Babylon Security Analysis

by Pessimistic

This report is public

February 22, 2022

Abstract

In this report, we consider the security of smart contracts of Babylon project. Our task is to find and describe security issues in the smart contracts of the platform.

Disclaimer

Summary

In this report, we considered the security of Babylon smart contracts. We performed our audit according to the procedure described below.

The audit showed an issue with test and several issues of low severity. The overall code quality is good.

The project has a documentation. However, we consider it insufficient.

The scope of the audit is limited, which might negatively affect its overall effectiveness.

General recommendations

We recommend improving the documentation for the project.

Project overview

Project description

For the audit, we were provided with Babylon project on a private GitHub repository, commit 3fd77c43cd5ba8dcac2d940ddc0390b94f5d01ab.

The scope of the audit included only:

- **strategies/Strategy.sol** file and its dependencies.
- **gardens/Garden.sol** file and its dependencies.

The documentation for the project was provided as a link. However, we consider this documentation insufficient.

One test out of 605 does not pass, the code coverage of the scope is 93.75%.

The total LOC of audited sources is 2706.

Procedure

In our audit, we consider the following crucial features of the code:

1. Whether the code is secure.
2. Whether the code corresponds to the documentation (including whitepaper).
3. Whether the code meets best practices.

We perform our audit according to the following procedure:

- Automated analysis
    - We scan the project’s codebase with the automated tool SmartCheck.
    - We manually verify (reject or confirm) all the issues found by the tool.
- Manual audit
    - We manually analyze the codebase for security vulnerabilities.
    - We assess the overall project structure and quality.
- Report
    - We reflect all the gathered information in the report.

Manual analysis

The contracts were completely manually analyzed, their logic was checked. Besides, the results of the automated analysis were manually verified. All the confirmed issues are described below.

Critical issues

Critical issues seriously endanger project security. They can lead to loss of funds or other catastrophic consequences. The contracts should not be deployed before these issues are fixed.

The audit showed no critical issues.

Medium severity issues

Medium issues can influence project operation in the current implementation. Bugs, loss of potential income, and other non-critical failures fall into this category, as well as potential problems related to incorrect system management. We highly recommend addressing them.

Insufficient documentation

The project has a documentation, the codebase is covered with additional comments. However, some crucial details are not described, e.g., whether strategies can use the whole provided capital, how profits are calculated, how contracts interact with each other, etc.

The documentation is a critical part that helps to improve security and reduce risks. It should explicitly explain the purpose and behavior of the contracts, their interactions, and main design choices. It is also essential for any further integrations.

The developers confirmed that they plan to improve the documentation.

Tests issues

604 tests out of 605 pass successfully. However, one test fails.

The team reported code coverage to be over 90%. We could not reproduce the results.

We recommend always running the tests and estimating code coverage. All tests should pass, and they should cover at least main scenarios.

Low severity issues

Low severity issues do not directly affect project operation. However, they might lead to various problems in future versions of the code. We recommend fixing them or explaining why the team has chosen a particular option.

Code quality

- In **Strategy** contract, there is a hardcoded address `0xccE114848A694152Ba45a8caff440Fcb12f73862` at line 124, which is against best practices. We recommend replacing it with a variable and assigning its value via a constructor, initializer, or setter method.

  > *Comment from the developers: We agree and we are slowly moving them to immutables.*

- In **Garden** contract, the check at line 918 is redundant since expression at lines 915–917 ensures that the condition is always hold.

- Consider declaring functions as `external` instead of `public` to improve code readability and optimize gas consumption in:
  - **Garden** contract at line 578.
  - **Strategy** contract at line 580.

- Variables `maxAllocationPercentage` and `maxGasFeePercentage` declared in **Strategy** contract at lines 221–223 are unused.

  > *Comment from the developers: They are not used in the Solidity code directly but they are used by off-chain actors to view information and make decision and caclucaltion around capital allocation for strategies.*

Notes

The scope of the audit

The goal of the audit is to assess the security of **Garden** and **Strategy** contracts and their dependencies. However, they interact with **Operation**, **MasterSwapper**, **GardenValuer**, **RewardsDistributor** and **StrategyNFT** contracts that lie outside the scope of the audit. It limits the overall effectiveness of the audit since a partial review of the codebase might miss some integration issues.

Besides, this audit does not cover other parts of the project's contract system. We recommend allocating additional resources to secure the whole codebase.

Unchecked integrations

The system does not check integration addresses, tokens used by these integrations, etc. It is critically important to verify these integrations since they can call functions like `invokeApprove`, `invokeApproveFromIntegration`, etc.

According to the documentation, the system integrates with protocols from this list.

Overpowered roles

The project has special roles with specific powers that the system depends on heavily.

The owner of the **Controller** contract can:

- Delete candidates from strategies.
- Change `duration` and `maxTradeSlippagePercentage` parameters of strategies.
- Add new Keepers.

Keepers can:

- Control the execution process of strategies by calling proper methods.
- Provide (and thus manipulate) the result of offline votings for new strategies.

The Creator of the Garden can:

- Change the Creator of the Garden.
- Change parameters of the Garden.

> *Comment from the developers: The system needs keepers to operate so it is very important they are not malicious. That is the reason to have a DAO to control and own Controller (already under decentralized governance).*

Incorrect unwind amount calculation

In **Strategy** contract, the `unwindStrategy` function allows to unwind a part of the capital. However, it does not consider the unspent tokens of the Strategy. As a result, the function will transfer the unwind amount and unspent tokens to the Garden at line 437.

> *Comment from the developers: The impact is very limited as usually a strategy executes all the capital or at least it should execute it, so there should not be unspent tokens on it. Having unspent amount in a strategy does not make sense to leave there when it can be used in other strategies by the garden.*

This analysis was performed by Pessimistic:

Daria Korepanova, Security Engineer
Evgeny Marchenko, Senior Security Engineer
Boris Nikashin, Analyst
Irina Vikhareva, Project Manager

February 22, 2022