaked = previousStaked.add(msg.value);
    stakes[_msgSender()] = stakeHolder(finalStaked, block.timestamp);
    return true;
}

//remove stake
function removeStake(uint256 _numberOfTokens)
    public
    payable
    returns (bool)
{
    require(
        stakes[_msgSender()].stakeTime >= (block.timestamp + 7 days),
        "You have to stake for minimum 7 days."
    );
    require(
        stakes[_msgSender()].amount ==
            _numberOfTokens.mul(GAIN.decimalFactor()),
        "You donot have enough token to unstake"
    );
    uint256 stake = stakes[_msgSender()].amount;
    stakes[_msgSender()].amount = (stakes[_msgSender()].amount).sub(
        _numberOfTokens.mul(tokenPrice)
    );
```

Description

Variable validation is not performed in these functions: createstake, removeStake

Remediation

Put validation : variable is not empty and > 0

Status: Acknowledged by the Auditee

6. Make variables constant

Line	Code
19	uint256 public tokenPrice = 10000000000000000;
21	uint256 public APY = 530; // 5.3%

Description
APY, tokenPrice variable values will be unchanged.

Remediation
Please add the "constant" keyword.

Status: Fixed

7. Error message correction

Line	Code

517	`//remove stake`
	`function removeStake(uint256 _numberOfTokens)`
	`  public`
	`  payable`
	`  returns (bool)`
	`{`
	`  require(`
	`  stakes[_msgSender()].stakeTime >= (block.timestamp + 7 days),`
	`  "You have to stake for minimum 7 days."`
	`  );`
	`  require(`
	`  stakes[_msgSender()].amount ==`
	`  _numberOfTokens.mul(GAIN.decimalFactor()),`
	`  "You donot have enough token to unstake"`
	`  );`

Description
In removeStake function, the error message is not correct.

Remediation
It should be like - You do not have enough tokens to unstake.

Status: Acknowledged by the Auditee

8. Ambiguous error message

Line	Code
490	require(msg.value == _numberOfTokens.mul(tokenPrice), "Price value mismatch");

Description
The above-mentioned error message does not explain exactly the error of the operation.

Remediation
As error messages are intended to notify users about failing conditions, they should provide enough information to make appropriate corrections to interact with the system.

Uninformative error messages greatly damage the overall user experience, thus lowering the system's quality. Consider fixing the specific instances mentioned above and reviewing the entire codebase to make sure every error message is informative and user-friendly.

Status: Fixed

Functional test

File : GainToken.sol

Function Names	Testing results
getOwner	Passed
decimals	Passed
decimalFactor	Passed
symbol	Passed
name	Passed
totalSupply	Passed
balanceOf	Passed
transfer	Passed
allowance	Passed
approve	Passed
transferFrom	Passed
increaseAllowance	Passed
decreaseAllowance	Passed
_transfer	Passed
_approve	Passed
_mint	Passed
_burn	Passed

File : Staking.sol

Function Names	Testing results
createStake	Function input parameters lack of check, Critical operation lacks event log
removeStake	Function input parameters lack of check, Error message correction, Critical operation lacks event log
stakeOf	Passed
isStakeholder	Passed
removeStakeholder	Passed
rewardOf	Passed
calculateReward	Passed
viewReward	Passed
distributeRewards	Passed
withdrawReward	Passed

Automated Testing

Slither

Slither log >> GainToken.sol

```
INFO:Detectors:
Contract locking ether found:
        Contract GainToken (GainToken.sol#294-458) has payable functions:
         - GainToken.constructor() (GainToken.sol#307-313)
        But does not have a function to withdraw the ether
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#contracts-that-lock-ether
INFO:Detectors:
GainToken.allowance(address,address).owner (GainToken.sol#351) shadows:
        - Ownable.owner() (GainToken.sol#259-261) (function)
GainToken._approve(address,address,uint256).owner (GainToken.sol#431) shadows:
        - Ownable.owner() (GainToken.sol#259-261) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Context._msgData() (GainToken.sol#233-236) is never used and should be removed
GainToken._burn(address,uint256) (GainToken.sol#449-457) is never used and should be removed
GainToken._mint(address,uint256) (GainToken.sol#442-447) is never used and should be removed
SafeMath.div(uint256,uint256) (GainToken.sol#88-90) is never used and should be removed
SafeMath.div(uint256,uint256,string) (GainToken.sol#103-114) is never used and should be removed
SafeMath.mod(uint256,uint256) (GainToken.sol#127-129) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (GainToken.sol#142-149) is never used and should be removed
SafeMath.mul(uint256,uint256) (GainToken.sol#63-75) is never used and should be removed
SafeMath.sub(uint256,uint256) (GainToken.sol#30-32) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Variable GainToken._decimalFactor (GainToken.sol#305) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (GainToken.sol#234)" inContext (GainToken.sol#224-237)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
GainToken.constructor() (GainToken.sol#307-313) uses literals with too many digits:
        - _totalSupply = 2000000 * _decimalFactor (GainToken.sol#312)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (GainToken.sol#275-277)
increaseAllowance(address,uint256) should be declared external:
        - GainToken.increaseAllowance(address,uint256) (GainToken.sol#381-391)
decreaseAllowance(address,uint256) should be declared external:
        - GainToken.decreaseAllowance(address,uint256) (GainToken.sol#393-406)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:GainToken.sol analyzed (5 contracts with 75 detectors), 18 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> GainToken.sol

```
INFO:Detectors:
Staking (Staking.sol#460-614) contract sets array length with a user-controlled value:
        - stakeholders.pop() (Staking.sol#566)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#array-length-assignment
INFO:Detectors:
GainToken.allowance(address,address).owner (Staking.sol#350) shadows:
        - Ownable.owner() (Staking.sol#259-261) (function)
GainToken._approve(address,address,uint256).owner (Staking.sol#430) shadows:
        - Ownable.owner() (Staking.sol#259-261) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Staking.removeStake(uint256) (Staking.sol#518-544) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)((stakes[_msgSender()].stakeTime + 7) <= block.timestamp,You have to stake for minimum 7 seconds.) (Staking
.sol#523-526)
        - require(bool,string)(stakes[_msgSender()].amount == _numberOfTokens.mul(tokenPrice),You have to unstake all your tokens) (Staki
ng.sol#527-530)
Staking.calculateReward(address) (Staking.sol#576-590) uses timestamp for comparisons
        Dangerous comparisons:
        - (stakes[_stakeholder].stakeTime + 7) <= block.timestamp (Staking.sol#582)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Context._msgData() (Staking.sol#233-236) is never used and should be removed
SafeMath.mod(uint256,uint256) (Staking.sol#127-129) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (Staking.sol#142-149) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Variable GainToken._decimalFactor (Staking.sol#304) is not in mixedCase
Struct Staking.stakeHolder (Staking.sol#467-470) is not in CapWords
Parameter Staking.createStake(uint256)._numberOfTokens (Staking.sol#485) is not in mixedCase
Parameter Staking.removeStake(uint256)._numberOfTokens (Staking.sol#518) is not in mixedCase
Parameter Staking.stakeOf(address)._stakeholder (Staking.sol#547) is not in mixedCase
Variable Staking.GAIN (Staking.sol#465) is not in mixedCase
Variable Staking.APY (Staking.sol#473) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (Staking.sol#234)" inContext (Staking.sol#224-237)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
Reentrancy in Staking.removeStake(uint256) (Staking.sol#518-544):
        External calls:
        - _msgSender().transfer(totalAmount) (Staking.sol#536)
        State variables written after the call(s):
        - _burn(_msgSender(),_numberOfTokens.mul(GAIN.decimalFactor())) (Staking.sol#542)
                - _balances[account] = _balances[account].sub(amount,ERC2020: burn amount exceeds balance) (Staking.sol#450-453)
        - removeStakeholder(_msgSender()) (Staking.sol#541)
                - stakeholders[s] = stakeholders[stakeholders.length - 1] (Staking.sol#565)
                - stakeholders.pop() (Staking.sol#566)
        - stakes[_msgSender()] = stakeHolder(0,0) (Staking.sol#540)
        - totalTokenStaked = totalTokenStaked.sub(_numberOfTokens.mul(GAIN.decimalFactor())) (Staking.sol#537-539)
        Event emitted after the call(s):
        - Transfer(account,address(0),amount) (Staking.sol#454)
                - _burn(_msgSender(),_numberOfTokens.mul(GAIN.decimalFactor())) (Staking.sol#542)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4
INFO:Detectors:
Variable Staking.stakeOf(address)._stakeholder (Staking.sol#547) is too similar to Staking.stakeholders (Staking.sol#463)
Variable Staking.removeStakeholder(address)._stakeholder (Staking.sol#562) is too similar to Staking.stakeholders (Staking.sol#463)
Variable Staking.rewardOf(address)._stakeholder (Staking.sol#571) is too similar to Staking.stakeholders (Staking.sol#463)
Variable Staking.viewReward(address)._stakeholder (Staking.sol#592) is too similar to Staking.stakeholders (Staking.sol#463)
Variable Staking.calculateReward(address)._stakeholder (Staking.sol#576) is too similar to Staking.stakeholders (Staking.sol#463)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
GainToken.constructor() (Staking.sol#306-312) uses literals with too many digits:
        - _totalSupply = 2000000 * _decimalFactor (Staking.sol#311)
Staking.slitherConstructorVariables() (Staking.sol#460-614) uses literals with too many digits:
        - tokenPrice = 100000000000000000 (Staking.sol#471)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
```

```
        - tokenPrice = 1000000000000000 (Staking.sol#471)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
Staking.APY (Staking.sol#473) should be constant
Staking.tokenPrice (Staking.sol#471) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (Staking.sol#275-277)
increaseAllowance(address,uint256) should be declared external:
        - GainToken.increaseAllowance(address,uint256) (Staking.sol#380-390)
decreaseAllowance(address,uint256) should be declared external:
        - GainToken.decreaseAllowance(address,uint256) (Staking.sol#392-405)
createStake(uint256) should be declared external:
        - Staking.createStake(uint256) (Staking.sol#485-515)
removeStake(uint256) should be declared external:
        - Staking.removeStake(uint256) (Staking.sol#518-544)
stakeOf(address) should be declared external:
        - Staking.stakeOf(address) (Staking.sol#547-549)
rewardOf(address) should be declared external:
        - Staking.rewardOf(address) (Staking.sol#571-573)
viewReward(address) should be declared external:
        - Staking.viewReward(address) (Staking.sol#592-597)
distributeRewards() should be declared external:
        - Staking.distributeRewards() (Staking.sol#600-606)
withdrawReward() should be declared external:
        - Staking.withdrawReward() (Staking.sol#609-613)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:Staking.sol analyzed (6 contracts with 75 detectors), 41 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
root@server:/chetan/gaza/mycontracts#
```

SOLIDITY STATIC ANALYSIS

Selfdestruct:

INTERNAL ERROR in module Selfdestruct: Cannot convert undefined or null to object
Pos: not available

Gas & Economy

This on local calls:

INTERNAL ERROR in module This on local calls: Cannot convert undefined or null to object
Pos: not available

Delete dynamic array:

INTERNAL ERROR in module Delete dynamic array: Cannot convert undefined or null to object
Pos: not available

For loop over dynamic array:

INTERNAL ERROR in module For loop over dynamic array: Cannot convert undefined or null to object
Pos: not available

Ether transfer in loop:

INTERNAL ERROR in module Ether transfer in loop: Cannot convert undefined or null to object
Pos: not available

ERC

ERC20:

INTERNAL ERROR in module ERC20: Cannot convert undefined or null to object
Pos: not available

Miscellaneous

Constant/View/Pure functions:

INTERNAL ERROR in module Constant/View/Pure functions: Cannot convert undefined or null to object
Pos: not available

Similar variable names:

INTERNAL ERROR in module Similar variable names: Cannot convert undefined or null to object
Pos: not available

No return:

INTERNAL ERROR in module No return: Cannot convert undefined or null to object
Pos: not available

Guard conditions:

INTERNAL ERROR in module Guard conditions: Cannot convert undefined or null to object
Pos: not available

String length:

INTERNAL ERROR in module String length: Cannot convert undefined or null to object
Pos: not available

Staking.sol

SOLIDITY STATIC ANALYSIS

last results for:
contracts/Staking.sol

Security

Transaction origin:

INTERNAL ERROR in module Transaction origin: Cannot convert undefined or null to object
Pos: not available

Check-effects-interaction:

INTERNAL ERROR in module Check-effects-interaction: Cannot convert undefined or null to object
Pos: not available

Inline assembly:

INTERNAL ERROR in module Inline assembly: Cannot convert undefined or null to object
Pos: not available

Block timestamp:

INTERNAL ERROR in module Block timestamp: Cannot convert undefined or null to object
Pos: not available

Low level calls:

INTERNAL ERROR in module Low level calls: Cannot convert undefined or null to object
Pos: not available

Selfdestruct:

INTERNAL ERROR in module Selfdestruct: Cannot convert undefined or null to object
Pos: not available

Gas & Economy

This on local calls:

INTERNAL ERROR in module This on local calls: Cannot convert undefined or null to object
Pos: not available

Delete dynamic array:

INTERNAL ERROR in module Delete dynamic array: Cannot convert undefined or null to object
Pos: not available

For loop over dynamic array:

INTERNAL ERROR in module For loop over dynamic array: Cannot convert undefined or null to object
Pos: not available

Ether transfer in loop:

INTERNAL ERROR in module Ether transfer in loop: Cannot convert undefined or null to object
Pos: not available

ERC

ERC20:

INTERNAL ERROR in module ERC20: Cannot convert undefined or null to object
Pos: not available

Miscellaneous

Constant/View/Pure functions:

INTERNAL ERROR in module Constant/View/Pure functions: Cannot convert undefined or null to object
Pos: not available

Similar variable names:

INTERNAL ERROR in module Similar variable names: Cannot convert undefined or null to object
Pos: not available

No return:

INTERNAL ERROR in module No return: Cannot convert undefined or null to object
Pos: not available

Guard conditions:

INTERNAL ERROR in module Guard conditions: Cannot convert undefined or null to object
Pos: not available

String length:

INTERNAL ERROR in module String length: Cannot convert undefined or null to object
Pos: not available

SOLHINT LINTER

GainToken.sol

```
contracts/GainToken.sol:1:1: Error: Compiler version 0.5.17 does not satisfy the r semver requirement

contracts/GainToken.sol:227:28: Error: Code contains empty blocks
```

Staking.sol

```
contracts/Staking.sol:1:1: Error: Compiler version 0.5.17 does not satisfy the r semver requirement

contracts/Staking.sol:227:28: Error: Code contains empty blocks

contracts/Staking.sol:465:23: Error: Variable name must be in mixedCase

contracts/Staking.sol:467:5: Error: Contract name must be in CamelCase

contracts/Staking.sol:473:20: Error: Variable name must be in mixedCase

contracts/Staking.sol:513:57: Error: Avoid to make time-based decisions in your business logic

contracts/Staking.sol:524:61: Error: Avoid to make time-based decisions in your business logic

contracts/Staking.sol:537:9: Error: Possible reentrancy vulnerabilities. Avoid state changes after transfer.

contracts/Staking.sol:540:9: Error: Possible reentrancy vulnerabilities. Avoid state changes after transfer.

contracts/Staking.sol:582:61: Error: Avoid to make time-based decisions in your business logic
```

Disclaimer

Quillhash audit is not a security warranty, investment advice, or an endorsement of the GainPool platform. This audit does not provide a security or correctness guarantee of the audited smart contracts. The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the GainPool Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

Closing Summary

In this report, we have considered the security of the GainPool platform. We performed our audit according to the procedure described above.

The audit showed some low and some informational level severity issues. The majority of the concerns addressed above have been acknowledged, implemented, and verified.

QuillAudits

Canada, India, Singapore and United Kingdom

audits.quillhash.com

audits@quillhash.com