Gitcoin Passport Eligibity 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

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
	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:

```
e347146
```

Scope

```
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	4
Informational	6
Gas	0
Total	12

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.

```
error GitcoinPassportDecoder__CannotInitializeInLogic();
bool s_isLogicContract;

constructor() {
    s_isLogicContract = true;
}

function initialize(...) public initializer {
    if (s_isLogicContract)
        revert GitcoinPassportDecoder__CannotInitializeInLogic();

        ...
        _;
}
```

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 reccomended to follow best practices to increase code readibility and to future proof any potential changes to the data structures.

Proof of Concept: Initialize the credentials variable.

Change

```
Credential memory credential;
// Set provider to the credential struct from the mappedProviders
    mapping
credential.provider = mappedProviders[mappedProvidersIndex];
// Set the hash to the credential struct from the hashes array
credential.hash = hashes[hashIndex];
// Set the issuanceDate of the credential struct to the item at the
    current index of the issuanceDates array
credential.time = issuanceDates[hashIndex];
// Set the expirationDate of the credential struct to the item at the
    current index of the expirationDates array
credential.expirationTime = expirationDates[hashIndex];

// Set the hashIndex with the finished credential struct
passportMemoryArray[hashIndex] = credential;

hashIndex += 1;
```

to

```
Credential memory credential = new Credential(mappedProviders[
    mappedProvidersIndex], hashes[hashIndex], issuanceDates[hashIndex],
    expirationDates[hashIndex]);

// Set the hashIndex with the finished credential struct
passportMemoryArray[hashIndex] = credential;

hashIndex += 1;
```

Low

[L-1] Centralization Risk for trusted owners

Description: Contracts have owners with privileged rights to perform admin tasks and need to be trusted to not perform malicious updates or drain funds.

Impact:

11 Found Instances

- Found in contracts/GitcoinPassportDecoder.sol Line: 93

```
    function pause() public onlyOwner {
```

- Found in contracts/GitcoinPassportDecoder.sol Line: 97

```
    function unpause() public onlyOwner {
```

- Found in contracts/GitcoinPassportDecoder.sol Line: 101

```
    function _authorizeUpgrade(address) internal override onlyOwner
        {}
```

- Found in contracts/GitcoinPassportDecoder.sol Line: 114

```
    function setEASAddress(address _easContractAddress) external
        onlyOwner {
```

- Found in contracts/GitcoinPassportDecoder.sol Line: 126

```
    function setGitcoinResolver(address _gitcoinResolver) external
        onlyOwner {
```

- Found in contracts/GitcoinPassportDecoder.sol Line: 138

```
    function setPassportSchemaUID(bytes32 _schemaUID) public
        onlyOwner {
```

- Found in contracts/GitcoinPassportDecoder.sol Line: 150

```
1    function setScoreSchemaUID(bytes32 _schemaUID) public onlyOwner
        {
```

- Found in contracts/GitcoinPassportDecoder.sol Line: 162

```
1    function setMaxScoreAge(uint64 _maxScoreAge) public onlyOwner {
```

- Found in contracts/GitcoinPassportDecoder.sol Line: 176

```
1    function setThreshold(uint256 _threshold) public onlyOwner {
```

- Found in contracts/GitcoinPassportDecoder.sol Line: 190

```
1    function addProviders(string[] memory providers) external
        onlyOwner {
```

- Found in contracts/GitcoinPassportDecoder.sol Line: 231

```
1    function createNewVersion(string[] memory providers) external
        onlyOwner {
```

Proof of Concept: Implement AccessControl functionality to define a DEFAULT_ADMIN_ROLE and switch the appropriate functions to only being able to be called by the admin. While leaving any fund management to the owner.

[L-2]: `public` functions not used internally could be marked `external`

Description: Instead of marking a function as `public`, consider marking it as `external` if it is not used internally. This will save on gas optimizations.

Impact:

10 Found Instances

- Found in contracts/GitcoinPassportDecoder.sol Line: 86

```
1    function initialize() public initializer {
```

- Found in contracts/GitcoinPassportDecoder.sol Line: 93

```
1    function pause() public onlyOwner {
```

- Found in contracts/GitcoinPassportDecoder.sol Line: 97

```
1    function unpause() public onlyOwner {
```

- Found in contracts/GitcoinPassportDecoder.sol Line: 106

```
1    function getProviders(uint32 version) public view returns (
        string[] memory) {
```

- Found in contracts/GitcoinPassportDecoder.sol Line: 138

```
1    function setPassportSchemaUID(bytes32 _schemaUID) public
        onlyOwner {
```

- Found in contracts/GitcoinPassportDecoder.sol Line: 150

```
1    function setScoreSchemaUID(bytes32 _schemaUID) public onlyOwner
        {
```

- Found in contracts/GitcoinPassportDecoder.sol Line: 162

```
1    function setMaxScoreAge(uint64 _maxScoreAge) public onlyOwner {
```

- Found in contracts/GitcoinPassportDecoder.sol Line: 176

```
1    function setThreshold(uint256 _threshold) public onlyOwner {
```

- Found in contracts/GitcoinPassportDecoder.sol Line: 463

```
1    function isHuman(address user) public view returns (bool) {
```

- Found in contracts/GitcoinPassportEligibility.sol Line: 63

```
1    function getWearerStatus(address wearer, uint256 /*_hatId*/ )
```

Proof of Concept: Example 1...Change `function initialize()public initializer` to `function initialize()external initializer`.

[L-3]: Empty Block

Description: There is an unused function which can be removed.

Impact:

1 Found Instances

- Found in contracts/GitcoinPassportDecoder.sol Line: 101

```
1    function _authorizeUpgrade(address) internal override onlyOwner
        {}
```

Proof of Concept: Add functionality to the code block.

[L-4]: Unused Custom Error

Description: There is an unused custom error that is taking up space.

Impact:

1 Found Instances

- Found in contracts/GitcoinPassportDecoder.sol Line: 75

```
1    error ScoreDoesNotMeetThreshold(uint256 score);
```

Proof of Concept: It is recommended that the definition be removed when custom error is unused.

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 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 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 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 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 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 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 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 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 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 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 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 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.