Quantstamp Security Assessment Certificate

August 12th 2021 — Quantstamp Verified

Good Ghosting Core Files Audit

This smart contract audit was prepared by Quantstamp, the leader in blockchain security.

Executive Summary

Type	DeFi App
Auditors	Poming Lee, Research Engineer Fayçal Lalidji, Security Auditor Cristiano Maciel da Silva, Research Engineer
Timeline	2021-07-19 through 2021-08-12
EVM	Muir Glacier
Languages	Solidity, Javascript
Methods	Architecture Review, Unit Testing, Computer-Aided Verification, Manual Review
Specification	README
Documentation Quality	High
Test Quality	High

Source Code

Repository	Commit
goodghosting-smart-contracts	f1729c4
goodghosting-smart-contracts	5e2eb63

Total Issues	4 (3 Resolved)
High Risk Issues	0 (0 Resolved)
Medium Risk Issues	0 (0 Resolved)
Low Risk Issues	2 (2 Resolved)
Informational Risk Issues	2 (1 Resolved)
Undetermined Risk Issues	0 (0 Resolved)

High Risk	The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.
Medium Risk	The issue puts a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or is reasonably likely to lead to moderate financial impact.
Low Risk	The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low-impact in view of the client's business circumstances.
Informational	The issue does not post an immediate risk, but is relevant to security best practices or Defence in Depth.
Undetermined	The impact of the issue is uncertain.
Unresolved	Acknowledged the existence of the risk, and decided to accept it without engaging in special efforts to control it.
Acknowledged	The issue remains in the code but is a result of an intentional business or design decision. As such, it is supposed to be addressed outside the programmatic means, such as: 1) comments, documentation, README, FAQ; 2) business processes; 3) analyses showing that the issue shall have no negative consequences in practice (e.g., gas analysis, deployment settings).
Resolved	Adjusted program implementation, requirements or constraints to eliminate the risk.
Mitigated	Implemented actions to minimize the impact or likelihood of the risk.

Summary of Findings

During auditing, we found 3 potential issues of various levels of severity: 1 low-severity, and 2 informational-level findings. We made 1 best practices recommendations. We recommend addressing the findings before going live.

Disclaimer: For the current audit, Quantstamp was requested to audit only the following core files of the project: `GoodGhosting.sol`, `GoodGhostingPolygon.sol`, `GoodGhostingPolygonWhitelisted.sol`, `MerkleDistributor.sol`.

2021-08-09 update: during this reaudit, the admin team has brought all of the status of findings either into fixed or acknowledged. In addition, QSP-4, is uncovered by GoodGhosting team and discussed with Quantstamp team. Furthermore, the branch coverage of the tests has been increased from `89.44%` to `90.51%`.

ID	Description	Severity	Status
QSP-1	Unlimited allowance for external contract	Low	Fixed
QSP-2	An attacker could game the platform and gain the bonus from it	Low	Mitigated
QSP-3	Missing input checks	Informational	Fixed
QSP-4	Privileged roles and ownership	Informational	Acknowledged

Quantstamp Audit Breakdown

Quantstamp's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.

Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow / underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Methodology

The Quantstamp auditing process follows a routine series of steps:

1. Code review that includes the following
   i. Review of the specifications, sources, and instructions provided to Quantstamp to make sure we understand the size, scope, and functionality of the smart contract.
   ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   iii. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Quantstamp describe.
2. Testing and automated analysis that includes the following:
   i. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
   ii. Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, and actionable recommendations to help you take steps to secure your smart contracts.

Toolset

The notes below outline the setup and steps performed in the process of this audit.

Setup

Tool Setup:

- SolidityCoverage v0.7.16
- Slither v0.8.0

Steps taken to run the tools:

`npm install solidity-coverage` add `plugins: ["solidity-coverage"]` to `truffle-config.js` change `truffle-config.js` to add a development network

```
development: {
            host: "127.0.0.1",     // Localhost (default: none)
            gasPrice: 100000000000,
            port: 8545,            // Standard Ethereum port (default: none)
            network_id: "5555",       // Any network (default: none)
           },
```

modifying the `secret.localnet.json` to have recipient[1,2]_address from the new network

```
    {
      "mnemonic": "",
      "infura_api_key": "",
      "recipient1_address": "0xb56ec59083bca56e374f25677108cb4534a474d7",
      "recipient2_address": "0xb538d7a6d7495689e2219b26c3e189e2ad3c92e7",
      "usdt_token_address": "",
      "stmx_token_address": "",
      "eth_usd_aggregator_address": "",
      "stmx_usd_aggregator_address": "",
      "usdt_usd_aggregator_address": ""
    }
```

comment out `L2-L3` in `truffle-config.js` `ganache-cli --networkId 5555` `truffle migrate --network development` `truffle run coverage --network development` Installed the Slither tool: `pip install slither-analyzer` Run Slither from the project directory: `slither .`

Findings

QSP-1 Unlimited allowance for external contract

Severity: *Low Risk*

Status: Fixed

File(s) affected: `contracts\GoodGhosting.sol`

Description: In `contracts\GoodGhosting.sol::constructor`: the allowance of `lendingPool` for spending `_inboundCurrency` that is held by `contracts\GoodGhosting.sol` is set to a very large number. This could be used by `lendingPool` to drain the contract, whenever the `lendingPool` becomes malicious, or being controlled by malicious users.

Recommendation: Set up the allowance only when it is necessary, and never make it unlimited.

QSP-2 An attacker could game the platform and gain the bonus from it

Severity: *Low Risk*

Status: Mitigated

Description: The finding is uncovered by the admin team and discussed with Quantstamp team about Capped Games with a small number set for `contracts/GoodGhosting.sol::maxPlayersCount`. The finding is that an attacker could exploit a game capped to a small number of players (i.e., 100 players max) and gain the external incentives sent to the game as sponsorship.

Exploit Scenario: Sample Steps for the Exploit:

1. The admin team deploys a game capped at 100 players (or any other small max number of players that can be exploited by the attacker)
2. The admin team sends external incentives (i.e., 5K in aDAI – or any other amount) to the contract
3. Malicious actor uses some sort of script to join and early withdraw from the game with 99 different accounts, potentially transferring funds between accounts
4. Malicious actor uses a 100th account (not used earlier) to join the game and remain till the end of the game, making all regular deposits. In summary, this 100th account is the only winner in the game.

PS: variations of the above where the attacker does not take all the spots available, but a considerable number of them is also possible, increasing the profit for the attacker (due to the sponsorship sent to the contract)
Considerations: In the scenario above, the only active player in the game would be the malicious actor. If the player completes all deposits in the game, all the additional incentives (5K) would go exclusively to this player. The current Cap logic, considers the number of players in the game without making distinction between players that are "active" or “early withdraw”(the contract only checks “players.length” which still contains players that early withdraw”. So in the scenario above, a single malicious actor, could potentially take all the spots available in the game and cash out the entire additional incentives when the game completes, without requiring a lot of capital (just enough for 1 deposit + paying the early withdrawal fees (which is generally set at 1% of the deposited amount, so considering 99 accounts, with a segment deposit amount of 100 DAI * 1%, that would be close to the amount of a second deposit (i.e., deposit of 100 DAI => 1% * 100 = 1DAI. Then 1DAI * 99 fake accounts, 99 DAI).

Update:
2021-08-09 update:
The admin team refactored the contract and added additional unit tests to use a state variable to control the active players in the game. When a new player joins the game this counter is increased, and when a player early withdraws, this counter is decreased. Also refactored the validation in the `contracts/GoodGhosting.sol::_joinGame` to use this new state variable instead.
By implementing the above, the platform still has the possibility of someone using fake accounts to take all the available spots and receive the full incentives. However, the malicious user would require an upfront amount equal to `depositAmount * MaxPlayersCount` (instead of a smaller amount – in the exploit sample above, before the fix the attacker would need the equivalent of just 2 deposits, and after the fix it would need the amount for 100 deposits), since the player would need to keep all the accounts active in the game (i.e., no early withdrawals).

QSP-3 Missing input checks

Severity: *Informational*

Status: Fixed

File(s) affected: `contracts\GoodGhosting.sol, contracts\GoodGhostingPolygon.sol`

Description: Should check:

1. `contracts\GoodGhosting.sol::constructor`: if `_segmentLength` is not zero.
2. `contracts\GoodGhosting.sol::constructor`: if `_inboundCurrency` is not zero.
3. `contracts\GoodGhosting.sol::constructor`: if `_lendingPoolAddressProvider` is not zero.
4. `contracts\GoodGhosting.sol::constructor`: if `_dataProvider` is not zero.
5. `contracts\GoodGhostingPolygon.sol::constructor`: if `_incentiveController` is not zero.
6. `contracts\GoodGhostingPolygon.sol::constructor`: if `_matic` is not zero.

Recommendation: Add relevant checks to the contract.

QSP-4 Privileged roles and ownership

Severity: Informational

Status: Acknowledged

File(s) affected: `contracts\GoodGhosting.sol`

Description: The owner of the `contracts\GoodGhosting.sol` can pause the contract at any point in time, and stopping users from interacting with the contract, for instance, through the following functions: `joinGame`, `makeDeposit`, `withdraw`, `earlyWithdraw`.

Recommendation: This centralization of power needs to be made clear to the users, especially depending on the level of privilege the contract allows to the owner.

Update:
2021-08-09 update:
The admin team commented on this issue as follows:

```
The team acknowledges the existence of the centralization of power reported in the audit report. Such centralization of power is mitigated by making it clear to users in our documentation available at
https://docs.goodghosting.com/goodghosting/game-mechanics-and-technical.

The team also wants to correct a small misinterpretation described in the report, but significant from an user perspective.

The issue in the report reads as:
“The owner of the contracts\GoodGhosting.sol can pause the contract at any point in time, and stopping users from interacting with the contract, for instance, through the
following functions: joinGame , makeDeposit , withdraw , earlyWithdraw .”

We want to point out that the function “withdraw” is not affected by the “pause” functionality (neither is “redeemFromExternalPool” which may be called from “withdraw”, in case it was not invoked before). This guarantees
that players will always have access to their principal amount at the end of the game.
```

Automated Analyses

Slither

Slither has output 18 results, all of them were examined and found out to be false positives.

Adherence to Best Practices

[Fixed] 1. According to the solidity 0.6.11 documentation, division on integers always round towards zero. It is suggested to check if all divisions that will be rounded towards zero are intended.

Test Results

Test Suite Results

All tests have passed.

```
Contract: GoodGhosting
  pre-flight checks
     ✓ checks if DAI and aDAI contracts have distinct addresses (0ms)
     ✓ checks that contract starts holding 0 Dai and 0 aDai (71ms)
     ✓ checks if player1 received minted DAI tokens (63ms)
     ✓ reverts if the contract is deployed with 0% early withdraw fee (826ms, 1485287 gas)
     ✓ reverts if the contract is deployed with invalid inbound token address (273ms, 1485092 gas)
     ✓ reverts if the contract is deployed with invalid lending pool address (294ms, 1485152 gas)
     ✓ reverts if the contract is deployed with segment count as 0 (184ms, 1485403 gas)
     ✓ reverts if the contract is deployed with segment length as 0 (450ms, 1485426 gas)
     ✓ reverts if the contract is deployed with segment payment as 0 (184ms, 1485389 gas)
     ✓ reverts if the contract is deployed with invalid data provider address (144ms, 1485228 gas)
     ✓ reverts if the contract is deployed with early withdraw fee more than 10% (167ms, 1485276 gas)
     ✓ reverts if the contract is deployed with admin fee more than 20% (471ms, 1485250 gas)
     ✓ reverts if the contract is deployed with max player count equal to zero (161ms, 1485298 gas)
     ✓ accepts setting type(uint256).max as the max number of players (164ms, 3789797 gas)
  when the contract is deployed
     ✓ checks if the contract's variables were properly initialized (153ms)
     ✓ checks if game starts at segment zero (340ms)
     ✓ checks incentive token address is set (215ms, 4550098 gas)
  when the time passes for a game
     ✓ checks if the game segments increase (143ms)
     ✓ checks if the game completes when last segment completes (260ms)
  when an user tries to join a game
     ✓ reverts if the contract is paused (68ms, 53166 gas)
     ✓ reverts if user does not approve the contract to spend dai (339ms, 114374 gas)
     ✓ reverts if the user tries to join after the first segment (353ms, 66618 gas)
     ✓ reverts if the user tries to join the game twice (153ms, 361430 gas)
     ✓ reverts if more players than maxPlayersCount try to join (658ms, 4358794 gas)
     ✓ increases activePlayersCount when a new player joins (120ms, 292962 gas)
     ✓ second player can join after cap spot (maxPlayersCount) is open by an early withdraw (751ms, 4533203 gas)
     ✓ early withdraw player can rejoin if spot (maxPlayersCount) is available (852ms, 4492440 gas)
     ✓ stores the player(s) who joined the game (509ms, 495924 gas)
     ✓ transfers the first payment to the contract (108ms, 292962 gas)
     ✓ emits the event JoinedGame (176ms, 292962 gas)
  when a player tries to rejoin
     ✓ reverts if user tries to rejoin the game after segment 0 (265ms, 414632 gas)
     ✓ reverts if a user tries to rejoin the game in segment 0 without doing an early withdraw (476ms, 361430 gas)
     ✓ user can rejoin the game on segment 0 after an early withdrawal (239ms, 556005 gas)
     ✓ verifies the player info stored in the contract after user rejoins after an early withdraw (580ms, 556029 gas)
     ✓ does not increase the number of players when a user rejoins the game on segment 0 after an early withdrawal (658ms, 739823 gas)
  when an user tries to make a deposit

     ✓ reverts if the contract is paused (58ms, 53188 gas)
     ✓ reverts if user didn't join the game (67ms, 68202 gas)
     ✓ reverts if user tries to deposit during segment 0 (354ms, 361472 gas)
     ✓ reverts if user is making a deposit during segment n (last segment) (153ms, 361459 gas)
     ✓ reverts if user tries to deposit after the game ends (873ms, 1069033 gas)
     ✓ reverts if user is making a duplicated deposit for the same segment (302ms, 516338 gas)
     ✓ reverts if user forgot to deposit for previous segment (479ms, 363358 gas)
     ✓ user can deposit successfully if all requirements are met (502ms, 446959 gas)
     ✓ transfers the payment to the contract (242ms, 446959 gas)
     ✓ makes sure the total principal amount increases (577ms, 446959 gas)
     ✓ makes sure the player info stored in contract is updated (240ms, 446959 gas)
     ✓ makes sure that the winner array contains the player address that makes the last segment deposit (877ms, 1044534 gas)
  when a user withdraws before the end of the game
     ✓ reverts if the contract is paused (445ms, 53167 gas)
     ✓ reverts if the game is completed (32ms, 22551 gas)
     ✓ reverts if a non-player tries to withdraw (158ms, 316395 gas)
     ✓ sets withdrawn flag to true after user withdraws before end of game (496ms, 348014 gas)
     ✓ reverts if user tries to withdraw more than once (476ms, 372290 gas)
     ✓ withdraws user balance subtracted by early withdraw fee (185ms, 348014 gas)
     ✓ fee collected from early withdrawal is part of segment deposit so it should generate interest (559ms, 562660 gas)
     ✓ withdraws user balance subtracted by early withdraw fee when not enough withdrawable balance in the contract (185ms, 348014 gas)
     ✓ emits EarlyWithdrawal event when user withdraws before end of game (177ms, 348014 gas)
     ✓ reverts if user tries to pay next segment after early withdraw (223ms, 415338 gas)
     ✓ user is able to withdraw in the last segment (835ms, 919002 gas)
     ✓ user is able to withdraw in the last segment when 2 players join the game and one of them early withdraws when the segment amount is less than withdraw amount (1785ms, 2049171 gas)
  when an user tries to redeem from the external pool
     ✓ reverts if game is not completed (39ms, 21675 gas)
     ✓ reverts if funds were already redeemed (1007ms, 1119208 gas)
     ✓ allows anyone to redeem from external pool when game is completed (976ms, 1044606 gas)
     ✓ transfer funds to contract then redeems from external pool (1021ms, 1096590 gas)
     ✓ emits event FundsRedeemedFromExternalPool when redeem is successful (1562ms, 1298084 gas)
```

```
    ✓ checks the interest is updated correctly when admin fees is more than 0% (1526ms, 1298072 gas)
    ✓ checks the interest is updated correctly when admin fees is 0 % (1756ms, 3855706 gas)
    ✓ checks totalIncentiveAmount is set when additional incentives are sent to the contract (1685ms, 1298084 gas)
    ✓ emits WinnersAnnouncement event when redeem is successful (995ms, 1096662 gas)
  when incentive token is defined
     ✓ sets totalIncentiveAmount to amount sent to contract (1038ms, 1208151 gas)
     ✓ sets totalIncentiveAmount to zero if no amount is sent to contract (953ms, 1123105 gas)
when no one wins the game
    ✓ transfers interest to the owner in case no one wins (999ms, 914818 gas)
    ✓ transfers principal to the user in case no one wins (1007ms, 949464 gas)
when an user tries to withdraw
    ✓ reverts if user tries to withdraw more than once (1536ms, 1156997 gas)
    ✓ reverts if user tries to withdraw before the game ends (573ms, 323064 gas)
    ✓ reverts if a non-player tries to withdraw (1056ms, 1118956 gas)
    ✓ reverts if a player tries withdraw after doing an early withdraw (516ms, 371967 gas)
    ✓ user is able to withdraw when the contract is paused (400ms, 383868 gas)
    ✓ sets withdrawn flag to true after user withdraws (1007ms, 1133860 gas)
    ✓ withdraws from external pool on first withdraw if funds weren't redeemed yet (974ms, 1106527 gas)
    ✓ makes sure the player that withdraws first before funds are redeemed from external pool gets equal interest (if winner) (3217ms, 2317872 gas)
    ✓ makes sure the winners get equal interest (2564ms, 2343357 gas)
    ✓ pays a bonus to winners in form of early withdraw fees and losers get their principle back (1935ms, 1635843 gas)
    ✓ pays a bonus to winners in form of early withdraw fees and interest earned and losers get their principle back (2656ms, 1797922 gas)
    ✓ pays a bonus to winners and losers get their principle back (2923ms, 1613335 gas)
    ✓ emits Withdrawal event when user withdraws (1619ms, 1133932 gas)
  when incentive token is defined
     ✓ pays additional incentive to winners when incentive is sent to contract (1928ms, 1530939 gas)
     ✓ does not pay additional incentive to winners if incentive is not sent to contract (1742ms, 1427995 gas)
admin tries to withdraw fees with admin percentage fee greater than 0
  reverts
     ✓ when funds were not redeemed from external pool (1530ms, 1068058 gas)
     ✓ when game has not completed yet (550ms, 315546 gas)
     ✓ when admin tries to withdraw fees again (1459ms, 1406325 gas)
     ✓ someone other than admin tries to withdraw the fees (1492ms, 1320309 gas)
  with no winners in the game
     ✓ does not revert when there is no interest generated (neither external interest nor early withdrawal fees) (368ms, 395771 gas)
     ✓ withdraw fees when there's only early withdrawal fees (757ms, 730278 gas)
     ✓ withdraw fees when there's only interest generated by external pool (828ms, 833016 gas)
     ✓ withdraw fees when there's both interest generated by external pool and early withdrawal fees (915ms, 892357 gas)
     ✓ withdraw incentives sent to contract (805ms, 5067393 gas)
  with winners in the game
     ✓ does not revert when there is no interest generated (neither external interest nor early withdrawal fees) (1007ms, 1147792 gas)
     ✓ withdraw fees when there's only early withdrawal fees (1761ms, 1543026 gas)
     ✓ withdraw fees when there's only interest generated by external pool (1556ms, 1382075 gas)
10610000000000000000000
6000000000000000000
10010000000000000000000
9509500000000000000000
5005000000000000000000
     ✓ withdraw fees when there's both interest generated by external pool and early withdrawal fees (1775ms, 1705105 gas)
     ✓ does not withdraw any incentives sent to contract (1813ms, 5809690 gas)
  admin tries to withdraw fees with admin percentage fee equal to 0 and no winners
     ✓ does not revert when there is no interest generated (1641ms, 1374306 gas)
     ✓ withdraw fees when there's only interest generated by external pool (792ms, 832696 gas)
     ✓ withdraw fees when there's only early withdraw fees and no winners in the game (838ms, 770727 gas)
     ✓ withdraw fees when there are both early withdraw fees and interest and no winners in the game (909ms, 929432 gas)
     ✓ withdraw incentives sent to contract (932ms, 5067154 gas)
  as a Pausable contract
    checks Pausable access control
     ✓ does not revert when admin invokes pause() (0ms)
     ✓ does not revert when admin invokes unpause() (63ms, 30891 gas)
     ✓ reverts when non-admin invokes pause() (188ms, 22288 gas)
     ✓ reverts when non-admin invokes unpause() (72ms, 53223 gas)
    checks Pausable contract default behavior
      checks Pausable contract default behavior
        ✓ pauses the contract (25ms)
        ✓ unpauses the contract (329ms, 30917 gas)

Contract: GoodGhostingPolygon
  pre-flight checks
    ✓ checks if DAI and aDAI contracts have distinct addresses (0ms)
    ✓ checks that contract starts holding 0 Dai and 0 aDai (23ms)
    ✓ checks if player1 received minted DAI tokens (12ms)
    ✓ reverts if the contract is deployed with 0% early withdraw fee (159ms, 1513661 gas)
    ✓ reverts if the contract is deployed with early withdraw fee more than 10% (153ms, 1513674 gas)
    ✓ reverts if the contract is deployed with admin fee more than 20% (535ms, 1513648 gas)
    ✓ reverts if the contract is deployed with max player count equal to zero (556ms, 1513696 gas)
    ✓ reverts if the contract is deployed with invalid inbound token address (653ms, 1513490 gas)
    ✓ reverts if the contract is deployed with invalid lending pool address (635ms, 1513526 gas)
    ✓ reverts if the contract is deployed with segment count as 0 (690ms, 1513801 gas)
    ✓ reverts if the contract is deployed with segment length as 0 (686ms, 1513824 gas)
    ✓ reverts if the contract is deployed with segment payment as 0 (673ms, 1513787 gas)
    ✓ reverts if the contract is deployed with invalid data provider address (638ms, 1513602 gas)
    ✓ reverts if the contract is deployed with invalid incentive controller address (678ms, 1538738 gas)
    ✓ reverts if the contract is deployed with invalid matic token address (790ms, 1538745 gas)
    ✓ accepts setting type(uint256).max as the max number of players (621ms, 4162042 gas)
  when the contract is deployed
    ✓ checks if the contract's variables were properly initialized (610ms)
    ✓ checks if game starts at segment zero (11ms)
  when an user tries to redeem from the external pool
    ✓ reverts if funds were already redeemed (1103ms, 1193387 gas)
    ✓ allows to redeem from external pool when game is completed (1457ms, 1044496 gas)
    ✓ transfer funds to contract then redeems from external pool (1814ms, 1170841 gas)
    ✓ emits event FundsRedeemedFromExternalPool when redeem is successful (1798ms, 1170841 gas)
    ✓ emits WinnersAnnouncement event when redeem is successful (1722ms, 1170841 gas)
    ✓ allocates external rewards sent to contract to the players (2697ms, 1206788 gas)
  when incentive token is defined
     ✓ sets totalIncentiveAmount to amount sent to contract (1966ms, 1282330 gas)
     ✓ sets totalIncentiveAmount to zero if no amount is sent to contract (1915ms, 1197284 gas)
  when no one wins the game
    ✓ transfers interest to the owner in case no one wins (1599ms, 968914 gas)
    ✓ transfers principal to the user in case no one wins (1787ms, 1003506 gas)
  when an user tries to withdraw
    ✓ reverts if user tries to withdraw more than once (1916ms, 1248252 gas)
    ✓ reverts if a non-player tries to withdraw (713ms, 316385 gas)
    ✓ sets withdrawn flag to true after user withdraws (1758ms, 1225137 gas)
    ✓ withdraws from external pool on first withdraw if funds weren't redeemed yet (1719ms, 1187052 gas)
    ✓ makes sure the player that withdraws first before funds are redeemed from external pool gets equal interest (if winner) (3597ms, 2436793 gas)
    ✓ makes sure the winners get equal interest (4828ms, 2466478 gas)
    ✓ pays a bonus to winners and losers get their principle back (2997ms, 1704546 gas)
    ✓ emits Withdrawal event when user withdraws (1977ms, 1225137 gas)
  when incentive token is defined
     ✓ pays additional incentive to winners when incentive is sent to contract (2663ms, 1622150 gas)
     ✓ does not pay additional incentive to winners if incentive is not sent to contract (2626ms, 1519206 gas)
  admin tries to withdraw fees with admin percentage fee greater than 0
    reverts
     ✓ when funds were not redeemed from external pool (1660ms, 1067948 gas)
     ✓ when admin tries to withdraw fees again (1915ms, 1481321 gas)
    with no winners in the game
     ✓ does not revert when there is no interest generated (neither external interest nor early withdrawal fees) (784ms, 470727 gas)
     ✓ withdraw fees when there's only early withdrawal fees (980ms, 746283 gas)
     ✓ withdraw fees when there's only interest generated by external pool (1618ms, 907972 gas)
     ✓ withdraw fees when there's both interest generated by external pool and early withdrawal fees (1723ms, 1004686 gas)
     ✓ withdraw incentives sent to contract (1656ms, 5514642 gas)
    with winners in the game
     ✓ does not revert when there is no interest generated (neither external interest nor early withdrawal fees) (1730ms, 1222788 gas)
     ✓ withdraw fees when there's only early withdrawal fees (1888ms, 1618011 gas)
     ✓ withdraw fees when there's only interest generated by external pool (2495ms, 1457071 gas)
10610000000000000000000
6000000000000000000
10010000000000000000000
9509500000000000000000
5005000000000000000000
     ✓ withdraw fees when there's both interest generated by external pool and early withdrawal fees (2674ms, 1780090 gas)
     ✓ does not withdraw any incentives sent to contract (2693ms, 6256859 gas)
  admin tries to withdraw fees with admin percentage fee equal to 0 and no winners
     ✓ does not revert when there is no interest generated (2655ms, 5610972 gas)
     ✓ withdraw incentives sent to contract (1541ms, 5514355 gas)

Contract: GoodGhostingPolygonWhitelisted
  pre-flight checks
    ✓ checks if DAI and aDAI contracts have distinct addresses (0ms)
    ✓ checks that contract starts holding 0 Dai and 0 aDai (548ms)
    ✓ checks if player1 received minted DAI tokens (568ms)
    ✓ reverts if the contract is deployed with 0% early withdraw fee (762ms, 1530114 gas)
    ✓ reverts if the contract is deployed with early withdraw fee more than 10% (817ms, 1530127 gas)
    ✓ reverts if the contract is deployed with admin fee more than 20% (894ms, 1530101 gas)
    ✓ reverts if the contract is deployed with max player count equal to zero (859ms, 1530149 gas)
    ✓ accepts setting type(uint256).max as the max number of players (879ms, 4377447 gas)
    ✓ reverts if the contract is deployed with invalid inbound token address (849ms, 1529931 gas)
    ✓ reverts if the contract is deployed with invalid lending pool address (921ms, 1530003 gas)
    ✓ reverts if the contract is deployed with segment count as 0 (944ms, 1530254 gas)
    ✓ reverts if the contract is deployed with segment length as 0 (849ms, 1530277 gas)
    ✓ reverts if the contract is deployed with segment payment as 0 (885ms, 1530216 gas)
    ✓ reverts if the contract is deployed with invalid data provider address (370ms, 1530079 gas)
    ✓ reverts if the contract is deployed with invalid incentive controller address (881ms, 1555191 gas)
    ✓ reverts if the contract is deployed with invalid matic token address (869ms, 1555198 gas)
  when the contract is deployed
    ✓ checks if the contract's variables were properly initialized (838ms)
    ✓ checks if game starts at segment zero (705ms)
  when an user tries to join a game
    ✓ reverts if the contract is paused (789ms, 54327 gas)
    ✓ reverts if user does not approve the contract to spend dai (762ms, 116326 gas)
    ✓ reverts if the user tries to join after the first segment (749ms, 68569 gas)
    ✓ reverts when a non-whitelisted player tries to join the game (696ms, 25833 gas)
    ✓ reverts when whitelisted user tries to join using joinGame() instead of joinWhitelistedGame(...) (726ms, 22281 gas)
    ✓ reverts when non-whitelisted user tries to join using joinGame() joinWhitelistedGame(...) (726ms, 22281 gas)
```

```
      ✓ reverts if the user tries to join the game twice (893ms, 365339 gas)
      ✓ reverts if more players than maxPlayersCount try to join (1015ms, 4747437 gas)
      ✓ stores the player(s) who joined the game (940ms, 499838 gas)
      ✓ emits the event JoinedGame (771ms, 294920 gas)

·---------------------------------------------------------------|---------------------------|-------------|----------------------------·
|              Solc version: 0.6.11+commit.5ef660b1             ·  Optimizer enabled: true  ·  Runs: 1500  ·  Block limit: 6718946 gas  |
················································|···························|·············|·····························
|  Methods                                                                                                                              |
····································|···························|·····|·····|·····|·····|·····
|  Contract                        ·  Method                   ·  Min  ·  Max  ·  Avg  ·  # calls  ·  usd (avg)  |
····································|···························|·····|·····|·····|·····|·····
|  GoodGhosting                    ·  adminFeeWithdraw         ·  51130  ·  84802  ·  73079  ·  31  ·  -  |
····································|···························|·····|·····|·····|·····|·····
|  GoodGhosting                    ·  earlyWithdraw            ·  54136  ·  66736  ·  59014  ·  32  ·  -  |
····································|···························|·····|·····|·····|·····|·····
|  GoodGhosting                    ·  joinGame                 ·  133105  ·  263868  ·  228032  ·  111  ·  -  |
····································|···························|·····|·····|·····|·····|·····
|  GoodGhosting                    ·  makeDeposit              ·  94903  ·  144103  ·  107803  ·  208  ·  -  |
····································|···························|·····|·····|·····|·····|·····
|  GoodGhosting                    ·  pause                    ·  -  ·  -  ·  30891  ·  11  ·  -  |
····································|···························|·····|·····|·····|·····|·····
|  GoodGhosting                    ·  redeemFromExternalPool   ·  50868  ·  92526  ·  72497  ·  47  ·  -  |
····································|···························|·····|·····|·····|·····|·····
|  GoodGhosting                    ·  unpause                  ·  -  ·  -  ·  30917  ·  2  ·  -  |
····································|···························|·····|·····|·····|·····|·····
|  GoodGhosting                    ·  withdraw                 ·  34646  ·  119311  ·  53117  ·  30  ·  -  |
····································|···························|·····|·····|·····|·····|·····
|  GoodGhostingPolygon             ·  adminFeeWithdraw         ·  51947  ·  105574  ·  80798  ·  24  ·  -  |
····································|···························|·····|·····|·····|·····|·····
|  GoodGhostingPolygon             ·  earlyWithdraw            ·  55857  ·  66714  ·  60877  ·  4  ·  -  |
····································|···························|·····|·····|·····|·····|·····
|  GoodGhostingPolygon             ·  joinGame                 ·  158868  ·  263868  ·  234400  ·  44  ·  -  |
····································|···························|·····|·····|·····|·····|·····
|  GoodGhostingPolygon             ·  makeDeposit              ·  94881  ·  136540  ·  107877  ·  144  ·  -  |
····································|···························|·····|·····|·····|·····|·····
|  GoodGhostingPolygon             ·  redeemFromExternalPool   ·  88128  ·  166815  ·  131304  ·  35  ·  -  |
····································|···························|·····|·····|·····|·····|·····
|  GoodGhostingPolygon             ·  withdraw                 ·  34652  ·  221426  ·  78731  ·  24  ·  -  |
····································|···························|·····|·····|·····|·····|·····
|  GoodGhostingPolygonWhitelisted  ·  joinWhitelistedGame      ·  160824  ·  250826  ·  220825  ·  6  ·  -  |
····································|···························|·····|·····|·····|·····|·····
|  GoodGhostingPolygonWhitelisted  ·  pause                    ·  -  ·  -  ·  30914  ·  1  ·  -  |
····································|···························|·····|·····|·····|·····|·····
|  IncentiveControllerMock         ·  mint                     ·  -  ·  -  ·  65947  ·  1  ·  -  |
····································|···························|·····|·····|·····|·····|·····
|  LendingPoolAddressesProviderMock  ·  deposit                ·  45857  ·  75869  ·  48176  ·  26  ·  -  |
····································|···························|·····|·····|·····|·····|·····
|  LendingPoolAddressesProviderMock  ·  setUnderlyingAssetAddress  ·  27579  ·  27591  ·  27590  ·  254  ·  -  |
····································|···························|·····|·····|·····|·····|·····
|  LendingPoolAddressesProviderMock  ·  transfer               ·  21234  ·  36246  ·  22496  ·  24  ·  -  |
····································|···························|·····|·····|·····|·····|·····
|  MockERC20Mintable               ·  approve                  ·  24882  ·  44106  ·  42078  ·  562  ·  -  |
····································|···························|·····|·····|·····|·····|·····
|  MockERC20Mintable               ·  mint                     ·  35858  ·  65858  ·  57933  ·  424  ·  -  |
····································|···························|·····|·····|·····|·····|·····
|  Deployments                     ·                           ·                 ·  % of limit  ·             |
····································|···························|·····|·····|·····|·····|·····
|  GoodGhosting                    ·                           ·  2577106  ·  2577766  ·  2577195  ·  38.4 %  ·  -  |
····································|···························|·····|·····|·····|·····|·····
|  GoodGhostingPolygon             ·                           ·  2949375  ·  2950011  ·  2949469  ·  43.9 %  ·  -  |
····································|···························|·····|·····|·····|·····|·····
|  GoodGhostingPolygonWhitelisted  ·                           ·  3164780  ·  3165164  ·  3164821  ·  47.1 %  ·  -  |
····································|···························|·····|·····|·····|·····|·····
|  IncentiveControllerMock         ·                           ·  -  ·  -  ·  1265289  ·  18.8 %  ·  -  |
····································|···························|·····|·····|·····|·····|·····
|  LendingPoolAddressesProviderMock  ·                         ·  -  ·  -  ·  1184692  ·  17.6 %  ·  -  |
····································|···························|·····|·····|·····|·····|·····
|  MockERC20Mintable               ·                           ·  759965  ·  760085  ·  759974  ·  11.3 %  ·  -  |
·---------------------------------------------------------------|-----|-----|-----|-----|-----·

  192 passing (48m)
```

Code Coverage

Excellent code coverage has been achieved for this project.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	100	90.51	100	100	
GoodGhosting.sol	100	93	100	100	
GoodGhostingPolygon.sol	100	85.71	100	100	
GoodGhostingPolygonWhitelisted.sol	100	100	100	100	
MerkleDistributor.sol	100	100	100	100	
contracts/merkle/	100	100	100	100	
IMerkleDistributor.sol	100	100	100	100	
contracts/moola/	100	100	100	100	
ILendingPoolCore.sol	100	100	100	100	
MAToken.sol	100	100	100	100	
MILendingPool.sol	100	100	100	100	
contracts/quickswap/	100	100	100	100	
IPair.sol	100	100	100	100	
IRouter.sol	100	100	100	100	
IStake.sol	100	100	100	100	
All files	**100**	**90.51**	**100**	**100**	

Appendix

File Signatures

The following are the SHA-256 hashes of the reviewed files. A file with a different SHA-256 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different SHA-256 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review.

Contracts

```
490b71bf0b71057a985e03bc414083f05f4a9e24e04caa0aa74f0ff7fda0de94 ./_audited/GoodGhosting.sol
c2f0cffeb94ce4bfac1c941899bc28ab7d7b222a6aa921ba9e463431c1fb70c8 ./_audited/GoodGhostingPolygon.sol
f8122ec18ecf7b2fb9d876d2ede94ebdfee41510c765e39aff7ddc12212b8851 ./_audited/GoodGhostingPolygonWhitelisted.sol
a4bfdaa05b6b382b97acbfaef214c4d782b3bff7c38a06f510a82b3ee3a77248 ./_audited/MerkleDistributor.sol
```

Tests

```
4c1f623f4023c53c35add93b39d18779fd2c339593d585345b5fa098e9cbe486 ./test/.eslintrc.js
0fc5a99abeff7459f612a605bdee6a052f4660532405e9a3de255313b5e58cb7 ./test/contracts.js
6ee5c97085035cfaad94fef5429a7db8ac1993e6614e3296328fbf27dd869987 ./test/GoodGhosting.test.js
f01d935b7589c7b3c95d03e92ae8e091095edb8baf3105a1ae8cc9c4f53511e2 ./test/GoodGhostingGasEstimate.test.js
4e628e78c1348019a67b1627bae5f422a53bd54472b9e3ff43d09593bb033383 ./test/GoodGhostingPolygon.test.js
3649c631e1c9f64dda7780e22142c266b867c05297a303baabcd50045c63f9f7 ./test/GoodGhostingPolygonWhitelisted.test.js
e49ead36c1e4e18d417e58d23dbba9bc9343f37239c4dc778c793da966f69155 ./test/GoodGhosting_Attacker_Sends_DAI_Externally.test.js
ba369393ac8d4f4c2f75bf127f0fda87065eb10f099eca6d3affd2a50fb3fde0 ./test/GoodGhosting_Funds_Redeemed_From_Contract_On_EarlyWithdraw.test.js
f9600192affa626dbea919a29ca7e651a6cf58cd2673356a3e81c8b01aeb2aa3 ./test/GoodGhosting_No_Player_Wins.test.js
f294a647fadad34758a335e2f5ce30f19c5bcae3e0ff0ce6dfaf05091d08d51b ./test/GoodGhosting_Send_AToken_Externally.test.js
```

Changelog

- 2021-07-28 - Initial report
- 2021-08-12 - Final report

About Quantstamp

Quantstamp is a Y Combinator-backed company that helps to secure blockchain platforms at scale using computer-aided reasoning tools, with a mission to help boost the adoption of this exponentially growing technology.

With over 1000 Google scholar citations and numerous published papers, Quantstamp's team has decades of combined experience in formal verification, static analysis, and software verification. Quantstamp has also developed a protocol to help smart contract developers and projects worldwide to perform cost-effective smart contract security scans.

To date, Quantstamp has protected $5B in digital asset risk from hackers and assisted dozens of blockchain projects globally through its white glove security assessment services. As an evangelist of the blockchain ecosystem, Quantstamp assists core infrastructure projects and leading community initiatives such as the Ethereum Community Fund to expedite the adoption of blockchain technology.

Quantstamp's collaborations with leading academic institutions such as the National University of Singapore and MIT (Massachusetts Institute of Technology) reflect our commitment to research, development, and enabling world-class blockchain security.