



Gitcoin Passport Eligibility Module Report

Version 1.0

Jacob Homanics

May 27, 2024

Gitcoin Passport Deocder Report

Jacob Homanics

May 27, 2024

Prepared by: Jacob Homanics

Table of Contents

- Table of Contents
- Protocol Summary
- Disclaimer
- Risk Classification
- Audit Details
 - Scope
 - Roles
- Executive Summary
 - Issues found
- Findings
- High
- Medium
- Informational

Protocol Summary

GitcoinPassportDecoder is used to retrieve an addresses' passport score entirely onchain.

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
Likelihood	High	H	H/M	M
	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:

```
1 e347146
```

Scope

```
1 GitcoinPassportDecoder.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	1
Medium	1
Low	0
Informational	6
Gas	0
Total	8

Findings

High

[H-1] GitcoinPassportDecoder is not protected from being self destructed.

Description: GitcoinPassportDecoder is an upgradeable smart contract. This means that there is a possibility of an exploit where a hacker can call self destruct on it to remove it from the blockchain.

Impact: Potentially removes the current implementation from the current address which the instance resides.

Proof of Concept: Add a constructor to ensure initialize cannot be called on the logic contract.

```
1 error GitcoinPassportDecoder__CannotInitializeInLogic();
2 bool s_isLogicContract;
3
4 constructor() {
5     s_isLogicContract = true;
6 }
7
8 function initialize(...) public initializer {
9     if (s_isLogicContract)
10         revert GitcoinPassportDecoder__CannotInitializeInLogic();
11
12     ...
13     _;
14 }
```

Medium

[M-1] GitcoinPassportDecoder::getPassport(address).credential is never initialized.

Description: The local variable, credential, is never initialized. May result in undesired effects.

Impact: In it's current implementation, there does not seem to be a vulnerability. However, it is recommended to follow best practices to increase code readability and to future proof any potential changes to the data structures.

Proof of Concept: Initialize the credentials variable.

Change

```
1 Credential memory credential;
2 // Set provider to the credential struct from the mappedProviders
  mapping
3 credential.provider = mappedProviders[mappedProvidersIndex];
4 // Set the hash to the credential struct from the hashes array
5 credential.hash = hashes[hashIndex];
6 // Set the issuanceDate of the credential struct to the item at the
  current index of the issuanceDates array
7 credential.time = issuanceDates[hashIndex];
8 // Set the expirationDate of the credential struct to the item at the
  current index of the expirationDates array
9 credential.expirationTime = expirationDates[hashIndex];
10
11 // Set the hashIndex with the finished credential struct
12 passportMemoryArray[hashIndex] = credential;
13
14 hashIndex += 1;
```

to

```
1 Credential memory credential = new Credential(mappedProviders[
    mappedProvidersIndex], hashes[hashIndex], issuanceDates[hashIndex],
    expirationDates[hashIndex]);
2
3 // Set the hashIndex with the finished credential struct
4 passportMemoryArray[hashIndex] = credential;
5
6 hashIndex += 1;
```

Informational

[I-1] GitcoinPassportDecoder::setEASAddress(address)._easContractAddress parameter does not follow the mixedCase naming convention, resulting in potential confusion from code reviewers/tools

Description: Variables/parameters should not start with an underscore.

Impact: Reduces the understanding and potential interactability 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 variables 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 `GitcoinPassportDecoder::setEASAddress(address)._easContractAddress` to `GitcoinPassportDecoder::setEASAddress(address).easContractAddress` to satisfy the requirement of variables being in mixedCase.

[I-2] GitcoinPassportDecoder::setGitcoinResolver(address)._gitcoinResolver parameter does not follow the mixedCase naming convention, resulting in potential confusion from code reviewers/tools

Description: Variables/parameters should not start with an underscore.

Impact: Reduces the understanding and potential interactability 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 variables 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 `GitcoinPassportDecoder::setGitcoinResolver(address)._gitcoinResolver` to `GitcoinPassportDecoder::setGitcoinResolver(address).gitcoinResolver` to satisfy the requirement of variables being in mixedCase.

[I-3] GitcoinPassportDecoder::setPassportSchemaUID(bytes32)._schemaUID parameter does not follow the mixedCase naming convention, resulting in potential confusion from code reviewers/tools

Description: Variables/parameters should not start with an underscore.

Impact: Reduces the understanding and potential interactability 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 variables 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 `GitcoinPassportDecoder::setPassportSchemaUID(bytes32)._schemaUID` to `GitcoinPassportDecoder::setPassportSchemaUID(bytes32).schemaUID` to satisfy the requirement of variables being in mixedCase.

[I-4] GitcoinPassportDecoder::setScoreSchemaUID(bytes32)._schemaUID parameter does not follow the mixedCase naming convention, resulting in potential confusion from code reviewers/tools

Description: Variables/parameters should not start with an underscore.

Impact: Reduces the understanding and potential interactability 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 variables 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 `GitcoinPassportDecoder::setScoreSchemaUID(bytes32)._schemaUID` to `GitcoinPassportDecoder::setScoreSchemaUID(bytes32).schemaUID` to satisfy the requirement of variables being in mixedCase.

[I-5] GitcoinPassportDecoder::setMaxScoreAge(uint64)._maxScoreAge parameter does not follow the mixedCase naming convention, resulting in potential confusion from code reviewers/tools

Description: Variables/parameters should not start with an underscore.

Impact: Reduces the understanding and potential interactability 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 variables 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 `GitcoinPassportDecoder::setMaxScoreAge(uint64).maxScoreAge` to `GitcoinPassportDecoder::setMaxScoreAge(uint64).maxScoreAge` to satisfy the requirement of variables being in mixedCase.

[I-6] GitcoinPassportDecoder::setThreshold(uint256)._threshold parameter does not follow the mixedCase naming convention, resulting in potential confusion from code reviewers/tools

Description: Variables/parameters should not start with an underscore.

Impact: Reduces the understanding and potential interactability 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 variables 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 `GitcoinPassportDecoder::setThreshold(uint256)._threshold` to `GitcoinPassportDecoder::setThreshold(uint256).threshold` to satisfy the requirement of variables being in mixedCase.