



**ANASTASIA LABS**

# Security Audit Report

Business Confidential

Date: March 7, 2024  
Project: Pondora Core  
Version 1.0

# Contents

<b>Confidentiality statement</b>	<b>3</b>
<b>Scope and Disclaimer</b>	<b>4</b>
<b>Assessment overview</b>	<b>5</b>
<b>Assessment components</b>	<b>6</b>
<b>Executive summary</b>	<b>7</b>
<b>Code base without staking validator</b>	<b>8</b>
Repository . . . . .	8
Commit . . . . .	8
Files audited . . . . .	8
<b>Code base including staking validator</b>	<b>9</b>
Repository . . . . .	9
Commit . . . . .	9
Files audited . . . . .	9
<b>Finding severity ratings</b>	<b>10</b>
<b>Findings</b>	<b>11</b>
ID-101 Using list.any instead of list.find . . . . .	12
ID-102 Using the staking validator instead of minting policies for modules . . . . .	13

# Confidentiality statement

This document is the exclusive property of Anastasia Labs and Pondora . This document contains proprietary and confidential information. Duplication, redistribution, or use, in whole or in part, in any form, requires consent of both Anastasia Labs and Pondora .

Pondora may share this document with third parties under non-disclosure agreements to demonstrate security compliance.

## Scope and Disclaimer

The scope of this audit is limited to the on-chain code specified in the files audited section. The code responsible for off-chain transaction building was not reviewed in this audit.

Aiken, the language that was used to write the Pondora Core smart contracts is relatively new; as such, the UPLC that it generates may not correctly correspond to the intentions of the Aiken code. The correctness of the Aiken compilation pipeline and the Aiken standard library are both out of the scope of this audit.

Anastasia Labs prioritized identifying the weakest security vectors as well as prominent areas where code quality can be improved.

The scope of the audit did not include the creation of additional unit or property-based testing of the contracts.

The findings and recommendations contained herein reflect the information gathered by Anastasia Labs during the course of the assessment, and exclude any changes or modifications made outside of that period.

### Protocol optimization

Anastasia Labs undertook a thorough examination of the protocol by auditing two distinct versions of the code base. This comprehensive approach was adopted to accommodate recent improvements and code restructuring. The audited versions include:

- Code base without the staking validator:

This version represents the software prior to the integration of the staking validator component. It serves as a benchmark to assess the security and functionality of the initial implementation.

- Code base including the staking validator:

This version incorporates the staking validator component into the software architecture. The addition of the staking validator introduces enhanced functionality and security measures to the system.

## Assessment overview

From January 6th, 2024 to February 29th, 2024, Pondora engaged Anastasia Labs to evaluate and conduct a security assessment of its Pondora Core protocol. All code revision was performed following industry best practices.

Phases of code auditing activities include the following:

- Planning – Customer goals are gathered.
- Discovery – Perform code review to identify potential vulnerabilities, weak areas, and exploits.
- Attack – Confirm potential vulnerabilities through testing and perform additional discovery upon new access.
- Reporting – Document all found vulnerabilities.

The engineering team has also conducted a comprehensive review of protocol optimization strategies.

# Assessment components

## Manual revision

Our manual code auditing is focused on a wide range of attack vectors, including but not limited to.

- UTXO Value Size Spam (Token Dust Attack)
- Large Datum or Unbounded Protocol Datum
- EUTXO Concurrency DoS
- Unauthorized Data modification
- Multisig PK Attack
- Infinite Mint
- Incorrect Parameterized Scripts
- Other Redeemer
- Other Token Name
- Arbitrary UTXO Datum
- Unbounded protocol value
- Foreign UTXO tokens
- Double or Multiple satisfaction
- Locked Ada
- Locked non Ada values
- Missing UTXO authentication
- UTXO contention

## Executive summary

The Pondora Protocol represents a novel approach to smart contract architecture that aims to maximize both user experience and capital efficiency. By utilizing separate outputs for assets and the rules governing their spending, an asset owner may update these rules in a modular fashion. The protocol consist of two components:

- Pondora Core - This allows asset owners to lock their asset(s) in a smart contract and associate any number of unlock conditions to each unique output, with each condition represented by a specific Pondora Module.
- Pondora Modules - Each module is a separate smart contract that implements a specific type of exchange condition. For example, trading and lending are separate modules, allowing an owner to opt in to each feature per individual listing.

This component structure provides the following functionality:

- To set multiple trade conditions, such as asking for different tokens and/or quantities in exchange for the same asset(s).
- To maximize capital efficiency by utilizing the same asset(s) across multiple modules simultaneously. For example, they may use the same tokens in an offer to purchase a specific NFT or in an offer to fund a loan of specific terms, at the same time.
- To support an unprecedented level of bulk operation by allowing thousands of these conditions to be efficiently updated within a single transaction, resulting in massive network fee and time savings.

# Code base without staking validator

## Repository

<https://github.com/pondora-org/pondora-contracts>

## Commit

8b44bd3da651f6a27c422ad735b3128c916dd952

## Protocol component

Pondora Core

## Files audited

SHA256 Checksum	Files
8bf274b9f006b0d806947404f74921ec99ef6bfab34002be5387feee20a081bd	validators/conditions.ak
7359e21a4106f5a96cfe829d0503657d122e2459d2837e98d1cbb6a3919101f0	validators/listing.ak
1c1cb2bbf23b4109f42d77151ff73f6afa485e49b8765580630034c76ab40f4a	lib/types.ak
8de1ff3732bcfbfe73cdcb991861302297ab21d62daa603b091e0b188707dcf6	lib/validation/conditions.ak
87abd72fcff627a6808f84c128d1db77a46c45ffbbe5ea90b44a68f15badf9c2	lib/validation/listing_spend.ak

# Code base including staking validator

## Repository

<https://github.com/pondora-org/pondora-contracts>

## Commit

c9a4268578b22c3419d4dc18b2caea2191e344ce

## Protocol component

Pondora Core

## Files audited

SHA256 Checksum	Files
644bfb76459080a7f824e24eab310b4467e6a86d306befc4070061019e27917b	validators/conditions.ak
a2c6c50cc8b8179db0c51e5a32618796ec497721d57cada963396c3a2677ecf2	validators/listing.ak
ce61340efbbdef018d08e0eec27901e0378c5c5988f2faade97b1fb6b0174b5a	lib/exo/types.ak
e424bef6bd3a218a2c130273c4f1dd8641914306cfea7d915ce50eb636159915	lib/exo/utils.ak

# Finding severity ratings

The following table defines levels of severity and score range that are used throughout the document to assess vulnerability and risk impact.

## Findings for code base without staking validator

	Level	Severity	Findings
	5	Critical	0
	4	Major	0
	3	Medium	0
	2	Minor	0
	1	Informational	2

## Findings for code base including staking validator

	Level	Severity	Findings
	5	Critical	0
	4	Major	0
	3	Medium	0
	2	Minor	0
	1	Informational	0

# Findings

## ID-101 Using `list.any` instead of `list.find`

	Level	Severity	Status
	1	Informational	Resolved

### Code base

This finding refers to the **code base without staking validator**:

Repository	Commit
<a href="https://github.com/pondora-org/pondora-contracts">https://github.com/pondora-org/pondora-contracts</a>	8b44bd3da651f6a27c422ad735b3128c916dd952

### Description

The function `validate_listing_spend` currently utilizes the function **`list.find`** to locate a valid module token that is allocated to the output fee address. In this context, the function returns the first element that satisfies the specified predicate.

```
expect Some(_valid_fee_output) =  
list.find(  
  outputs,  
  fn(output) {  
    if output.address != fee_address {  
      False  
    } else {  
      ...  
    }  
  }  
)
```

### Recommendation

While both **`list.any`** and **`list.find`** serve a similar purpose, we recommend utilizing **`list.any`** instead due to its more efficient CPU and memory utilization. Additionally, **`list.find`** returns an option type, which necessitates pattern matching on **`expect`**, potentially incurring higher overhead costs overall.

### Resolution

Resolved

## ID-102 Using the staking validator instead of minting policies for modules

Level	Severity	Status
1	Informational	Resolved

### Code base

This finding refers to the **code base without staking validator**:

Repository	Commit
<a href="https://github.com/pondora-org/pondora-contracts">https://github.com/pondora-org/pondora-contracts</a>	8b44bd3da651f6a27c422ad735b3128c916dd952

### Description

The current implementation relies on minting policies for modules, which are used to validate the list spending validator. However, a drawback of these modules is that for every execution, they necessitate minting a garbage token that must be directed to a fee address. This leads to an unnecessary utilization of the minting policy.

### Recommendation

We strongly recommend adopting the staking validator (withdraw zero trick) over minting policies for modules. This solution entails the Spending Validator verifying that the Staking Validator is invoked within the same transaction, thereby consolidating the logic to execute once at the Staking Validator. This approach significantly reduces script size and simplifies business logic.

Staking Validators are pivotal, not only for enhancing stake control logic but also for minimizing script size and optimizing CPU and memory usage.

This protocol design approach should be applied to both listing and module contracts. The listing validator must validate the presence of the withdrawal stakehash in the transaction, while the module must ensure that the listing UTXO is spent according to the prescribed business logic

### Resolution

Resolved