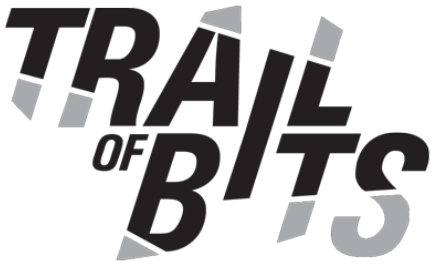

DODO

Security Assessment

September 18th, 2020

Prepared For:
Radar Bear | DODOEX
radarbear@dodoex.io

Prepared By:
Natalie Chin | *Trail of Bits*
natalie.chin@trailofbits.com

Michael Colburn | *Trail of Bits*
michael.colburn@trailofbits.com

Dominik Teiml | *Trail of Bits*
dominik.teiml@trailofbits.com

Review Summary

Code Maturity Evaluation

Project Dashboard

Appendix A. Code Maturity Classifications

Review Summary

From August 31 to September 18, 2020, Trail of Bits performed an assessment of the DODO smart contracts with three engineers over three person-weeks, and reported 10 issues ranging from high to informational severity.

Throughout this assessment, we sought to answer various questions about the security of DODO. We focused on flaws that would allow an attacker to:

- Gain unauthorized access to user funds
- Bypass access controls to modify contract state
- Interfere with interactions between DODO components

Of the findings reported, the high-severity issue related to the level of influence the owner address had over the DODO system, where an attacker would have only needed to compromise an owner's private key to withdraw funds at a significant discount. The low-severity issues pertained to copy-pasted third-party dependencies, meaning DODO may not be aware of upstream changes in other contracts. Another issue allowed Liquidity Providers to unintentionally burn tokens while withdrawing funds from the pool by allowing tokens to be sent to `address(0)`. Two undetermined-severity issues related to the usage of Solidity compiler optimizations and `ABIEncoderV2`, which may have unexpected side effects as these features are not enabled by default. Several informational issues were also raised regarding functions that didn't adhere to the `checks-effects-interactions` pattern in the event that other safeguards against reentrancy were removed in the future, removal of ineffective conditional checks on largely fluctuating gas prices, and ensuring all state-changing functions emitted events to help users and third-party apps stay up-to-date with the blockchain state.

On the following page, we review the maturity of the codebase and the likelihood of future issues. In each area of control, we rate the maturity from strong to weak, or missing, and give a brief explanation of our reasoning. DODO should consider these steps to improve their security maturity:

- Use crytic.io for any new code development.
- Use fuzzing or symbolic execution to test the correctness of the arithmetic functions.
- Follow best practices for privileged accounts, e.g., use a multi-sig wallet for the owner, and consider the use of an HSM (see our HSM recommendations).

Code Maturity Evaluation

Category Name	Description
Access Controls	**Satisfactory.** Appropriate access controls were in place for performing privileged operations.
Arithmetic	**Satisfactory.** The contracts included use of safe arithmetic functions. No potential overflows were identified in areas where these functions were not used.
Assembly Use	**Not Applicable.** The contracts did not make use of assembly except as part of the EIP-1167 `CloneFactory`, which was out of scope.
Centralization	**Moderate.** The contract owner address had the ability to modify many critical system parameters after deployment. The contracts are currently deployed with a 1-of-3 multisig wallet set as the owner. A malicious insider with access to any of the three keys would therefore be able to access liquidity at an arbitrary discount.
Contract Upgradeability	**Not Applicable.** The contracts use migrations to update contract logic.
Function Composition	**Strong.** Functions and contracts were organized and scoped appropriately.
Front-Running	**Strong.** No front-running issues were identified.
Monitoring	**Moderate.** We identified several administrative functions that would have benefited from events.
Specification	**Moderate.** High-level documentation and some in-line comments describing the functionality of the protocol were available. The contract source code would benefit from NatSpec comments for contracts and functions.
Testing & Verification	**Satisfactory.** The repositories included tests for a variety of scenarios.

Project Dashboard

Commit hashes of the reviewed repositories:

- `dodo-smart-contract`: 146a2a2

Smart contracts located in the `contracts/helper` directory were considered out of scope for this assessment.

Appendix A. Code Maturity Classifications

Code Maturity Classes	
Category Name	**Description**
Access Controls	Related to the authentication and authorization of components.
Arithmetic	Related to the proper use of mathematical operations and semantics.
Assembly Use	Related to the use of inline assembly.
Centralization	Related to the existence of a single point of failure.
Upgradeability	Related to contract upgradeability.
Function Composition	Related to separation of the logic into functions with clear purpose.
Front-Running	Related to resilience against front-running.
Key Management	Related to the existence of proper procedures for key generation, distribution, and access.
Monitoring	Related to use of events and monitoring procedures.
Specification	Related to the expected codebase documentation.
Testing & Verification	Related to the use of testing techniques (unit tests, fuzzing, symbolic execution, etc.).

Rating Criteria	
Rating	**Description**
Strong	The component was reviewed and no concerns were found.
Satisfactory	The component had only minor issues.
Moderate	The component had some issues.
Weak	The component led to multiple issues; more issues might be present.
Missing	The component was missing.

Not Applicable	The component is not applicable.
Not Considered	The component was not reviewed.
Further Investigation Required	The component requires further investigation.