Report for:

Akropolis

November 2021

Version: 1.0

Prepared By: Extropy.IO
Email: info@extropy.io
Telephone: +44 1865261424

Table of Contents

1. Executive Summary

Extropy was contracted to conduct a code review and vulnerability assessment of the Akropolis project
This report presents the findings of that audit, conducted between 10/11/21 and 23/11/2021.

1.1. Assessment Summary

In general the contracts are well designed and implemented. Best practices for smart contract development have been mostly followed. We also include some advice related to questions raised in the audit notes.
This report highlights a number of issues that we recommend be addressed.

Phase	Description	Critical	High	Medium	Low	Info	Total
1	Initial Audit	0	0	1	1	5	7

2. Using This Report

To facilitate the dissemination of the information within this report throughout your organisation, this document has been divided into the following clearly marked and separable sections.

Executive Summary	Management level, strategic overview of the assessment and the risks posed to the business
Technical Summary	An overview of the assessment from a more technical perspective, including a defined scope and any caveats which may apply
Technical Findings	Detailed discussion (including evidence and recommendations) for each individual security issue which was identified
Methodologies	Audit process and tools used

2.1. Disclaimer

The audit makes no statements or warranty about utility of the code, safety of the code, suitability of the business model, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only

2.2. Client Confidentiality

This document contains Client Confidential information and may not be copied without written permission.

2.3. Proprietary Information

The content of this document should be considered proprietary information. Extropy gives permission to copy this report for the purposes of disseminating information within your organisation or any regulatory agency.

Document Version Control			
Data Classification	Client Confidential		
Client Name	Akropolis		
Document Title	Akropolis Audit		
Author	Extropy Audit Team		

Document History			
Issue No.	**Issue Date**	**Issued By**	**Change Description**
1.0	24/11/2021	Laurence Kirk	Released to client

3. Technical Summary

3.1. Scope

This audit is of code at commit b0831d5e2cbc14e0492ae3d95b9545357c216680

4. Technical Findings

The remainder of this document is technical in nature and provides additional detail about the items already discussed, for the purposes of remediation and risk assessment.

.

5. Issues Found

5.1. Contract's code size exceeds 24576 bytes

Risk Rating	Medium

Affects : BasisStrategy

Description:

Contract code size exceeds 24,576 bytes (a limit introduced in Spurious Dragon). The contract's size is 35.83 KiB.

This contract may not be deployable on mainnet. Consider enabling the optimizer turning off revert strings, or move some functionality to libraries.

5.2. Unused Variable

Risk Rating	Low

Affects : BasisStrategy

Description:

Variable `newFunds` is assigned but never used.

uint256 newFunds = vault.update(amount, loss); (line 361)

Remediation
Call the Vault's update function, just don't store the return value, i.e. the amount of underlier token that was just sent from the Vault to the Strategy. Double-check that this value should not be used anywhere before removing it.

5.3. Use Constant

Risk Rating	Informational

Affects : BasisStrategy

Description:

require(_buffer < 1_000_000, "!_buffer"); (line 229)

Remediation
should use the existing `MAX_BPS` constant instead of the hardcoded `1_000_000` value
require(_buffer < MAX_BPS, "!_buffer");

5.4. Events Access

Risk Rating	Informational

Affects : BasisStrategy

Description:

Should emit an event for `setGovernance` (line 287)

Remediation
Emit an event in the function

5.5. External call

Risk Rating	Informational

Affects : BasisStrategy

Description:

Function `snapshot()` on line 589 is marked `public` and has no access control, inside this function, `forceToSyncState()` is called on the MCDEX LiquidityPool contract.

This function could be marked as external and it should call the existing `getMarginAccount()` function rather than duplicating the code on line 601.

5.6. Unused Internal Function

Risk Rating	Informational

Affects : BasisStrategy

Description:

Function `_swapTokenOut` on line 906 has no calling function and it is not marked `virtual`, meaning it is not intended for inheritance either.

Remediation

Remove the function if not needed.

5.7. Split Withdraw Function

Risk Rating	Informational

Affects : BasisVault

Description:

The withdraw function could be broken down further to take the removal of funds from strategy out and in a separate function.
Rebalancing the Vault when funds are removed from the strategy may need to be considered.

5.8. Additional Notes

Basis Vault Contract
No issues were found in this contract, the share values are calculated correctly both during deposits and withdrawals.

Questions raised in audit notes
Although no specific vulnerabilities were found, to mitigate against flash loan attacks, there are some standard approaches, a cap can be placed on the size of a trade, also a mechanism can be introduced to prevent for example deposit and withdrawals by the same account in one transaction.
Regarding oracle manipulation, a TWAP oracle is highly resistant to oracle manipulation attacks. However, due to the nature of its implementation (time weighted), it may not respond quickly enough to moments of high market.

6. Tool List

The following tools were used during the assessment:

Tools Used	Description	Resources
SWC Registry	Vulnerability database	https://swcregistry.io/

7. General Audit Goals

We audit the code in accordance with the following criteria:

Sound Architecture

This audit includes assessments of the overall architecture and design choices. Given the subjective nature of these assessments, it will be up to the development team to determine whether any changes should be made.

Smart Contract and Rust Best Practices

This audit will evaluate whether the codebase follows the current established best practices for smart contract development.

Code Correctness

This audit will evaluate whether the code does what it is intended to do.

Code Quality

This audit will evaluate whether the code has been written in a way that ensures readability and maintainability.

Security

This audit will look for any exploitable security vulnerabilities, or other potential threats to the users.

Although we have commented on the application design, issues of crypto-economics, game theory and suitability for business purposes as they relate to this project are beyond the scope of this audit.