ackee blockchain

IPOR

IporToken

by Ackee Blockchain

21.11.2022

Contents

1. Document Revisions

1.0	Final report	November 9, 2022
1.1	Fix review report	November 21, 2022

2. Overview

This document presents our findings in reviewed contracts.

2.1. Ackee Blockchain

Ackee Blockchain is an auditing company based in Prague, Czech Republic, specializing in audits and security assessments. Our mission is to build a stronger blockchain community by sharing knowledge – we run free certification courses School of Solana, Summer School of Solidity and teach at the Czech Technical University in Prague. Ackee Blockchain is backed by the largest VC fund focused on blockchain and DeFi in Europe, Rockaway Blockchain Fund.

2.2. Audit Methodology

1. **Technical specification/documentation** - a brief overview of the system is requested from the client and the scope of the audit is defined.
2. **Tool-based analysis** - deep check with automated Solidity analysis tools and Woke is performed.
3. **Manual code review** - the code is checked line by line for common vulnerabilities, code duplication, best practices and the code architecture is reviewed.
4. **Local deployment + hacking** - the contracts are deployed locally and we try to attack the system and break it.
5. **Unit and fuzzy testing** - run unit tests to ensure that the system works as expected, potentially write missing unit or fuzzy tests.

2.3. Finding classification

A *Severity* rating of each finding is determined as a synthesis of two sub-ratings: *Impact* and *Likelihood*. It ranges from *Informational* to *Critical*.

If we have found a scenario in which an issue is exploitable, it will be assigned an impact rating of *High*, *Medium*, or *Low*, based on the direness of the consequences it has on the system. If we haven't found a way, or the issue is only exploitable given a change in configuration (such as deployment scripts, compiler configuration, use of multi-signature wallets for owners, etc.) or given a change in the codebase, then it will be assigned an impact rating of *Warning* or *Info*.

Low to *High* impact issues also have a *Likelihood*, which measures the probability of exploitability during runtime.

The full definitions are as follows:

Severity

		Likelihood			
		High	**Medium**	**Low**	**-**
Impact	**High**	Critical	High	Medium	-
	Medium	High	Medium	Medium	-
	Low	Medium	Medium	Low	-
	Warning	-	-	-	Warning
	Info	-	-	-	Info

Table 1. Severity of findings

Impact

- **High** - Code that activates the issue will lead to undefined or catastrophic consequences for the system.
- **Medium** - Code that activates the issue will result in consequences of serious substance.
- **Low** - Code that activates the issue will have outcomes on the system that are either recoverable or don't jeopardize its regular functioning.
- **Warning** - The issue cannot be exploited given the current code and/or configuration (such as deployment scripts, compiler configuration, use of multi-signature wallets for owners, etc.), but could be a security vulnerability if these were to change slightly. If we haven't found a way to exploit the issue given the time constraints, it might be marked as a "Warning" or higher, based on our best estimate of whether it is currently exploitable.
- **Info** - The issue is on the borderline between code quality and security. Examples include insufficient logging for critical operations. Another example is that the issue would be security-related if code or configuration (see above) was to change.

Likelihood

- **High** - The issue is exploitable by virtually anyone under virtually any circumstance.
- **Medium** - Exploiting the issue currently requires non-trivial preconditions.
- **Low** - Exploiting the issue requires strict preconditions.

2.4. Review team

Member's Name	Position
Lukáš Böhm	Lead Auditor
Miroslav Škrabal	Auditor
Josef Gattermayer, Ph.D.	Audit Supervisor

2.5. Disclaimer

We've put our best effort to find all vulnerabilities in the system, however our findings shouldn't be considered as a complete list of all existing issues. The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them.

3. Executive Summary

IPOR (Inter-Protocol Offered Rate) protocol works as a weighted index average of several different borrowing and lending sources. Handling and selecting the most relevant sources would be done via IPOR Decentralized Autonomous Organization (DAO) to achieve a complete decentralized system. The transparent mathematical formulas calculate a weighted average.

Revision 1.0

IPOR team engaged Ackee Blockchain to perform a security review of the Ipor protocol parts, specifically `IporToken` and Ipor mining (`John` and `PowerIpor` contracts), within a period between October 17 and November 9, 2022 and the lead auditor was Lukáš Böhm. The audit has been performed on the commit `01c08c3`. At the client's request, the report was divided into two parts. This report covers `IporToken` contract only.

We began our review using static analysis tools, namely Slither and Woke. We then took a deep dive into the logic of the contracts. During the review, we paid particular attention to:

- ensuring the arithmetic of the system is correct,
- detecting possible reentrancies in the code,
- ensuring access controls are not too relaxed or too strict,
- looking for common issues such as data validation,
- ensuring the token handling logic is correct.

The protocol architecture is well-designed, and the code quality is above average. The code lacks in-code NatSpec documentation for some functions. Nevertheless, IPOR team provides high-quality documentation for the protocol and its components. This is very appreciated, especially for contracts with a mathematical reward logic. The team also provides a helpful diagram to understand liquidity mining mechanics better. We encourage the team to continue with a professional approach to make the project transparent, understandable, and easy to use. The project contains thousands of lines of TypeScript tests with excellent code coverage.

Our review resulted in 2 findings, ranging from Info to Warning severity. In the protocol, no actual thread has been found, and most issues are about the code performance and quality.

Ackee Blockchain recommends IPOR:

- improve the code quality by adding NatSpec documentation,
- pay more attention to the code performance and gas usage,
- address all reported issues.

See Revision 1.0 for the system overview of the codebase.

Revision 1.1

The fix review was done on November 21 on the given commit: `a1a3657` in a public repository.

See Revision 1.1 for the review of the updated codebase and additional information we consider essential for the current scope.

The status of all reported issues has been updated and can be seen in the findings table. The acknowledged issue contains the client’s comments.

4. Summary of Findings

The following table summarizes the findings we identified during our review. Unless overridden for purposes of readability, each finding contains:

- a *Description*,
- an *Exploit scenario*,
- a *Recommendation* and if applicable
- a *Solution*.

There might often be multiple ways to solve or alleviate the issue, with varying requirements regarding the necessary changes to the codebase. In that case, we will try to enumerate them all, clarifying which solves the underlying issue better (albeit possibly only with architectural changes) than others.

	Severity	Reported	Status
W1: Usage of `solc` optimizer	Warning	1.0	Acknowledged
I1: Redundant inheritance of Ownable	Info	1.0	Fixed

Table 2. Table of Findings

5. Report revision 1.0

5.1. System Overview

This section contains an outline of the audited contracts. Note that this is meant for understandability purposes and does not replace project documentation.

Contracts

Contracts we find important for better understanding are described in the following section.

IporToken

The contract inherits form `ERC20` standard. It contains view function `getContractId()` for a safe multi-contract composition and constructor that mints specific number of tokens to the DAO address.

Actors

This part describes actors of the system, their roles, and permissions.

User

Users can interact with the contract as with standard ERC20 tokens.

5.2. Trust model

In the current scope, there is no role with a special privilege over the contract, that could violate the system's trust model.

W1: Usage of `solc` optimizer

Impact:	Warning	Likelihood:	N/A
Target:	`**/*`	Type:	Compiler configuration

Description

The project uses `solc` optimizer. Enabling `solc` optimizer may lead to unexpected bugs.

The Solidity compiler was audited in November 2018, and the audit concluded that the optimizer may not be safe.

Vulnerability scenario

A few months after deployment, a vulnerability is discovered in the optimizer. As a result, it is possible to attack the protocol.

Recommendation

Until the `solc` optimizer undergoes more stringent security analysis, opt-out using it. This will ensure the protocol is resilient to any existing bugs in the optimizer.

Fix 1.1

Currently we are using optimizer in already deployed IPOR Protocol smart contracts. Liquidity Mining is a part of IPOR Protocol so will be part of public repo ipor-protocol where we are using optimizer. We will monitor future issues related with Optimizer.

Go back to Findings Summary

I1: Redundant inheritance of Ownable

Impact:	Info	Likelihood:	N/A
Target:	IporToken.sol	Type:	Dead code

Description

The `IporToken` inherits from `IporOwnable`. However, it does not use any functionality the Ownable pattern provides. As such, it only adds unnecessary complexity to the code and lowers the readability, implying that the IporToken is somehow ownable.

Recommendation

Remove the inheritance of `IporOwnable` in the `IporToken` contract.

Fix 1.1

Redundant inheritance was removed from a contract.

Go back to Findings Summary

6. Report revision 1.1

No significant changes were performed in the contract, and no new vulnerabilities were found. One reported issue was fixed, and the second one was acknowledged.

Appendix A: How to cite

Please cite this document as:

Ackee Blockchain, IPOR: IporToken, 21.11.2022.

Appendix B: Glossary of terms

The following terms might be used throughout the document:

Superclass/Ancestor of C

A contract that C inherits/derives from.

Subclass/Child of C

A contract that inherits/derives from C.

Syntactic contract

A Solidity contract. May have an inheritance chain, and may be deployed.

Deployed contract

An EVM account with non-zero code. If its source was written in Solidity, it was created through at least one syntactic contract. If that contract had superclasses (parents), it would be composed of multiple syntactic contracts.

Init/initialization function

A non-constructor function that serves as an initializer. Often used in upgradeable contracts.

External entrypoint

A `public` or `external` function.

Public/Publicly-accessible function/entrypoint

An `external` or `public` function that can be successfully executed by any network account.

Mutating function

A non-`view` and non-`pure` function.

ackee

Thank You

Ackee Blockchain a.s.

Prague, Czech Republic

hello@ackeeblockchain.com

https://discord.gg/z4KDUbuPxq