Hats Eligibility Module Report

Version 1.0

Jacob Homanics

May 27, 2024

Hats Module Report

Jacob Homanics

May 27, 2024

Prepared by: Jacob Homanics

Table of Contents

Protocol Summary

The Hats Modules are an extension of the core Hats Protocol to allow for eligiblity modules that unlock capabilities for the users. These are pretty abstract and allow for an unlimited number of use cases.

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:

```
9d650fc
```

Scope

```
HatsModule.sol
IHatsModule.sol
HatsEligibilityModule.sol
```

Roles

N/A

Executive Summary

There were only informational vulnerabilities that focused around making the protocol easier to use/understand. This is to be expected as Hats Modules are meant to be used as a template for others to implement custom functionality for.

The tools used were VSCode, Slither, and Aderyn.

Issues found

Severity	Number of issues found
High	0
Medium	0
Low	0
Informational	5
Gas	0
Total	5

Findings

Informational

[I-1] HatsModule::IMPLEMENTATION 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 `HatsModule::IMPLEMENTATION` 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 `HatsModule::IMPLEMENTATION` to `HatsModule::getImplementation` to satisfy the requirement of functions being in mixedCase.

[I-2] HatsModule::HATS 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 `HatsModule::HATS` 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 `HatsModule::HATS` to `HatsModule::getHats` to satisfy the requirement of functions being in mixedCase.

[I-3] HatsModule::setUp(bytes)._initData variable does not follow the mixedCase naming convention, resulting in potential confusion from code reviewers

Description: Underscores should not be used in to start variable names.

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 `HatsModule::setUp(bytes)._initData` to `HatsModule::setUp(bytes).initData` to satisfy the requirement of functions being in mixedCase.

[I-4] IHatsModule::IMPLEMENTATION 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 `IHatsModule::IMPLEMENTATION` 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 `IHatsModule::IMPLEMENTATION` to `IHatsModule::getImplementation` to satisfy the requirement of functions being in mixedCase.

[I-5] IHatsModule::HATS 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 `IHatsModule::HATS` 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 `IHatsModule::HATS` to `IHatsModule::getHats` to satisfy the requirement of functions being in mixedCase.