Gitcoin Passport Eligibity Module Report

Version 1.0

Jacob Homanics

May 10, 2024

Gitcoin Passport Eligibity Module Report

Jacob Homanics

May 10, 2024

Prepared by: Jacob Homanics

Table of Contents

Protocol Summary

GitcoinPassportEligibility Module is a module used primarliy for the Hats ecosystem to be integrated as an eligiblity module for Hats. It utilizes gitcoin passport to determine eligiblity based on whether an addresses' score passes a certain threshold.

Disclaimer

Jacob Homanics makes all efforts to find as many vulnerabilities in the code in the given time period, but holds no responsibilities for the findings provided in this document. A security audit by Jacob Homanics is not an endorsement of the underlying business or product. The audit was time-boxed and the review of the code was solely on the security aspects of the Solidity implementation of the contracts.

Risk Classification

		Impact		
		High	Medium	Low
	High	H	H/M	M
Likelihood	Medium	H/M	M	M/L
	Low	M	M/L	L
	Informational	None	None	None
	Gas	None	None	None

We use the CodeHawks severity matrix to determine severity. See the documentation for more details.

Audit Details

The findings in this document correspond with the following commit hash: Commit Hash:

```
ba4b2663761d1809a0e16b5b716ce376aabc531c
```

Scope

```
GitcoinPassportEligibility.sol
```

Roles

N/A

Executive Summary

The codebase is small and served a single purpose, resulting in no major or critical risks. However we found several Informational or Gas vulnerabilities.

The tools used were VSCode, Slither, and Aderyn.

Issues found

Severity	Number of issues found
High	0
Medium	0
Low	0
Informational	3
Gas	1
Total	4

Findings

Informational

[I-1] GitcoinPassportEligibility::GITCOIN_PASSPORT_DECODER function does not follow the mixedCase naming convention, resulting in potential confusion from code reviewers

Description: All caps naming convention is reserved for constant variables. Although `GitcoinPassportEligibil` `::GITCOIN_PASSPORT_DECODER` returns an immutable constant value, it is still a function. Thus it should follow the mixedCase naming convention.

Impact: Reduces the understanding and potential interactibility of the protocol, and muddies up automated tool's results..

Proof of Concept: Patrick Collins, a leader security smart contract auditor and educator follows the mixedCase naming convention. Alongside automated tools like Slither and Aderyn to report instances of functions not being correctly in mixedCase. Newcomers and the majority of developers, auditors, and researchers will follow these conventions. Alongside muddying up the information that is returned from the automated tools.

Recommended Mitigation: Rename `GitcoinPassportEligibility::GITCOIN_PASSPORT_DECODER` to `GitcoinPassportEligibility::gitcoinPassportDecoder` to satisfy the requirement of functions being in mixedCase.

[I-2] GitcoinPassportEligibility::SCORE_CRITERION function does not follow the mixedCase naming convention, resulting in potential confusion from code reviewers

Description: All caps naming convention is reserved for constant variables. Although `GitcoinPassportEligibil` `::SCORE_CRITERION` returns an immutable constant value, it is still a function. Thus it should follow the mixedCase naming convention.

Impact: Reduces the understanding and potential interactibility of the protocol, and muddies up automated tool's results..

Proof of Concept: Patrick Collins, a leader security smart contract auditor and educator follows the mixedCase naming convention. Alongside automated tools like Slither and Aderyn to report instances of functions not being correctly in mixedCase. Newcomers and the majority of developers, auditors, and researchers will follow these conventions. Alongside muddying up the information that is returned from the automated tools.

Recommended Mitigation: Rename `GitcoinPassportEligibility::SCORE_CRITERION` to `GitcoinPassportEligibility::scoreCriterion` to satisfy the requirement of functions being in mixedCase.

[I-3] GitcoinPassportEligibility::getWearerStatus' first parameter, _wearer, does not follow the mixedCase naming convention, resulting in potential confusion from code reviewers

Description: The underscore naming convention is an outdated practice for function parameters.

Impact: Reduces the understanding and potential interactibility of the protocol, and muddies up automated tool's results.

Proof of Concept: Patrick Collins, a leader security smart contract auditor and educator follows the mixedCase naming convention. Alongside automated tools like Slither and Aderyn to report instances of functions not being correctly in mixedCase. Newcomers and the majority of developers, auditors, and researchers will follow these conventions. Alongside muddying up the information that is returned from the automated tools.

Recommended Mitigation: Rename `GitcoinPassportEligibility::getWearerStatus`' first parameter, `_wearer`, to `wearer` to satisfy the requirement of functions being in mixedCase.

[I-4] GitcoinPassportEligibility::isHuman' first parameter, _wearer, does not follow the mixedCase naming convention, resulting in potential confusion from code reviewers

Description: The underscore naming convention is an outdated practice for function parameters.

Impact: Reduces the understanding and potential interactibility of the protocol, and muddies up automated tool's results.

Proof of Concept: Patrick Collins, a leader security smart contract auditor and educator follows the mixedCase naming convention. Alongside automated tools like Slither and Aderyn to report instances of functions not being correctly in mixedCase. Newcomers and the majority of developers, auditors, and researchers will follow these conventions. Alongside muddying up the information that is returned from the automated tools.

Recommended Mitigation: Rename `GitcoinPassportEligibility::isHuman`' first parameter, `_wearer`, to `wearer` to satisfy the requirement of functions being in mixedCase.

Gas

[G-1] `GitcoinPassportEligibility::getWearerStatus` does not have the most efficient visibility type.

Description: `GitcoinPassportEligibility::getWearerStatus` is not called within `GitcoinPassportEligibility`, however its visibility is **`public`**.

Impact: Increases the gas cost of calling the function.

Proof of Concept: We can see that through fuzz testing public and external functions with the same parameters and operations, the external function resulted in costing less gas to call.

```
test_externalFunction(uint256[20])(runs: 257, delta: 255839, ~:
255839)
```

```
test_publicFunction(uint256[20])(runs: 257, delta: 257286, ~: 257286)
```

Recommended Mitigation: Change `GitcoinPassportEligibility::getWearerStatus`'s visibility from **`public`** to `external`.

[G-2] `GitcoinPassportEligibility::isHuman`'s local function calls are not optimized for gas.

Description: `GitcoinPassportEligibility::isHuman` contains several view function calls of the same function.

Impact: Increases the gas cost of calling the function.

Proof of Concept: Optimizing the function reduces the gas costs.

```
test_isHuman()(gas: 282444)
```

```
test_isHumanOptimized()(gas: 282363)
```

Recommended Mitigation: Store the `GITCOIN_PASSPORT_DECODER` and `SCORE_CRITERION` return values in local variables within the function.