HashEx

# GameFactory

## smart contracts audit report

Prepared for:

hifigamingsociety.org

Authors: HashEx audit team

August 2021

# Contents

# Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice at the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the disclaimer below – please make sure to read it in full.

DISCLAIMER: By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and HashEx and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers, and other representatives) (HashEx) owe no duty of care towards you or any other person, nor does HashEx make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties, or other terms of any kind except as set out in this disclaimer, and HashEx hereby excludes all representations, warranties, conditions, and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, HashEx hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against HashEx, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of the use of this report, and any reliance on this report.

The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.

# Introduction

HashEx was commissioned by the HiFi Gaming Society to perform an audit of their smart contracts. The audit was conducted between July 02 and July 20, 2021.

The code is located in the @HiFiDeFi-Finance/Game-Factory-Contract GitHub repository. The audited contracts are deployed to BSC testnet at address 0xD0c1CDf47B7aaA0b203922ad4AfBf46f0C6A65b2.
Documentation can be found in the litepaper [1].

The purpose of this audit was to achieve the following:

- Identify potential security issues with smart contracts.
- Formally check the logic behind given smart contracts.

Information in this report should be used to understand the risk exposure of smart contracts, and as a guide to improving the security posture of smart contracts by remediating the issues that were identified.

**Update**: HiFi Gaming Society has responded to this report. Individual responses were added below each item in the section. The updated code is located in the @HiFiDeFi-Finance/Game-Factory-Contract GitHub repository at the 9c549ddf commit.

Initial contract is deployed at: 0xD0c1CDf47B7aaA0b203922ad4AfBf46f0C6A65b2

**Update 2:** HiFi Gaming Society has responded to this report. Individual responses were added below each item in the section. All critical and all High severity issues were fixed. The updated code is deployed at: 0x1cFBe679F12E93b948A5fDAB0f5f52a56537dEa7. The same code is deployed to the BSC network at address 0x4fE92E89bf8500871f990a54CCd4Eb5F06308Cc5.

# Contracts overview

### GameFactory

Locking tokens, withdrawing, rewards distribution.
A user locks their tokens choosing the type of lock. The owner calls the method batchAddRewardCandidates assigning amounts of reward to users. Then the user can call unfreeze() to unfreeze their tokens to claim and now the user can claim their rewards.

# Found issues

| ID | Title | Severity | Status |
|---|---|---|---|
| 01 | GameFactory: A user can multiply their rewards unlimited times | Critical | Fixed |
| 02 | GameFactory: Owner’s ability to withdraw staked tokens | High | Fixed |
| 03 | GameFactory: Input data not checked | High | Fixed |
| 04 | GameFactory: No checks on reward amounts set by the owner | High | Fixed |
| 05 | GameFactory: Rewriting significant data | High | Fixed |
| 06 | GameFactory: Possibility of the token being changed | High | Fixed |
| 07 | GameFactory: Excess Owner’s powers | High | Fixed |
| 08 | GameFactory: totalStaked is not updated | Medium | Fixed |
| 09 | GameFactory: Unnecessary payable keyword | Medium | Fixed |
| 10 | GameFactory: Complicated calculation | Medium | Fixed |
| 11 | GameFactory: Excess Owner’s powers | Medium | Fixed |
| 12 | GameFactory: Solidity range is too wide | Low | Fixed |
| 13 | GameFactory: No explicit visibility | Low | Fixed |
| 14 | GameFactory: Indexed keyword is missing | Low | Fixed |

| | | | |
|---|---|---|---|
| 15 | GameFactory: Hardcoded payment token address | Low | Fixed |
| 16 | GameFactory: Using experimental features of the compiler | Low | Fixed |
| 17 | GameFactory: Implementation of views of public variables | Low | Informed |
| 18 | GameFactory: extra contracts are defined | Low | Fixed |
| 19 | GameFactory: an extra check | Low | Fixed |
| 20 | GameFactory: possible arrays’ lengths mismatch | Low | Fixed |
| 21 | GameFactory: an extra check | Low | Fixed |
| 22 | GameFactory: an extra check | Low | Fixed |
| 23 | GameFactory: granularity isn’t taken into account | Low | Fixed |
| 24 | GameFactory: redundant SafeMath usage | Low | Informed |
| 25 | GameFactory: sending burned tokens | Low | Fixed |
| 26 | GameFactory: General recommendations | Low | Informed |
| 27 | GameFactory: Insufficient balance (update 2) | Low | Informed |
| 28 | GameFactory: Redundant calculations (update 2) | Low | Informed |

| #01 | GameFactory: A user can multiply their rewards unlimited times | Critical |
|---|---|---|

`freeze()` function multiplies `thawingCandidates[msg.sender].approvedAmount` amount of HI-FI tokens, see L2246-2247. In a scenario, when a user calls `freeze()`, their `thawingCandidates.approvedAmount` is added to `stakeForEarningLists` and reset their `thawingCandidates.approvedAmount`. Then they can call `unfreeze()` and get the value back and call the function `freeze()` again, the user increases `stakeForEarningLists` again.

**Recommendation:** To fix the issue we recommend checking `thawingCandidates[msg.sender].endTime` is less than `block.timestamp` at least. It's also recommended to delete L318 at all to fix the issue.

**Update:** during the revision 78676e0 was added checking if `thawingCandidates[msg.sender].endTime` is less than `block.timestamp`.
**Update 2:** the issue was fixed. The line was deleted.

| #02 | GameFactory: Owner's ability to withdraw staked tokens | High |
|---|---|---|

`onlyOwner recoverTokenByAdmin()` function transfers the contract's balance to an arbitrary address.

**Recommendation:** Do not leave such a possibility since it makes the contract strongly dependent on the admin's account. Or make it possible to withdraw the amount of fees only.

**Update:** the issue was fixed. The function was removed. A new function was added, the one that withdraws the admin's commission only.

| #03 | GameFactory: input data not checked | High |
|---|---|---|

Set functions have no safety guards for updated parameters.

The general recommendation is to add requirements to check set data.

**Update:** in the refactored contract periods and prices are only checked if they are more than 0. We recommend checking also whether they are less than some appropriate values.

**Update 2:** the issue was fixed by adding limits of the input parameters.

| #04 | GameFactory: No checks on reward amounts set by the owner | High |
|---|---|---|

`onlyOwner batchAddRewardCandidates()` sets `approveAmounts` of tokens (user's rewards) but doesn't check whether the tokens were sent to the contract (L2479, L2480, L2484). This can lead to discrepancies when a reward is set to a user, but the user can't withdraw it because of the lack of tokens in the contract.

**Update:** the refactored code of the function could be improved by running the for-loop just once and checking the contract's balance then. The checking doesn't make the call safer since it tests whether the balance is higher than the sum of input `approveAmounts`, but not the total amounts of tokens that users are able to withdraw.

**Update 2:** the issue was fixed since the appropriate checking was implemented.

| #05 | GameFactory: Rewriting significant data | High |
|---|---|---|

`cancel()` function rewrites `rewardCandidates[msg.sender].approvedAmount` instead of adding value.

**Update:** the issue was fixed.

| #06 | GameFactory: possibility of the token being changed | High |
|---|---|---|

Since the Owner is able to change the address of the payment token (L2381) users can get back other tokens. Or it can be set to `address(0)` and break down the contract. We do not recommend changing a token's address after it was initially set.

**Update:** the issue was fixed by removing this possibility.

| #07 | GameFactory: Excess Owner's powers | High |
|---|---|---|

L2333: The Owner is able to un-whitelist a user before they withdraw their tokens.

**Recommendation:** do not check whether msg.sender is in White-list during withdrawal.

**Update:** the issue was fixed by removing checking whether the user is in White-list.

### #08 GameFactory: totalStaked is not updated Medium

`withdrawStakedToken()` L2353 doesn't update `totalStaked` variables. It causes confusing statistics of staked tokens. `StakedWithdrawn` represents a confusing amount without commission.

**Recommendation:** set appropriate values each time when the staked amount is changed.

**Update:** the issue was fixed.

### #09 GameFactory: Unnecessary payable keyword Medium

`withdrawStakedToken(), withdrawBoost(), stakeForBoost(), stakeToken()` functions are payable for no reason. It may cause accidental sending of funds to the contracts.

**Recommendation:** We recommend removing the payable keyword since the contract doesn't receive money.

**Update:** the issue was fixed by removing the `payable` keyword.

### #10 GameFactory: Complicated calculation Medium

At L2355 the amount of Fee is multiplied by the value of `stakeType`. It's not obvious what it's needed for. If such behavior is necessary, it should at least be described.

**Recommendation**: define and set different fees depending on the `stakeType`.

**Update:** the issue was fixed. Different fees were defined.

### #11 GameFactory: Excess Owner's powers Medium

The Owner is able to change `_goldItemPrice`, `_silverItemPrice` and `_bronzeItemPrice` values. Changing that might be a cause of a wrong calculation of the amount to withdraw at L2335.

**Recommendation:** do not make the Owner able to change the values or store userBoostItemBalance as an absolute value of staked amount.

**Update:** Fixed. Setters of the variable were removed.

### #12 GameFactory: solidity range is too wide Low

`pragma solidity` range of (`>=0.6.0 <0.8.3`) is too wide. We recommend fixing the compiler version to a specific one. We recommend not mixing the pre and post 0.8.0 versions.

**Update:** the issue was fixed by defining compiler version 0.8.0 as required.

---

### #13 GameFactory: no explicit visibility — Low

L2048-2104: no explicit visibility for

```
bool isBurnable = false;
uint256 _goldItemPrice;
uint256 _silverItemPrice;
uint256 _bronzeItemPrice;
GameStatistic platformStatistic;
uint256 totalWithdrawnTokenByAdmin;
uint256 totalWithdrawnETHByAdmin.
```

**Update:** the issue was fixed except for L88 where the visibility still is not set.

### #14 GameFactory: indexed keyword is missing — Low

Event parameters lack indexed parameters for addresses (L2083-L2095).

**Recommendation:** We recommend adding an `indexed` keyword for the address parameters to make them searchable.

**Update:** the issue was fixed by adding the keyword.

### #15 GameFactory: hardcoded payment token address — Low

L2107: It's better to set the address via constructor as a parameter.

**Update:** the issue was fixed by setting the address as a constructor parameter.

### #16 GameFactory: using experimental features of the compiler — Low

We recommend not using experimental features of Solidity compiler, see pragma experimental `ABIEncoderV2` in L9 of GameFactory.

**Update:** the issue was fixed by removing the feature.

### #17 GameFactory: implementation of views of public variables — Low

No need to implement getters for public variables since that's already implemented implicitly.

### #18 GameFactory: extra contracts are defined — Low

ERC721 contracts are not used.

**Recommendation:** we recommend removing unused contracts.

**Update:** Fixed. The import of the contract was removed.

### #19 GameFactory: an extra check — Low

L2267, L2304: No need to check allowance and balance before `transferFrom()` since it must be checked inside of `transferFrom()` method.

**Update:** the extra checking has been removed.

### #20 GameFactory: possible arrays' lengths mismatch — Low

`batchAddRewardCandidates()` contains possible arrays' lengths mismatch. We recommend adding require() which checks the lengths of the arrays.

**Update:** the arrays' lengths have been checked.

### #21 GameFactory: an extra check — Low

L2301: no need to check whether amount > 0 since the amount is set up above

**Update:** the issue was fixed by removing the extra check.

### #22 GameFactory: an extra check — Low

L2359, L2366: no need to check whether stakeForEarningLists[msg.sender] > 0 since there is a check below.

**Update:** the issue was fixed by removing the extra check.

### #23 GameFactory: granularity isn't taken into account (update 1) — Low

L102: `_baseStakeAmountForPlay` is set as `100` in the constructor. Which means `0.0000000000000001` Tokens.

**Update:** the issue was fixed by multiplying such values by $10^{18}$.

### #24 GameFactory: redundant SafeMath usage (update 1) Low

Compiler version 0.8.0 is defined. Therefore it's not necessary to use SafeMath anymore.

### #25 GameFactory: sending burned tokens Low

Burned tokens are sent to `address(0x1)`. We recommend to send that to the dead address ( `0x000000000000000000000000000000000000dEaD` ).

**Update:** the issue was fixed. The address where burned tokens were sent was changed.

### #26 General recommendations Low

We recommend renaming the `function stakeToken` to `stakeTokens`.

**Update:** the issue was fixed. The function was renamed to `stakeTokens`.

We recommend using the fixed pragma version and naming the variables according to Solidity code style.

### #27 Insufficient balance (update 2) Low

L592-593: the `amount` should be less or equal to the `availableCommissionBalance` and `contractTokenBalance.`

### #28 Redundant calculations (update 2) Low

L595-597: It's better to assign the burnable amount to a variable avoiding redundant calculations, and saving the gas.

The audited code provided with tests, coverage (lines) is about 65%.

Contracts are extremely dependent on the owner's account.

# Conclusion

1 critical and 4 high severity issues were found.

The contracts are highly dependent on the owner's account. Users stake tokens for play and for earning. The owner distributes rewards, but the GameFactory contract has no constraints that the users who staked more will get bigger rewards, it's based on the owner's decision. Users of the project have to trust the owner and that the owner's account is properly secured.

Audit includes recommendations on the code improving and preventing potential attacks.
**Update**: HiFi Gaming Society has responded to this report. Individual responses were added below each item in the section. The updated code is located in the @HiFiDeFi-Finance/Game-Factory-Contract GitHub repository at the 9c549ddf commit

Initial contract is deployed at: 0xD0c1CDf47B7aaA0b203922ad4AfBf46f0C6A65b2

**Update 2:** HiFi Gaming Society has responded to this report. Individual responses were added below each item in the section. The critical and all High severity issues were fixed.

The updated code is deployed to the BSC network at address 0x4fE92E89bf8500871f990a54CCd4Eb5F06308Cc5.

# References


1. Project litepaper: https://hifigamingsociety.org/litepaper.html

# Appendix A. Issues' severity classification

We consider an issue critical, if it may cause unlimited losses or breaks the workflow of the contract and could be easily triggered.

High severity issues may lead to limited losses or break interaction with users or other contracts under very specific conditions.

Medium severity issues do not cause the full loss of functionality but break the contract logic.

Low severity issues are typically nonoptimal code, unused variables, errors in messages. Usually, these issues do not need immediate reactions.

# Appendix B. List of examined issue types

Business logic overview

Functionality checks

Following best practices

Access control and authorization

Reentrancy attacks

Front-run attacks

DoS with (unexpected) revert

DoS with block gas limit

Transaction-ordering dependence

ERC/BEP and other standards violation

Unchecked math

Implicit visibility levels

Excessive gas usage

Timestamp dependence

Forcibly sending ether to a contract

Weak sources of randomness

Shadowing state variables

Usage of deprecated code

# Appendix C. Listing of Slither-Analyzer

```
INFO:Detectors:
GameFactory.recoverBNBByAdmin(address,uint256)
(contracts/GameFactory.sol#609-619) sends eth to arbitrary user
      Dangerous calls:
      - (success,None) = sendTo.call{value: amount}()
(contracts/GameFactory.sol#613)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-
send-ether-to-arbitrary-destinations
INFO:Detectors:
GameFactory.recoverTokenByAdmin(address,uint256)
(contracts/GameFactory.sol#626-641) ignores return value by
IERC20(paymentTokenAddress).transfer(to_wallet,realAmountTokens)
(contracts/GameFactory.sol#635-640)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-trans
fer
INFO:Detectors:
Reentrancy in GameFactory.claimReward() (contracts/GameFactory.sol#282-306):
      External calls:
      -
require(bool)(IERC20(paymentTokenAddress).transfer(msg.sender,thawingCandidat
es[msg.sender].approvedAmount.sub(commissionFee)))
(contracts/GameFactory.sol#292)
      State variables written after the call(s):
      - _resetThawing() (contracts/GameFactory.sol#303)
            - thawingCandidates[msg.sender].approvedAmount = 0
(contracts/GameFactory.sol#239)
            - thawingCandidates[msg.sender].startTime = 0
(contracts/GameFactory.sol#240)
            - thawingCandidates[msg.sender].endTime = 0
(contracts/GameFactory.sol#241)
            - thawingCandidates[msg.sender].status = false
(contracts/GameFactory.sol#242)
Reentrancy in GameFactory.stakeToken(uint256,uint256)
(contracts/GameFactory.sol#333-350):
      External calls:
      -
require(bool)(IERC20(paymentTokenAddress).transferFrom(msg.sender,address(thi
s),amount)) (contracts/GameFactory.sol#337)
```

```
      -
require(bool)(IERC20(paymentTokenAddress).transferFrom(msg.sender,address(thi
s),amount)) (contracts/GameFactory.sol#344)
      State variables written after the call(s):
      - platformStatistic.totalStakedForEarn =
platformStatistic.totalStakedForEarn.add(amount)
(contracts/GameFactory.sol#346)
Reentrancy in GameFactory.withdrawBoost(uint256,uint16)
(contracts/GameFactory.sol#390-409):
      External calls:
      -
require(bool)(IERC20(paymentTokenAddress).transfer(msg.sender,amount.mul(coun
t))) (contracts/GameFactory.sol#403)
      State variables written after the call(s):
      - userBoostItemBalance[msg.sender][itemType] =
userBoostItemBalance[msg.sender][itemType].sub(count)
(contracts/GameFactory.sol#405)
Reentrancy in GameFactory.withdrawStakedToken(uint256,uint256)
(contracts/GameFactory.sol#424-445):
      External calls:
      -
require(bool)(IERC20(paymentTokenAddress).transfer(msg.sender,amount.sub(comm
issionFee))) (contracts/GameFactory.sol#430)
      State variables written after the call(s):
      - stakeForPlayLists[msg.sender] =
stakeForPlayLists[msg.sender].sub(amount) (contracts/GameFactory.sol#431)
Reentrancy in GameFactory.withdrawStakedToken(uint256,uint256)
(contracts/GameFactory.sol#424-445):
      External calls:
      -
require(bool)(IERC20(paymentTokenAddress).transfer(msg.sender,amount.sub(comm
issionFee))) (contracts/GameFactory.sol#430)
      -
require(bool)(IERC20(paymentTokenAddress).transfer(msg.sender,amount.sub(comm
issionFee))) (contracts/GameFactory.sol#438)
      State variables written after the call(s):
      - platformStatistic.totalWithdrawn =
platformStatistic.totalWithdrawn.add(amount.sub(commissionFee))
(contracts/GameFactory.sol#440)
      - platformStatistic.totalCommission =
platformStatistic.totalCommission.add(commissionFee)
(contracts/GameFactory.sol#441)
```

```
      - stakeForEarningLists[msg.sender] =
stakeForEarningLists[msg.sender].sub(amount) (contracts/GameFactory.sol#439)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vuln
erabilities-1
INFO:Detectors:
GameFactory.constructor(address,address,address)._paymetTokenAddress
(contracts/GameFactory.sol#100) lacks a zero-check on :
            - paymentTokenAddress = _paymetTokenAddress
(contracts/GameFactory.sol#101)
GameFactory.recoverBNBByAdmin(address,uint256).sendTo
(contracts/GameFactory.sol#609) lacks a zero-check on :
            - (success,None) = sendTo.call{value: amount}()
(contracts/GameFactory.sol#613)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-ad
dress-validation
INFO:Detectors:
Reentrancy in GameFactory.claimReward() (contracts/GameFactory.sol#282-306):
      External calls:
      -
require(bool)(IERC20(paymentTokenAddress).transfer(msg.sender,thawingCandidat
es[msg.sender].approvedAmount.sub(commissionFee)))
(contracts/GameFactory.sol#292)
      State variables written after the call(s):
      - platformStatistic.totalRewarded =
platformStatistic.totalRewarded.add(thawingCandidates[msg.sender].approvedAmo
unt) (contracts/GameFactory.sol#295)
      - platformStatistic.totalCommission =
platformStatistic.totalCommission.add(commissionFee)
(contracts/GameFactory.sol#296)
Reentrancy in GameFactory.recoverBNBByAdmin(address,uint256)
(contracts/GameFactory.sol#609-619):
      External calls:
      - (success,None) = sendTo.call{value: amount}()
(contracts/GameFactory.sol#613)
      State variables written after the call(s):
      - totalWithdrawnETHByAdmin += amount (contracts/GameFactory.sol#615)
Reentrancy in GameFactory.recoverTokenByAdmin(address,uint256)
(contracts/GameFactory.sol#626-641):
      External calls:
      - IERC20(paymentTokenAddress).transfer(to_wallet,realAmountTokens)
(contracts/GameFactory.sol#635-640)
```

```
      State variables written after the call(s):
      - totalWithdrawnTokenByAdmin =
totalWithdrawnTokenByAdmin.add(realAmountTokens)
(contracts/GameFactory.sol#636)
Reentrancy in GameFactory.stakeForBoost(uint256)
(contracts/GameFactory.sol#356-383):
      External calls:
      -
require(bool,string)(IERC20(paymentTokenAddress).transferFrom(msg.sender,addr
ess(this),amount.sub(amount.mul(_burnFee).div(100))),Token Stake error)
(contracts/GameFactory.sol#371)
      - burn(msg.sender,amount.mul(_burnFee).div(100))
(contracts/GameFactory.sol#372)
            -
require(bool)(IERC20(paymentTokenAddress).transferFrom(from,address(0x1),amou
nt)) (contracts/GameFactory.sol#417)
      State variables written after the call(s):
      - platformStatistic.totalBurned =
platformStatistic.totalBurned.add(amount.mul(_burnFee).div(100))
(contracts/GameFactory.sol#373)
Reentrancy in GameFactory.stakeForBoost(uint256)
(contracts/GameFactory.sol#356-383):
      External calls:
      -
require(bool,string)(IERC20(paymentTokenAddress).transferFrom(msg.sender,addr
ess(this),amount.sub(amount.mul(_burnFee).div(100))),Token Stake error)
(contracts/GameFactory.sol#371)
      - burn(msg.sender,amount.mul(_burnFee).div(100))
(contracts/GameFactory.sol#372)
            -
require(bool)(IERC20(paymentTokenAddress).transferFrom(from,address(0x1),amou
nt)) (contracts/GameFactory.sol#417)
      -
require(bool,string)(IERC20(paymentTokenAddress).transferFrom(msg.sender,addr
ess(this),amount),Token Stake error) (contracts/GameFactory.sol#375)
      State variables written after the call(s):
      - platformStatistic.totalStakedForBoost =
platformStatistic.totalStakedForBoost.add(amount)
(contracts/GameFactory.sol#378)
      - userBoostItemBalance[msg.sender][itemType] =
userBoostItemBalance[msg.sender][itemType].add(1)
(contracts/GameFactory.sol#379)
```

```
Reentrancy in GameFactory.stakeToken(uint256,uint256)
(contracts/GameFactory.sol#333-350):
      External calls:
      -
require(bool)(IERC20(paymentTokenAddress).transferFrom(msg.sender,address(thi
s),amount)) (contracts/GameFactory.sol#337)
      State variables written after the call(s):
      - platformStatistic.totalStakedForPlay =
platformStatistic.totalStakedForPlay.add(amount)
(contracts/GameFactory.sol#339)
      - stakeForPlayLists[msg.sender] =
stakeForPlayLists[msg.sender].add(amount) (contracts/GameFactory.sol#338)
Reentrancy in GameFactory.stakeToken(uint256,uint256)
(contracts/GameFactory.sol#333-350):
      External calls:
      -
require(bool)(IERC20(paymentTokenAddress).transferFrom(msg.sender,address(thi
s),amount)) (contracts/GameFactory.sol#337)
      -
require(bool)(IERC20(paymentTokenAddress).transferFrom(msg.sender,address(thi
s),amount)) (contracts/GameFactory.sol#344)
      State variables written after the call(s):
      - stakeForEarningLists[msg.sender] =
stakeForEarningLists[msg.sender].add(amount) (contracts/GameFactory.sol#345)
Reentrancy in GameFactory.withdrawBoost(uint256,uint16)
(contracts/GameFactory.sol#390-409):
      External calls:
      -
require(bool)(IERC20(paymentTokenAddress).transfer(msg.sender,amount.mul(coun
t))) (contracts/GameFactory.sol#403)
      State variables written after the call(s):
      - platformStatistic.totalStakedForBoost =
platformStatistic.totalStakedForBoost.sub(amount.mul(count))
(contracts/GameFactory.sol#404)
Reentrancy in GameFactory.withdrawStakedToken(uint256,uint256)
(contracts/GameFactory.sol#424-445):
      External calls:
      -
require(bool)(IERC20(paymentTokenAddress).transfer(msg.sender,amount.sub(comm
issionFee))) (contracts/GameFactory.sol#430)
      State variables written after the call(s):
```

```
      - platformStatistic.totalWithdrawn =
platformStatistic.totalWithdrawn.add(amount.sub(commissionFee))
(contracts/GameFactory.sol#432)
      - platformStatistic.totalCommission =
platformStatistic.totalCommission.add(commissionFee)
(contracts/GameFactory.sol#433)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vuln
erabilities-2
INFO:Detectors:
Reentrancy in GameFactory.claimReward() (contracts/GameFactory.sol#282-306):
      External calls:
      -
require(bool)(IERC20(paymentTokenAddress).transfer(msg.sender,thawingCandidat
es[msg.sender].approvedAmount.sub(commissionFee)))
(contracts/GameFactory.sol#292)
      Event emitted after the call(s):
      -
ClaimedUserReward(msg.sender,thawingCandidates[msg.sender].approvedAmount.sub
(commissionFee),block.timestamp) (contracts/GameFactory.sol#299)
      - UpdatedStatisticData(platformStatistic)
(contracts/GameFactory.sol#300)
Reentrancy in GameFactory.recoverBNBByAdmin(address,uint256)
(contracts/GameFactory.sol#609-619):
      External calls:
      - (success,None) = sendTo.call{value: amount}()
(contracts/GameFactory.sol#613)
      Event emitted after the call(s):
      - Withdrawn(amount,sendTo) (contracts/GameFactory.sol#618)
Reentrancy in GameFactory.recoverTokenByAdmin(address,uint256)
(contracts/GameFactory.sol#626-641):
      External calls:
      - IERC20(paymentTokenAddress).transfer(to_wallet,realAmountTokens)
(contracts/GameFactory.sol#635-640)
      Event emitted after the call(s):
      - TokensWithdrawn(realAmountTokens,to_wallet)
(contracts/GameFactory.sol#637)
Reentrancy in GameFactory.stakeForBoost(uint256)
(contracts/GameFactory.sol#356-383):
      External calls:
      -
require(bool,string)(IERC20(paymentTokenAddress).transferFrom(msg.sender,addr
```

```
ess(this),amount.sub(amount.mul(_burnFee).div(100))),Token Stake error)
(contracts/GameFactory.sol#371)
      - burn(msg.sender,amount.mul(_burnFee).div(100))
(contracts/GameFactory.sol#372)
            -
require(bool)(IERC20(paymentTokenAddress).transferFrom(from,address(0x1),amou
nt)) (contracts/GameFactory.sol#417)
      -
require(bool,string)(IERC20(paymentTokenAddress).transferFrom(msg.sender,addr
ess(this),amount),Token Stake error) (contracts/GameFactory.sol#375)
      Event emitted after the call(s):
      - PurchasedBoostItem(msg.sender,itemType)
(contracts/GameFactory.sol#381)
      - UpdatedStatisticData(platformStatistic)
(contracts/GameFactory.sol#382)
Reentrancy in GameFactory.stakeToken(uint256,uint256)
(contracts/GameFactory.sol#333-350):
      External calls:
      -
require(bool)(IERC20(paymentTokenAddress).transferFrom(msg.sender,address(thi
s),amount)) (contracts/GameFactory.sol#337)
      Event emitted after the call(s):
      - TokensDeposited(amount,stakeType,msg.sender)
(contracts/GameFactory.sol#340)
Reentrancy in GameFactory.stakeToken(uint256,uint256)
(contracts/GameFactory.sol#333-350):
      External calls:
      -
require(bool)(IERC20(paymentTokenAddress).transferFrom(msg.sender,address(thi
s),amount)) (contracts/GameFactory.sol#337)
      -
require(bool)(IERC20(paymentTokenAddress).transferFrom(msg.sender,address(thi
s),amount)) (contracts/GameFactory.sol#344)
      Event emitted after the call(s):
      - TokensDeposited(amount,stakeType,msg.sender)
(contracts/GameFactory.sol#347)
      - UpdatedStatisticData(platformStatistic)
(contracts/GameFactory.sol#349)
Reentrancy in GameFactory.withdrawBoost(uint256,uint16)
(contracts/GameFactory.sol#390-409):
      External calls:
```

```
      -
require(bool)(IERC20(paymentTokenAddress).transfer(msg.sender,amount.mul(coun
t))) (contracts/GameFactory.sol#403)
      Event emitted after the call(s):
      - SoldBoostItem(msg.sender,itemType,count)
(contracts/GameFactory.sol#406)
      - UpdatedStatisticData(platformStatistic)
(contracts/GameFactory.sol#407)
Reentrancy in GameFactory.withdrawStakedToken(uint256,uint256)
(contracts/GameFactory.sol#424-445):
      External calls:
      -
require(bool)(IERC20(paymentTokenAddress).transfer(msg.sender,amount.sub(comm
issionFee))) (contracts/GameFactory.sol#430)
      Event emitted after the call(s):
      - StakedWithdrawn(amount,stakeType,msg.sender)
(contracts/GameFactory.sol#434)
Reentrancy in GameFactory.withdrawStakedToken(uint256,uint256)
(contracts/GameFactory.sol#424-445):
      External calls:
      -
require(bool)(IERC20(paymentTokenAddress).transfer(msg.sender,amount.sub(comm
issionFee))) (contracts/GameFactory.sol#430)
      -
require(bool)(IERC20(paymentTokenAddress).transfer(msg.sender,amount.sub(comm
issionFee))) (contracts/GameFactory.sol#438)
      Event emitted after the call(s):
      - StakedWithdrawn(amount,stakeType,msg.sender)
(contracts/GameFactory.sol#442)
      - UpdatedStatisticData(platformStatistic)
(contracts/GameFactory.sol#444)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vuln
erabilities-3
INFO:Detectors:
GameFactory.batchAddRewardCandidates(address[],uint256[])
(contracts/GameFactory.sol#577-595) uses timestamp for comparisons
      Dangerous comparisons:
      - isWhitelisted(msg.sender) && block.timestamp >
rewardCandidates[userAddresses[i]].lastRewardedTime.add(_addRewardCandidatePe
riod) (contracts/GameFactory.sol#582)
```

```
      - isWhitelisted(msg.sender) && block.timestamp >
rewardCandidates[userAddresses[i_scope_0]].lastRewardedTime.add(_addRewardCan
didatePeriod) (contracts/GameFactory.sol#588)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address)
(node_modules/@openzeppelin/contracts/utils/Address.sol#26-36) uses assembly
      - INLINE ASM
(node_modules/@openzeppelin/contracts/utils/Address.sol#32-34)
Address._verifyCallResult(bool,bytes,string)
(node_modules/@openzeppelin/contracts/utils/Address.sol#189-209) uses
assembly
      - INLINE ASM
(node_modules/@openzeppelin/contracts/utils/Address.sol#201-204)
console._sendLogPayload(bytes) (node_modules/hardhat/console.sol#7-14) uses
assembly
      - INLINE ASM (node_modules/hardhat/console.sol#10-13)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Different versions of Solidity is used:
      - Version used: ['0.8.0', '>=0.4.22<0.9.0', '^0.8.0']
      - ^0.8.0
(node_modules/@openzeppelin/contracts/access/AccessControl.sol#3)
      - ^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#3)
      - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#3)
      - ^0.8.0
(node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3)
      - ^0.8.0
(node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.s
ol#3)
      - ^0.8.0
(node_modules/@openzeppelin/contracts/token/ERC721/IERC721.sol#3)
      - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3)
      - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3)
      - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Strings.sol#3)
      - ^0.8.0
(node_modules/@openzeppelin/contracts/utils/introspection/ERC165.sol#3)
      - ^0.8.0
(node_modules/@openzeppelin/contracts/utils/introspection/IERC165.sol#3)
      - ^0.8.0
(node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#3)
```

```
      - 0.8.0 (contracts/GameFactory.sol#3)
      - 0.8.0 (contracts/Token.sol#2)
      - >=0.4.22<0.9.0 (node_modules/hardhat/console.sol#2)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts/access/AccessControl.sol#3)
necessitates a version too recent to be trusted. Consider deploying with
0.6.12/0.7.6
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts/access/Ownable.sol#3) necessitates a
version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#3) necessitates a
version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3) necessitates
a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.s
ol#3) necessitates a version too recent to be trusted. Consider deploying
with 0.6.12/0.7.6
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts/token/ERC721/IERC721.sol#3)
necessitates a version too recent to be trusted. Consider deploying with
0.6.12/0.7.6
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts/utils/Address.sol#3) necessitates a
version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts/utils/Context.sol#3) necessitates a
version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts/utils/Strings.sol#3) necessitates a
version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts/utils/introspection/ERC165.sol#3)
necessitates a version too recent to be trusted. Consider deploying with
0.6.12/0.7.6
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts/utils/introspection/IERC165.sol#3)
```

```
necessitates a version too recent to be trusted. Consider deploying with
0.6.12/0.7.6
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#3) necessitates
a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.0 (contracts/GameFactory.sol#3) necessitates a version too
recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.0 (contracts/Token.sol#2) necessitates a version too recent
to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version>=0.4.22<0.9.0 (node_modules/hardhat/console.sol#2) is too
complex
solc-0.8.0 is not recommended for deployment
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versi
ons-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256)
(node_modules/@openzeppelin/contracts/utils/Address.sol#54-59):
      - (success) = recipient.call{value: amount}()
(node_modules/@openzeppelin/contracts/utils/Address.sol#57)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string)
(node_modules/@openzeppelin/contracts/utils/Address.sol#122-133):
      - (success,returndata) = target.call{value: value}(data)
(node_modules/@openzeppelin/contracts/utils/Address.sol#131)
Low level call in Address.functionStaticCall(address,bytes,string)
(node_modules/@openzeppelin/contracts/utils/Address.sol#151-160):
      - (success,returndata) = target.staticcall(data)
(node_modules/@openzeppelin/contracts/utils/Address.sol#158)
Low level call in Address.functionDelegateCall(address,bytes,string)
(node_modules/@openzeppelin/contracts/utils/Address.sol#178-187):
      - (success,returndata) = target.delegatecall(data)
(node_modules/@openzeppelin/contracts/utils/Address.sol#185)
Low level call in GameFactory.recoverBNBByAdmin(address,uint256)
(contracts/GameFactory.sol#609-619):
      - (success,None) = sendTo.call{value: amount}()
(contracts/GameFactory.sol#613)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter GameFactory.enableWhitelist(bool)._enabled
(contracts/GameFactory.sol#125) is not in mixedCase
Parameter GameFactory.addToWhitelist(address)._newAddress
(contracts/GameFactory.sol#133) is not in mixedCase
```

```
Parameter GameFactory.removeFromWhitelist(address)._removedAddress
(contracts/GameFactory.sol#142) is not in mixedCase
Parameter GameFactory.isWhitelisted(address)._address
(contracts/GameFactory.sol#152) is not in mixedCase
Function GameFactory._getAddRewardCandidatePeriod()
(contracts/GameFactory.sol#203-205) is not in mixedCase
Function GameFactory._setBaseStakeAmountForPlay(uint256)
(contracts/GameFactory.sol#451-454) is not in mixedCase
Parameter
GameFactory._setBaseStakeAmountForPlay(uint256)._newBaseStakeAmountForPlay
(contracts/GameFactory.sol#451) is not in mixedCase
Function GameFactory._setBaseStakeAmountForEarn(uint256)
(contracts/GameFactory.sol#460-463) is not in mixedCase
Parameter
GameFactory._setBaseStakeAmountForEarn(uint256)._newBaseStakeAmountForEarn
(contracts/GameFactory.sol#460) is not in mixedCase
Function GameFactory._setBurFee(uint256) (contracts/GameFactory.sol#469-472)
is not in mixedCase
Parameter GameFactory._setBurFee(uint256)._newBurnFee
(contracts/GameFactory.sol#469) is not in mixedCase
Function GameFactory._setWithdrawFee(uint256)
(contracts/GameFactory.sol#478-481) is not in mixedCase
Parameter GameFactory._setWithdrawFee(uint256)._newWithdrawFee
(contracts/GameFactory.sol#478) is not in mixedCase
Function GameFactory._setThawingPeriod(uint256)
(contracts/GameFactory.sol#487-490) is not in mixedCase
Parameter GameFactory._setThawingPeriod(uint256)._newThawingPeriod
(contracts/GameFactory.sol#487) is not in mixedCase
Function GameFactory._setWithdrawLockingPeriod(uint256)
(contracts/GameFactory.sol#496-499) is not in mixedCase
Parameter
GameFactory._setWithdrawLockingPeriod(uint256)._newWithdrawLockingPeriod
(contracts/GameFactory.sol#496) is not in mixedCase
Function GameFactory._setMaximumAmountForBoostItem(uint256)
(contracts/GameFactory.sol#505-508) is not in mixedCase
Parameter
GameFactory._setMaximumAmountForBoostItem(uint256)._newMaximumAmountForBoostI
tem (contracts/GameFactory.sol#505) is not in mixedCase
Function GameFactory._setGoldItemPrice(uint256)
(contracts/GameFactory.sol#514-517) is not in mixedCase
Parameter GameFactory._setGoldItemPrice(uint256)._newGoldItemPrice
(contracts/GameFactory.sol#514) is not in mixedCase
```

```
Function GameFactory._setSilverItemPrice(uint256)
(contracts/GameFactory.sol#523-526) is not in mixedCase
Parameter GameFactory._setSilverItemPrice(uint256)._newSilverItemPrice
(contracts/GameFactory.sol#523) is not in mixedCase
Function GameFactory._setBronzeItemPrice(uint256)
(contracts/GameFactory.sol#532-535) is not in mixedCase
Parameter GameFactory._setBronzeItemPrice(uint256)._newBronzeItemPrice
(contracts/GameFactory.sol#532) is not in mixedCase
Function GameFactory._setAddRewardCandidatePeriod(uint256)
(contracts/GameFactory.sol#541-544) is not in mixedCase
Parameter
GameFactory._setAddRewardCandidatePeriod(uint256)._newAddRewardCandidatePerio
d (contracts/GameFactory.sol#541) is not in mixedCase
Parameter GameFactory.initWhitelist(address[])._whitelistees
(contracts/GameFactory.sol#550) is not in mixedCase
Parameter GameFactory.removeFromWhitelist(address[])._unwhitelistees
(contracts/GameFactory.sol#563) is not in mixedCase
Function GameFactory._setBurnStatus(bool) (contracts/GameFactory.sol#601-603)
is not in mixedCase
Parameter GameFactory._setBurnStatus(bool)._burnableStatus
(contracts/GameFactory.sol#601) is not in mixedCase
Parameter GameFactory.recoverTokenByAdmin(address,uint256).to_wallet
(contracts/GameFactory.sol#626) is not in mixedCase
Variable GameFactory._baseStakeAmountForPlay (contracts/GameFactory.sol#23)
is not in mixedCase
Variable GameFactory._baseStakeAmountForEarn (contracts/GameFactory.sol#24)
is not in mixedCase
Variable GameFactory._burnFee (contracts/GameFactory.sol#25) is not in
mixedCase
Variable GameFactory._baseUnitForWithdrawFee (contracts/GameFactory.sol#26)
is not in mixedCase
Variable GameFactory._thawingLockingPeriod (contracts/GameFactory.sol#27) is
not in mixedCase
Variable GameFactory._withdrawLockingPeriod (contracts/GameFactory.sol#28) is
not in mixedCase
Variable GameFactory._maximumAmountForBoostItem
(contracts/GameFactory.sol#29) is not in mixedCase
Variable GameFactory._addRewardCandidatePeriod (contracts/GameFactory.sol#30)
is not in mixedCase
Variable GameFactory._goldItemPrice (contracts/GameFactory.sol#32) is not in
mixedCase
Variable GameFactory._silverItemPrice (contracts/GameFactory.sol#33) is not
in mixedCase
```

```
Variable GameFactory._bronzeItemPrice (contracts/GameFactory.sol#34) is not
in mixedCase
Contract console (node_modules/hardhat/console.sol#4-1532) is not in CapWords
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-
solidity-naming-conventions
INFO:Detectors:
Variable GameFactory.constructor(address,address,address)._cfoAddress
(contracts/GameFactory.sol#100) is too similar to
GameFactory.constructor(address,address,address)._cmoAddress
(contracts/GameFactory.sol#100)
Variable GameFactory.constructor(address,address,address)._paymetTokenAddress
(contracts/GameFactory.sol#100) is too similar to
GameFactory.paymentTokenAddress (contracts/GameFactory.sol#20)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-
are-too-similar
INFO:Detectors:
HiFiToken.constructor() (contracts/Token.sol#7-9) uses literals with too many
digits:
      - _mint(msg.sender,100000000000000000000000000) (contracts/Token.sol#8)
console.slitherConstructorConstantVariables()
(node_modules/hardhat/console.sol#4-1532) uses literals with too many digits:
      - CONSOLE_ADDRESS = address(0x000000000000000000636F6e736F6c652e6c6f67)
(node_modules/hardhat/console.sol#5)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
grantRole(bytes32,address) should be declared external:
      - AccessControl.grantRole(bytes32,address)
(node_modules/@openzeppelin/contracts/access/AccessControl.sol#170-172)
revokeRole(bytes32,address) should be declared external:
      - AccessControl.revokeRole(bytes32,address)
(node_modules/@openzeppelin/contracts/access/AccessControl.sol#183-185)
renounceRole(bytes32,address) should be declared external:
      - AccessControl.renounceRole(bytes32,address)
(node_modules/@openzeppelin/contracts/access/AccessControl.sol#201-205)
renounceOwnership() should be declared external:
      - Ownable.renounceOwnership()
(node_modules/@openzeppelin/contracts/access/Ownable.sol#53-55)
transferOwnership(address) should be declared external:
      - Ownable.transferOwnership(address)
(node_modules/@openzeppelin/contracts/access/Ownable.sol#61-64)
```

```
name() should be declared external:
      - ERC20.name()
(node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#60-62)
symbol() should be declared external:
      - ERC20.symbol()
(node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#68-70)
decimals() should be declared external:
      - ERC20.decimals()
(node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#85-87)
totalSupply() should be declared external:
      - ERC20.totalSupply()
(node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#92-94)
balanceOf(address) should be declared external:
      - ERC20.balanceOf(address)
(node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#99-101)
transfer(address,uint256) should be declared external:
      - ERC20.transfer(address,uint256)
(node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#111-114)
allowance(address,address) should be declared external:
      - ERC20.allowance(address,address)
(node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#119-121)
approve(address,uint256) should be declared external:
      - ERC20.approve(address,uint256)
(node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#130-133)
transferFrom(address,address,uint256) should be declared external:
      - ERC20.transferFrom(address,address,uint256)
(node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#148-162)
increaseAllowance(address,uint256) should be declared external:
      - ERC20.increaseAllowance(address,uint256)
(node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#176-179)
decreaseAllowance(address,uint256) should be declared external:
      - ERC20.decreaseAllowance(address,uint256)
(node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#195-203)
enableWhitelist(bool) should be declared external:
      - GameFactory.enableWhitelist(bool) (contracts/GameFactory.sol#125-127)
addToWhitelist(address) should be declared external:
      - GameFactory.addToWhitelist(address)
(contracts/GameFactory.sol#133-136)
removeFromWhitelist(address) should be declared external:
      - GameFactory.removeFromWhitelist(address)
(contracts/GameFactory.sol#142-145)
getBaseStakeAmountForPlay() should be declared external:
```

```
	- GameFactory.getBaseStakeAmountForPlay()
(contracts/GameFactory.sol#179-181)
getBaseStakeAmountForEarn() should be declared external:
	- GameFactory.getBaseStakeAmountForEarn()
(contracts/GameFactory.sol#183-185)
getBurnFee() should be declared external:
	- GameFactory.getBurnFee() (contracts/GameFactory.sol#187-189)
getWithdrawFee() should be declared external:
	- GameFactory.getWithdrawFee() (contracts/GameFactory.sol#191-193)
getThawingLockingPeriod() should be declared external:
	- GameFactory.getThawingLockingPeriod()
(contracts/GameFactory.sol#195-197)
getWithdrawLockingPeriod() should be declared external:
	- GameFactory.getWithdrawLockingPeriod()
(contracts/GameFactory.sol#199-201)
_getAddRewardCandidatePeriod() should be declared external:
	- GameFactory._getAddRewardCandidatePeriod()
(contracts/GameFactory.sol#203-205)
getGoldItemPrice() should be declared external:
	- GameFactory.getGoldItemPrice() (contracts/GameFactory.sol#207-209)
getSilverItemPrice() should be declared external:
	- GameFactory.getSilverItemPrice() (contracts/GameFactory.sol#211-213)
getBronzeItemPrice() should be declared external:
	- GameFactory.getBronzeItemPrice() (contracts/GameFactory.sol#215-217)
getRewardStateByUser() should be declared external:
	- GameFactory.getRewardStateByUser()
(contracts/GameFactory.sol#219-221)
getThawingStateByUser() should be declared external:
	- GameFactory.getThawingStateByUser()
(contracts/GameFactory.sol#223-225)
getStakedAmountByUser(address) should be declared external:
	- GameFactory.getStakedAmountByUser(address)
(contracts/GameFactory.sol#227-229)
getStatistic() should be declared external:
	- GameFactory.getStatistic() (contracts/GameFactory.sol#231-233)
unfreeze() should be declared external:
	- GameFactory.unfreeze() (contracts/GameFactory.sol#248-264)
cancel() should be declared external:
	- GameFactory.cancel() (contracts/GameFactory.sol#269-277)
claimReward() should be declared external:
	- GameFactory.claimReward() (contracts/GameFactory.sol#282-306)
freeze() should be declared external:
	- GameFactory.freeze() (contracts/GameFactory.sol#311-326)
```

```
initWhitelist(address[]) should be declared external:
      - GameFactory.initWhitelist(address[])
(contracts/GameFactory.sol#550-558)
removeFromWhitelist(address[]) should be declared external:
      - GameFactory.removeFromWhitelist(address[])
(contracts/GameFactory.sol#563-571)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#public-function
-that-could-be-declared-external
INFO:Slither:. analyzed (16 contracts with 75 detectors), 140 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and
Github integration
```