CERTIK

Aergo (HPP) - Custody Staking

Security Assessment

CertiK Assessed on Dec 2nd, 2025

CertiK Assessed on Dec 2nd, 2025

Aergo (HPP) - Custody Staking

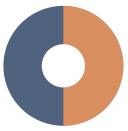
The security assessment was prepared by CertiK.

Executive Summary

TYPES	ECOSYSTEM	METHODS
Staking	EVM Compatible	Formal Verification, Manual Review, Static Analysis

LANGUAGE	TIMELINE
Solidity	Preliminary comments published on 12/02/2025 Final report published on 12/04/2025

Vulnerability Summary



2	1	0	1	0
Total Findings	Resolved	Partially Resolved	Acknowledged	Declined

🟧	1	Centralization	1 Acknowledged	Centralization findings highlight privileged roles & functions and their capabilities, or instances where the project takes custody of users' assets.
🟥	0	Critical		Critical risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.
🟧	0	Major		Major risks may include logical errors that, under specific circumstances, could result in fund losses or loss of project control.
🟨	0	Medium		Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
🟨	0	Minor		Minor risks can be any of the above, but on a smaller scale. They generally do not compromise the overall integrity of the project, but they may be less efficient than other solutions.
🟦	1	Informational	1 Resolved	Informational errors are often recommendations to improve the style of the code or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

TABLE OF CONTENTS | AERGO (HPP) - CUSTODY STAKING

CODEBASE | AERGO (HPP) - CUSTODY STAKING

Repository

https://github.com/hpp-io/contracts/blob/main/contracts/HPP_Custody_Staking.sol

https://github.com/hpp-io/contracts/commit/123f5838bcee7c6c13734b4ec30ff5ec2155af2b

Commit

123f5838bcee7c6c13734b4ec30ff5ec2155af2b

123f5838bcee7c6c13734b4ec30ff5ec2155af2b

AUDIT SCOPE | AERGO (HPP) - CUSTODY STAKING

hpp-io/contracts
HPP_Custody_Staking.sol

APPROACH & METHODS | AERGO (HPP) - CUSTODY STAKING

This audit was conducted for Aergo (HPP) to evaluate the security and correctness of the smart contracts associated with the Aergo (HPP) - Custody Staking project. The assessment included a comprehensive review of the in-scope smart contracts. The audit was performed using a combination of Static Analysis, Formal Verification, and Manual Review.

The review process emphasized the following areas:

- Architecture review and threat modeling to understand systemic risks and identify design-level flaws.
- Identification of vulnerabilities through both common and edge-case attack vectors.
- Manual verification of contract logic to ensure alignment with intended design and business requirements.
- Dynamic testing to validate runtime behavior and assess execution risks.
- Assessment of code quality and maintainability, including adherence to current best practices and industry standards.

The audit resulted in findings categorized across multiple severity levels, from informational to critical. To enhance the project's security and long-term robustness, we recommend addressing the identified issues and considering the following general improvements:

- Improve code readability and maintainability by adopting a clean architectural pattern and modular design.
- Strengthen testing coverage, including unit and integration tests for key functionalities and edge cases.
- Maintain meaningful inline comments and documentations.
- Implement clear and transparent documentation for privileged roles and sensitive protocol operations.
- Regularly review and simulate contract behavior against newly emerging attack vectors.

REVIEW NOTES | AERGO (HPP) - CUSTODY STAKING

Overview

The `HPPCustodyStaking` contract implements a staking mechanism designed to minimize on-chain asset risk by delegating asset holding to an external "Custody Wallet." Rather than retaining staked tokens within the contract, deposits are immediately forwarded to a designated custody address. Key features include:

- **Custodial Staking Flow:** Staking actions trigger an immediate transfer from the user to the contract, and subsequently to the custody wallet in the same transaction.
- **Queue-Based Unstaking:** Unstaking requests are placed into a cooldown queue. Users can have multiple concurrent cooldown requests, limited by `maxGlobalCooldownEntries` .
- **Allowance-Based Withdrawal:** Upon cooldown expiration, the contract pulls tokens back from the custody wallet (requiring the custody wallet to pre-approve the contract) and transfers them to the user.
- **Array Optimization:** Implements a "compaction" algorithm (`_compactCooldownArray`) using a sliding index pointer (`_firstValidIndex`) to manage gas costs associated with the cooldown queue.
- **Security Features:** Includes `Pausable` functionality, `ReentrancyGuard` on all state-changing external calls, and a two-step process for both ownership and custody wallet transfers.

External Dependencies

- **OpenZeppelin:** `IERC20` , `SafeERC20` , `Ownable2Step` , `Pausable` , `ReentrancyGuard` .
- **Token Compatibility:** The contract relies on standard ERC20 behavior.
- **Custody Wallet:** The system assumes the `custodyWallet` is an external address (EOA or Contract) capable of approving token allowances to the staking contract to facilitate withdrawals.

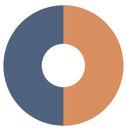
Privileged Functions

Functions callable by the `owner` (or the pending custody address) that manage configuration and critical addresses:

- `pause` / `unpause` : Freezes or resumes staking, unstaking, and withdrawing operations.
- `setCooldownDuration` : Updates the mandatory waiting period for new unstaking requests.
- `setMaxGlobalCooldownEntries` : Adjusts the limit on the number of simultaneous active cooldown requests a user can have.
- `proposeCustodyWallet` : Initiates the two-step process to change the address holding the funds.
- `acceptCustodyWallet` : Callable by the *pending* custody address to finalize the custody transfer.
- `rescueTokens` : Allows the owner to recover ERC20 tokens accidentally sent to the contract address.

Note: Since the contract forwards assets immediately, the safety of staked funds relies primarily on the security of the `custodyWallet` rather than the contract's internal storage. However, the owner controls the logic (pausing, cooldowns) that dictates user access to those funds.

FINDINGS | AERGO (HPP) - CUSTODY STAKING



2	0	1	0	0	0	1
Total Findings	Critical	Centralization	Major	Medium	Minor	Informational

This report has been prepared for Aergo (HPP) to identify potential vulnerabilities and security issues within the reviewed codebase. During the course of the audit, a total of 2 issues were identified. Leveraging a combination of Static Analysis, Formal Verification & Manual Review the following findings were uncovered:

ID	Title	Category	Severity	Status
AHC-01	**Centralization Risks In HPP_Custody_Staking.Sol**	**Centralization**	**Centralization**	● **Acknowledged**
AHC-02	Cooldown Array Never Trimmed Once All Entries Are Processed	Code Optimization	Informational	● Resolved

AHC-01 | Centralization Risks In HPP_Custody_Staking.Sol

Category	Severity	Location	Status
Centralization	● **Centralization**	**HPP_Custody_Staking.sol: 82~83, 86, 92, 104, 123, 130**	● **Acknowledged**

Description

In the contract `HPP_Custody_Staking.sol` the role `_owner` has authority over the functions shown `pause()` , `unpause()` , `setCooldownDuration()` , `setMaxGlobalCooldownEntries()` , `proposeCustodyWallet()` , `rescueTokens()` and `renouceOwnership()` . Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and pause/unpause contract, set contract global parameters or withdraw tokens held by this contract.

Recommendation

The audit team recommends that privileged operations controlled by the _owner in HPP_Custody_Staking.sol be protected through a phased approach to decentralization and improved key management. In the short term, implement time-locks on sensitive functions and transfer ownership to multi-signature wallets to reduce the risk of a single compromised key, and ensure privileged actions are publicly announced in a timely manner. For the longer term, introduce governance mechanisms such as a DAO or on-chain voting to transfer control from individuals to a community-driven process and increase transparency. Eventually, if possible, consider fully renouncing ownership or disabling high-risk functions to eliminate central points of control.

Alleviation

[Aergo (HPP), 12/03/2025]: The team acknowledged the issue and decided not to implement the recommended change in the current engagement.

AHC-02 | Cooldown Array Never Trimmed Once All Entries Are Processed

Category	Severity	Location	Status
Code Optimization	● Informational	HPP_Custody_Staking.sol: 238	● Resolved

Description

`_compactCooldownArray` exits early when `firstIdx >= list.length`, which is exactly the state after a user withdraws every pending cooldown. This prevents the helper from removing the stale entries, so the storage array only ever grows.

```
//233:239:contracts/HPP_Custody_Staking.sol/contracts/HPP_Custody_Staking.sol
function _compactCooldownArray(address user) internal {
    Cooldown[] storage list = _cooldowns[user];
    uint256 firstIdx = _firstValidIndex[user];
    uint256 n = list.length;

    if (firstIdx == 0 || firstIdx >= n) return;
```

Recommendation

Allow the function to run when `firstIdx == n` by clearing the array (e.g., `delete _cooldowns[user]`) and resetting `firstIdx`. This keeps per-user storage bounded and avoids growing state after every full withdrawal, even though correctness is unaffected today.

Alleviation

[Aergo (HPP), 12/03/2025]: The team heeded the advice and resolved the issue in commit
123f5838bcee7c6c13734b4ec30ff5ec2155af2b

FORMAL VERIFICATION | AERGO (HPP) - CUSTODY STAKING

Formal guarantees about the behavior of smart contracts can be obtained by reasoning about properties relating to the entire contract (e.g. contract invariants) or to specific functions of the contract. Once such properties are proven to be valid, they guarantee that the contract behaves as specified by the property. As part of this audit, we applied formal verification to prove that important functions in the smart contracts adhere to their expected behaviors.

Considered Functions And Scope

In the following, we provide a description of the properties that have been used in this audit. They are grouped according to the type of contract they apply to.

Verification of Ownable2step Properties

We verified *partial* properties of the public interfaces of those token contracts that implement the Ownable2Step interface. This involves:

- function `owner` that returns the current owner,
- functions `renounceOwnership` that removes ownership,
- function `transferOwnership` that transfers the ownership to a new owner,
- function `pendingOwner` that return the pending owner, and
- function `acceptOwnership` by which a pending owner can accept the ownership.

The properties that were considered within the scope of this audit are as follows:

Property Name	Title
ownable-renounceownership-correct	Ownership is Removed
ownable2step-acceptownership-succeed-normal	`acceptOwnership` Succeeds For Valid Inputs
ownable2step-pendingowner-succeed-normal	`pendingOwner` Always Succeeds
ownable2step-acceptownership-revert	`acceptOwnership` Prevents Invalid Inputs
ownable-transferownership-correct	Ownership is Transferred
ownable-owner-succeed-normal	`owner` Always Succeeds

Verification Results

In the remainder of this section, we list all contracts where formal verification of at least one property was not successful. There are several reasons why this could happen:

- False: The property is violated by the project.
- Inconclusive: The proof engine cannot prove or disprove the property due to timeouts or exceptions.

- Inapplicable: The property does not apply to the project.

Detailed Results For Contract HPPCustodyStaking (contracts/HPP_Custody_Staking.sol) In Commit 123f5838bcee7c6c13734b4ec30ff5ec2155af2b

Verification of Ownable2step Properties

Detailed Results for Function `renounceOwnership`

Property Name	Final Result	Remarks
ownable-renounceownership-correct	● True	

Detailed Results for Function `acceptOwnership`

Property Name	Final Result	Remarks
ownable2step-acceptownership-succeed-normal	● True	
ownable2step-acceptownership-revert	● True	

Detailed Results for Function `pendingOwner`

Property Name	Final Result	Remarks
ownable2step-pendingowner-succeed-normal	● True	

Detailed Results for Function `transferOwnership`

Property Name	Final Result	Remarks
ownable-transferownership-correct	● False	

Detailed Results for Function `owner`

Property Name	Final Result	Remarks
ownable-owner-succeed-normal	● True	

APPENDIX | AERGO (HPP) - CUSTODY STAKING

Finding Categories

Categories	Description
Centralization	Centralization findings detail the design choices of designating privileged roles or other centralized controls over the code.

Details on Formal Verification

Some Solidity smart contracts from this project have been formally verified. Each such contract was compiled into a mathematical model that reflects all its possible behaviors with respect to the property. The model takes into account the semantics of the Solidity instructions found in the contract. All verification results that we report are based on that model.

The following assumptions and simplifications apply to our model:

- Certain low-level calls and inline assembly are not supported and may lead to a contract not being formally verified.
- We model the semantics of the Solidity source code and not the semantics of the EVM bytecode in a compiled contract.

Formalism for property specifications

All properties are expressed in a behavioral interface specification language that CertiK has developed for Solidity, which allows us to specify the behavior of each function in terms of the contract state and its parameters and return values, as well as contract properties that are maintained by every observable state transition. Observable state transitions occur when the contract's external interface is invoked and the invocation does not revert, and when the contract's Ether balance is changed by the EVM due to another contract's "self-destruct" invocation. The specification language has the usual Boolean connectives, as well as the operator `\old` (used to denote the state of a variable before a state transition), and several types of specification clause:

Apart from the Boolean connectives and the modal operators "always" (written `[]`) and "eventually" (written `<>`), we use the following predicates to reason about the validity of atomic propositions. They are evaluated on the contract's state whenever a discrete time step occurs:

- `requires [cond]` - the condition `cond`, which refers to a function's parameters, return values, and contract state variables, must hold when a function is invoked in order for it to exhibit a specified behavior.
- `ensures [cond]` - the condition `cond`, which refers to a function's parameters, return values, and both `\old` and current contract state variables, is guaranteed to hold when a function returns if the corresponding requires condition held when it was invoked.
- `invariant [cond]` - the condition `cond`, which refers only to contract state variables, is guaranteed to hold at every observable contract state.

- `constraint [cond]` - the condition `cond` , which refers to both `\old` and current contract state variables, is guaranteed to hold at every observable contract state except for the initial state after construction (because there is no previous state); constraints are used to restrict how contract state can change over time.

Description of the Analyzed Ownable2Step Properties

Properties related to function `renounceOwnership`

ownable-renounceownership-correct

Invocations of `renounceOwnership()` must set ownership to address(0).

Specification:

```
ensures this.owner() == address(0);
```

Properties related to function `acceptOwnership`

ownable2step-acceptownership-revert

Function `acceptOwnership` must always fail if called by a sender that is not the pending owner.

Specification:

```
reverts_when msg.sender != this.pendingOwner();
```

ownable2step-acceptownership-succeed-normal

Function `acceptOwnership` always succeeds if it is called by the pending owner.

Specification:

```
reverts_only_when msg.sender != this.pendingOwner();
```

Properties related to function `pendingOwner`

ownable2step-pendingowner-succeed-normal

Function `pendingOwner` must always succeed if it does not run out of gas.

Specification:

```
reverts_only_when false;
```

Properties related to function `transferOwnership`

ownable-transferownership-correct

Invocations of `transferOwnership(newOwner)` must transfer the ownership to the `newOwner`.

Specification:

```
ensures this.owner() == newOwner;
```

Properties related to function `owner`

ownable-owner-succeed-normal

Function `owner` must always succeed if it does not run out of gas.

Specification:

```
reverts_only_when false;
```

DISCLAIMER | CERTIK

UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

Elevating Your Web3 Journey

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is the largest blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we’re able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.