

MYSO v2

Security assessment by HashEye · prepared for MYSO Finance

HASHEYE AUDITED

PROJECT	MYSO v2
CLIENT	MYSO Finance
CATEGORY	Blockchain
PUBLISHED	April 1, 2023
REPORT ID	research-myso-v2-2023-04-01-1bljca

This report was produced under HashEye's layered review process – **automated detection**, **pattern correlation**, and **senior manual verification** – with every finding signed off by a human reviewer. Full findings detail and on-chain attestation are available on the report page at hasheye.io/audits/research-myso-v2-2023-04-01-1bljca.

MYSOV2 SecurityAssessment(SummaryReport) April18,2023 Preparedfor: MYSOFinance
Preparedby:NatChin,RobertSchneider,andElvisSkozdopolj

About hashey Founded in 2012 and headquartered in New York, hashey provides technical security assessment and advisory services to some of the world's most targeted organizations. We combine high-end security research with a real-world attacker mentality to reduce risk and fortify code. With 100+ employees around the globe, we've helped secure critical software elements that support billions of end users, including Kubernetes and the Linux kernel. We maintain an exhaustive list of publications at <https://github.com/hashey-io/publications>, with links to papers, presentations, public audit reports, and podcast appearances. In recent years, hashey consultants have showcased cutting-edge research through presentations at CanSecWest, HCSS, Devcon, Empire Hacking, GrrCon, LangSec, NorthSec, the O'Reilly Security Conference, PyCon, REcon, Security BSides, and SummerCon. We specialize in software testing and code review projects, supporting client organizations in the technology, defense, and finance industries, as well as government entities. Notable clients include HashiCorp, Google, Microsoft, Western Digital, and Zoom. hashey also operates a center of excellence with regard to blockchain security. Notable projects include audits of Algorand, Bitcoin SV, Chainlink, Compound, Ethereum 2.0, MakerDAO, Matic, Uniswap, Web3, and Zcash. To keep up to date with our latest news and announcements, please follow hashey on Twitter and explore our public repositories at <https://github.com/hashey-io>. To engage us directly, visit our "Contact" page at <https://www.hashey.io/contact>, or email us at info@hashey.io. hashey, Inc. 228 Park Ave S #80688 New York, NY 10003 <https://www.hashey.io> info@hashey.io hashey 1MYSOFinanceSecurityAssessment PUBLIC

Notices and Remarks Copyright and Distribution ©2023 by hashey, Inc. All rights reserved. hashey hereby asserts its right to be identified as the creator of this report in the United Kingdom. This report is considered by hashey to be public information; it is licensed to MYSOFinance under the terms of the project statement of work and has been made public at MYSOFinance's request. Material within this report may not be reproduced or distributed in part or in whole without the express written permission of hashey. This is the canonical source for hashey publications; see the hashey Publications page. Reports accessed through any source other than that page may have been modified and should not be considered authentic. Test Coverage Disclaimer All activities undertaken by hashey in association with this project were performed in accordance with a statement of work and agreed upon project plan. Security assessment projects are time-boxed and often reliant on information that may be provided by a client, its affiliates, or its partners. As a result, the findings documented in this report should not be considered a comprehensive list of security issues, flaws, or defects in the target system or codebase. hashey uses automated testing techniques to rapidly test the controls and security properties of software. These techniques augment our manual security review work, but each has its limitations: for example, a tool may not generate a random edge case that violates a property or may not fully complete its analysis during the allotted time. Their use is also limited by the time and resource constraints of a project. hashey 2MYSOFinanceSecurityAssessment PUBLIC

Table of Contents About hashey 1 Notices and Remarks 2 Table of Contents 3 Executive Summary 4 Project Summary 6 Project Goals 7 Project Targets 8 Project Coverage 9 Codebase Maturity Evaluation 11 Summary of Findings 14 A. Vulnerability Categories 16 B. Code Quality Recommendations 18 C. Incident Response Recommendations 19 D. Token Integration Checklist 21 G. Fix Review Results 26 Detailed Fix Log 27 hashey 3MYSOFinanceSecurityAssessment PUBLIC

Executive Summary Engagement Overview MYSOFinance engaged hashey to review the security of its lending contracts. From March 20 to March 24, 2023, a team of three consultants conducted a security review of the client-provided source code, with two person-weeks of effort. Our testing efforts focused on identifying issues that would result in attackers stealing funds by abusing the arithmetic in the products. With access to source code and documentation, we performed a manual review of the target system. Observations and Impact In developing the target codebases, MYSOFinance implemented peer-to-pool and peer-to-peer lending pools. Our reviews suggested that the codebase would benefit from more effort toward unit and automated testing and arithmetic analysis. Issues found during this audit stem from:

Improper precision handling in arithmetic • Insufficiently documented information
During the audit, we discovered several high- and medium-severity findings that impact
many of the components, including: • Manipulation of Liquidity Provider prices • Loss of token through the
arranger fee configuration •
Rollback functionality incorrectly transferring collateral to caller instead of borrower •
Denial of service for lenders due to missing access control Recommendations
Based on the codebase maturity evaluation and findings identified during the security
review, we recommend that MYSO Finance address the following recommendations prior to deploying the peer-
to-peer and peer-to-pool lending products: •
Remediate issues found during this review. The findings described during this
review pose tangible risks to MYSO Finance. These findings should be addressed
immediately as part of a direct remediation. • Identify global and per-
function system invariants that are expected to hold
true against the system. These invariants should specify expected system
behavior when calling certain functions and specify the global system state. hashey
4MYSO Finance Security Assessment PUBLIC

Build a fuzzing suite to test end-to-end system interactions. Implementing the
aforementioned system invariants into fuzz tests will help to identify additional edge
cases that the system may not support. These practices should help uncover obscure bugs that may not be obvious. •
Enhance existing documentation to cover expected user flows and expected
data validation in the system. The coder relies on a number of assumptions in the
expected user flow. All contract entry points and expected checks should be explicitly documented.
The following tables provide the number of findings by severity and category. EXPOSURE ANALYSIS Severity Count
High 2 Medium 5 Low 3 Informational 4 Undetermined 2 CATEGORY BREAKDOWN Category Count AccessControl 2
Auditing and Logging 1 Authentication 1 Configuration 3 Data Validation 6 Patching 1 Timing 1
Undefined Behavior 1 hashey 5MYSO Finance Security Assessment PUBLIC

Project Summary Contact Information The following managers were associated with this project:
Dan Guido, Account Manager Brooke Langhorne, Project Manager dan@hashey.io brooke.langhorne@hashey.io
The following engineers were associated with this project: Nat Chin, Consultant Robert Schneider, Consultant
natalie.chin@hashey.io robert.schneider@hashey.io Elvis Skozdopolj, Consultant
elvis.skozdopolj@hashey.io Project Timeline
The significant events and milestones of the project are listed below. Date Event March 15, 2023 Pre-
project kickoff call March 27, 2023 Delivery of report draft March 27, 2023 Report readout meeting
April 18, 2023 Delivery of final report hashey 6MYSO Finance Security Assessment PUBLIC

Project Goals The engagement was scoped to provide a security assessment of the MYSO Loans.
Specifically, we sought to answer the following non-exhaustive list of questions: •
Is it possible for an attacker to steal funds? • Is the system susceptible to front-running? •
Does the system safely handle price feed changes? • Are all functions protected adequately with access controls?
• Does the system prevent signature replays? hashey 7MYSO Finance Security Assessment PUBLIC

Project Targets The engagement involved a review and testing of the following target. MYSO Smart Contracts v2
Repository <https://github.com/mysofinance/v2> Version 4ff4e1ee98278eca953ce6a412c57664937d11c0
Type Solidity Platform Ethereum hashey 8MYSO Finance Security Assessment PUBLIC

Project Coverage This section provides an overview of the analysis coverage of the review, as determined by
our high-level engagement goals. Our approaches include a best-efforts review of the following: Peer-to-
Peer Contracts. This set of contracts implements a borrower-lender matching
system, allowing borrowers to borrow directly from lender vaults. We reviewed these
contracts using Slither, our static analyzer and manual review. • Borrower Gateway
: This contract is the entry point for borrowers when they want to
lend money from the system. This contract interacts with the Address Registry ,
which keeps track of deployed vaults for lenders, and processes the transfer of
funds into and out of lender vaults. We reviewed this system focusing on access controls and its cross-
contract interactions with the LenderVaultImpl . •
LenderVaultFactory and Implementation: This contract is the entry point for
lenders to add money into the system, to withdraw funds, and to unlock collateral.
This contract contains a set of borrower-related functions that are intended to be
callable by only the Borrower Registry, a series of functions that are lender vault-
only controlled. We reviewed this system with a focus on access controls and its interaction with the
BorrowerGateway . We focused on analyzing the inflow and outflow of tokens. •
QuoteHandler: This contract is tasked with the verification and processing of off-
chain loans and the matching of on-chain loans. We briefly reviewed the
signature verification process, with a focus on replay attacks and signature forgery. •
Miscellaneous Helper Contracts (AddressRegistry, DataTypes): These consist of

helper functions created to support the above contracts. We reviewed these contracts using Slither and manual review. Peer-to-Pool-Contracts. This set of contracts allows lenders to add funds into a pool, and borrowers to borrow from pools. • Funding Pool : This contract is the entry point that borrowers and lenders would use to deposit, withdraw, subscribe, unsubscribe, and execute loan proposals. We briefly reviewed the arithmetic in this contract; however, this area requires additional exploration. • Loan Proposal Factory and Implementation: These contracts construct the state chain of a loan taken out from a pool. These contracts also handle the relevant checks before changing state, and the actual adjustment of that state. We briefly reviewed the data validation between state transitions and the progression of the state machine. Coverage Limitations Because of the time-boxed nature of testing work, it is common to encounter coverage limitations. The following list outlines the coverage limitations of the engagement and indicates system elements that may warrant further review: • Signature verification and Merkle trees: we recommend further investigation on the signature process for off-chain loans, as our review was non-conclusive. • Arithmetic: Due to the intricate nature of the lending pools, the arithmetic in the system requires closer investigation and further effort in automated testing. 9MYSO Finance Security Assessment PUBLIC

Codebase Maturity Evaluation hashey uses a traffic-light protocol to provide each client with a clear understanding of the areas in which its codebase is mature, immature, or underdeveloped. Deficiencies identified here often stem from root causes within the software development lifecycle that should be addressed through standardization measures (e.g., the use of common libraries, functions, or frameworks) or training and awareness programs. Category Summary Result Arithmetic Solidity v0.8.0+, which uses native overflow and underflow protection, is used throughout the codebase. We identified only a few instances of unchecked arithmetic in the codebase, aside from certain uses of loops as incrementors. With respect to the specific protocol actions, it was challenging to detect deviations between expected and current code behavior, as specifications on arithmetic formulas were not present. Moreover, automated program analysis techniques such as fuzzing were not applied to this part of the codebase. Further Investigation Required Auditing A few state-changing functions were missing events, which would have made it easier for the codebase to detect unexpected behavior. The MYSO Finance team currently uses OpenZeppelin Defender and a custom cron job to detect and alert to any state changes. We recommend ensuring that all state-changing functions in the system adequately emit events (Slither analyzer can help with this). Moderate Authentication/Access Controls The current set of contracts splits up access controls into a variety of different roles, which ensures that no single role is given too much power. The MYSO Finance team uses a multi-sig to execute special actions. Certain contracts make assumptions on the flow of access controls; these would benefit from further clarification and reasoning. Satisfactory Complexity Management The codebase supports a number of features, and splits up reusable code into libraries and helper contracts. We Weak hashey 10MYSO Finance Security Assessment PUBLIC

also found a few instances of duplicated code in the codebase, which should be addressed through the use of helper functions. However, some of these flows require callers to go through different entry points to call contracts; it would be beneficial to either document or simplify this process. Decentralization The privileged functionality is controlled by a multi-sig, which controls system parameters and adjust how contracts work together. We recommend providing users with additional documentation on the circumstances under which system parameters and privileges across all system roles would need to change. Moderate Documentation The MYSO Finance contracts lack documentation specifying expected data validation for cross-contract calls, and the identification of core components and critical code blocks. In a few reported cases, the code comments deviated from the code and were not in line with system expectations. We recommend MYSO Finance invest additional effort into written and flow-chart documentation on all important aspects of their system. Weak Transaction Reordering Risks Inherently by design, these contracts can be abused by MEV and/or sequencer reordering. As such, we recommend MYSO Finance analyze the system as a whole to ensure all potentials for front-running are understood, mitigated if possible, and documented if mitigation is impossible. Weak Low-Level Manipulation The assembly in the codebase is kept to a minimum. The signature verification scheme contains uses of off-chain signatures, which require additional investigation. Further Investigation Required Testing and Verification Although the code reaches 96.9% statement and 87.29% branch coverage, it tests primarily happy-path scenarios. Moreover, although this test suite is being used in a continuous integration pipeline, tests on Arbitrum were failing when we ran it. As such, we recommend investing Weak hashey 11MYSO Finance Security Assessment PUBLIC

also found a few instances of duplicated code in the codebase, which should be addressed through the use of helper functions. However, some of these flows require callers to go through different entry points to call contracts; it would be beneficial to either document or simplify this process. Decentralization The privileged functionality is controlled by a multi-sig, which controls system parameters and adjust how contracts work together. We recommend providing users with additional documentation on the circumstances under which system parameters and privileges across all system roles would need to change. Moderate Documentation The MYSO Finance contracts lack documentation specifying expected data validation for cross-contract calls, and the identification of core components and critical code blocks. In a few reported cases, the code comments deviated from the code and were not in line with system expectations. We recommend MYSO Finance invest additional effort into written and flow-chart documentation on all important aspects of their system. Weak Transaction Reordering Risks Inherently by design, these contracts can be abused by MEV and/or sequencer reordering. As such, we recommend MYSO Finance analyze the system as a whole to ensure all potentials for front-running are understood, mitigated if possible, and documented if mitigation is impossible. Weak Low-Level Manipulation The assembly in the codebase is kept to a minimum. The signature verification scheme contains uses of off-chain signatures, which require additional investigation. Further Investigation Required Testing and Verification Although the code reaches 96.9% statement and 87.29% branch coverage, it tests primarily happy-path scenarios. Moreover, although this test suite is being used in a continuous integration pipeline, tests on Arbitrum were failing when we ran it. As such, we recommend investing Weak hashey 12MYSO Finance Security Assessment PUBLIC

furthereffortstointoautomatedtesting(Echidna)and ensuringthatthoroughunitandintegrationtestsexist forallexpectedandunexpectedpaths. hashey 13MYSOFinanceSecurityAssessment PUBLIC

SummaryofFindings Thetablebelowsummarizesthefindingsofthereview,includingtypeandseveritydetails.
IDTitleTypeSeverity 1IncorrectconstantsDataValidationLow
2TokenscanbestolenthrougharrangerFeeDataValidation High 3ArrangercanfrontrunacceptLoanTermsTiming
Medium 4RiskofmisconfiguredloansDataValidation Informational
5LoanrollbackcantransferfundstocallerAccessControl Medium
6LPtokenpricecanbemanipulatedDataValidation Undetermined 7MissingimportantChainlinkpricefeedsafety
checks DataValidation Medium 8RiskofdenialofserviceattackviaunlockCollateralAccessControl Medium
9InsufficenteventgenerationAuditingand Logging Low
10LackofzerochecksonfunctionargumentsDataValidation Low 11DocumentationcanbeimprovedUndefined
Behavior Informational 12Insufficientprotectiononsensitiveownerprivate keys Configuration
Informational hashey 14MYSOFinanceSecurityAssessment PUBLIC

13UseofchainIDvalidationallowsre-usingsignature acrossforks Configuration Informational 14off-
chainsignatureschemamaybeatargetfor phishing Authentication High
15ProjectdependenciescontainvulnerabilitiesPatching Medium 16Insufficientprotectiononcross-
vaultsignature reuse Configuration Undetermined hashey 15MYSOFinanceSecurityAssessment PUBLIC

A.VulnerabilityCategories

Thefollowingtablesdescribethevulnerabilitycategories,severitylevels,anddifficulty levelsusedinthisdocument. VulnerabilityCategories CategoryDescription
AccessControlsInsufficientauthorizationorassessmentofrights
AuditingandLoggingInsufficientauditingofactionsorloggingofproblems
AuthenticationImproperidentificationofusers
ConfigurationMisconfiguredservers,devices,orsoftwarecomponents
CryptographyAbreachofsystemconfidentialityorintegrity DataExposureExposureofsensitiveinformation
DataValidationImproperrelianceonthestructureorvaluesofdata
DenialofServiceAsystemfailurewithanavailabilityimpact
ErrorReportingInsecureorinsufficientreportingoferrorconditions
PatchingUseofanoutdatedsoftwarepackageorlibrary
SessionManagementImproperidentificationofauthenticatedusers
TestingInsufficienttestmethodologyortestcoverage TimingRaceconditionsorotherorder-of-
operationsflaws UndefinedBehaviorUndefinedbehaviortriggeredwithintheprocess hashey
16MYSOFinanceSecurityAssessment PUBLIC

SeverityLevels SeverityDescription
InformationalTheissuedoesnotposean immediateriskbutisrelevanttosecuritybest practices.
UndeterminedTheextentoftheriskwasnotdeterminedduringthisengagement.
LowTheriskissmallorisnotonetheclienthasindicatedisimportant.
MediumUserinformationisatrisk;exploitationcouldposereputational,legal,or moderatefinancialrisks.
HighTheflawcouldaffectnumeroususersandhaveseriousreputational,legal, orfinancialimplications.
DifficultyLevels DifficultyDescription
UndeterminedThedifficultyofexploitationwasnotdeterminedduringthisengagement.
LowTheflawiswellknown;publictoolsforitsexploitationexistorcanbe scripted.
MediumAnattackermustwriteanexploitorwillneedin-depthknowledgeofthesystem.
HighAnattackermusthaveprivilegedaccesstotheprocess,mayneedtoknow
complextechnicaldetails,ormustdiscoverotherweaknessestoexploitthis issue. hashey
17MYSOFinanceSecurityAssessment PUBLIC

B.CodeQualityRecommendations • Usehelperfunctionstoreducecodeduplication.Forexample,considermoving
toUint128() intoahelperfunctioninsteadofusingitin LenderVaultImpl . • Rename toggleTokens to
setWhitelistState .Thefunctiondoesnot
automaticallytoggletheexistingstate, butrather setsittoaspecifiedargument. • UseSoliditybuilt-
intimeunitstoenhancereadability.The Constants library
storesaseriesofnumbers,withcodecommentsnexttothemthatidentifytheir meaning.UseofSolidityin-
builttimeunitscanmakethiscontentmucheasiertoread. • Differentiate IEvents and DataTypes forpeer-
to-poolandpeer-to-peerloans. Theidenticalnamingmaycauseadeveloperwhoattemptstoupdateafunc-
tionrelatedtooneoftheloantypestoaccidentallyeditthewrongone. •
Useimmutablevariableswherepossible.Thisensures thatvariables cannot accidentallybeoverwritten. ◦
isUSDBased in BaseOracle.sol canbedeclaredimmutable. hashey 18MYSOFinanceSecurityAssessment PUBLIC

C.IncidentResponseRecommendations

Inthissection,weproviderecommendationsaroundtheformulationofanincident responseplan.
Identifywho(eitherspecificpeopleorroles)isresponsibleforcarryingoutthe
mitigations(deployingsmartcontracts,pausingcontracts,upgradingthefrontend, etc.). •

Specifying these roles will strengthen the incident response plan and ease the execution of mitigating actions when necessary.

- Document internal processes for situations in which a deployed remediation does not work or introduces a new bug.
- Consider adding a fallback scenario that describes an action plan in the event of a failed remediation.

Clearly describe the intended process of contract deployment.

- Consider whether and under what circumstances MYSO Finance will make affected users whole after certain issues occur.
- Some scenarios to consider include an individual or aggregate loss, a loss resulting from user error, a contract flaw, and a third-party contract flaw.

Document how MYSO Finance plans to keep up to date on new issues, both to inform future development and to secure the deployment tool chain and the external on-chain and off-chain services that the system relies on.

- For each language and component, describe noteworthy sources for vulnerability news. Subscribe to updates for each source. Consider creating a special private Discord channel with a bot that will post the latest vulnerability news; this will help the team keep track of updates all in one place. Also consider assigning specific team members to keep track of the vulnerability news of a specific component of the system.
- Consider scenarios involving issues that would indirectly affect the system.
- Determine when and how the team would reach out to and on-board external parties (auditors, affected users, other protocol developers, etc.) during an incident.
- Some issues may require collaboration with external parties to efficiently remediate them.

hashey 19MYSO Finance Security Assessment PUBLIC

- Define contract behavior that is considered abnormal for off-chain monitoring.
- Consider adding more resilient solutions for detection and mitigation, especially in terms of specific alternate endpoints and queries for different data as well as status pages and support contacts for affected services.

Combine issues and determine whether new detection and mitigation scenarios are needed.

Perform periodic dry runs of specific scenarios in the incident response plan to find gaps and opportunities for improvement and to develop muscle memory.

- Document the intervals at which the team should perform dry runs of the various scenarios. For scenarios that are more likely to happen, perform dry runs more regularly. Create a template to be filled in after a dry run to describe the improvements that need to be made to the incident response.

hashey 20MYSO Finance Security Assessment PUBLIC

D. Token Integration Checklist

The following checklist provides recommendations for interactions with arbitrary tokens. Every unchecked item should be justified, and its associated risks, understood. For an up-to-date version of the checklist, see [crytic/building-secure-contracts](#).

For convenience, all Slither utilities can be run directly on a token address, such as the following: `slither-check-erc0xdac17f958d2ee523a2206206994597c13d831ec7TetherToken--ercerc20 slither-check-erc0x06012c8cf97BEaD5deAe237070F9587f8E7A266dKittyCore--ercerc721`

To follow this checklist, use the below output from Slither for the token: `slither-check-erc[contractName][optional:--ercERC_NUMBER] slither[target]--printhuman-summary slither[target]--printcontract-summary slither-prop.--contractContractName#requiresconfiguration, and use of Echidna and Manticore`

General Considerations

- ☐ The contract has a security review. Avoid interacting with contracts that lack a security review. Check the length of the assessment (i.e., the level of effort), the reputation of the security firm, and the number and severity of the findings.
- ☐ You have contacted the developers. You may need to alert their team to an incident. Look for appropriate contact on [blockchain-security-contacts](#).
- ☐ They have a security mailing list for critical announcements. Their team should advise users (like you!) when critical issues are found or when upgrades occur.

Contract Composition

- ☐ The contract avoids unnecessary complexity. The tokens should be a simple contract; a token with complex code requires a higher standard of review. Use Slither's `human-summary printertoidentifycomplexcode`.
- ☐ The contract uses `SafeMath`. Contracts that do not use `SafeMath` require a higher standard of review. Inspect the contract by hand for `SafeMath` usage.
- ☐ The contract has only a few non-token-related functions. Non-token-related functions increase the likelihood of an issue in the contract. Use Slither's `contract-summary printertobroadlyreviewthecodeusedinthecontract`.

hashey 21MYSO Finance Security Assessment PUBLIC

- ☐ The token has only one address. Tokens with multiple entry points for balance updates can break internal bookkeeping based on the address (e.g., `balances[token_address][msg.sender]` may not reflect the actual balance).
- ☐ The token is not upgradeable. Upgradeable contracts may change their rules over time. Use Slither's `human-summary printertodeterminewhetherthecontractisupgradeable`.

☐The owner has limited minting capabilities. Malicious or compromised owners can abuse minting capabilities. Use `Slither`'s human-summary printer to review minting capabilities, and consider manually reviewing the code.

☐The token is not pausable. Malicious or compromised owners can trap contracts relying on pausable tokens. Identify pausable code by hand.

☐The owner cannot blacklist the contract. Malicious or compromised owners can trap contracts relying on tokens with `blacklist`. Identify blacklisting features by hand.

☐The team behind the token is known and can be held responsible for abuse.

Contracts with an anonymous development team or teams that reside in legal shelters require a higher standard of review. ERC20 Tokens ERC20 Conformity Checks `Slither` includes a utility, `slither-check-erc`, that reviews the conformance of a token to many related ERC standards. Use `slither-check-erc` to review the following: ☐ `Transfer` and `transferFrom` return a boolean. Several tokens do not return a boolean on these functions. As a result, their calls in the contract might fail. ☐The `name`, `decimals`, and `symbol` functions are present if used. These functions are optional in the ERC20 standard and may not be present. ☐ `Decimals` returns a `uint8`. Several tokens incorrectly return a `uint256`. In such cases, ensure that the value returned is below 255.

☐The token mitigates the known ERC20 race condition. The ERC20 standard has a known ERC20 race condition that must be mitigated to prevent attackers from stealing tokens. `Slither` includes a utility, `slither-prop`, that generates unit tests and security properties that can discover many common ERC flaws. Use `slither-prop` to review the following: `hashey` 22MYS0FinanceSecurityAssessment PUBLIC

☐The contract passes all unit tests and security properties from `slither-prop`. Run the generated unit tests and then check the properties with `Echidna` and `Manticore`. Risks of ERC20 Extensions The behavior of certain contracts may differ from the original ERC specification. Conduct a manual review of the following conditions: ☐The token is not an ERC777 token and has no external function call in `transfer` or `transferFrom`. External calls in the transfer functions can lead to reentrancies. ☐ `Transfer` and `transferFrom` should not take a fee. Deflationary tokens can lead to unexpected behavior.

☐Potential interest earned from the token is taken into account. Some tokens distribute interest to token holders. This interest may be trapped in the contract if not taken into account. `Token Scarcity` Reviews of token scarcity issues must be executed manually. Check for the following conditions:

☐The supply is owned by more than a few users. If a few users own most of the tokens, they can influence operations based on the tokens' repartition.

☐The total supply is sufficient. Tokens with a low total supply can be easily manipulated.

☐The tokens are located in more than a few exchanges. If all the tokens are in one exchange, a compromise of the exchange could compromise the contract relying on the token.

☐Users understand the risks associated with a large amount of funds or flash loans. Contracts relying on the token balance must account for attackers with a large amount of funds or attack executed through flash loans.

☐The token does not allow flash minting. Flash minting can lead to substantial swings in the balance and the total supply, which necessitate strict and

comprehensive overflow checks in the operation of the token. `hashey` 23MYS0FinanceSecurityAssessment PUBLIC

ERC721 Tokens ERC721 Conformity Checks

The behavior of certain contracts may differ from the original ERC specification. Conduct a manual review of the following conditions: ☐ `Transfer` of tokens to the `0x0`

address reverts. Several tokens allow transfers to `0x0`

and consider token transferred to that address to have been burned; however,

the ERC721 standard requires that such transfers revert. ☐ `safeTransferFrom`

functions are implemented with the correct signature.

Several token contracts do not implement these functions. A transfer of NFTs to one

of those contracts can result in a loss of assets. ☐The `name`, `decimals`, and `symbol`

functions are present if used. These functions are optional in the ERC721 standard and may not be present.

☐If it is used, `decimals` returns a `uint8(0)`. Other values are invalid. ☐The `name` and `symbol`

functions can return an empty string. This behavior is allowed by the standard. ☐The `ownerOf`

function reverts if the `tokenId` is invalid or is set to a token that

has already been burned. The function cannot return `0x0`. This behavior is

required by the standard, but it is not always properly implemented.

☐A transfer of an NFT clears its approvals. This is required by the standard.

☐The token ID of an NFT cannot be changed during its lifetime. This is required by the standard.

Common Risks of the ERC721 Standard

To mitigate the risks associated with ERC721 contracts, conduct a manual review of the following conditions: ☐The

`onERC721Received` callback is taken into account. External calls in the

transfer functions can lead to reentrancies, especially when the callback is not explicit (e.g., in `safeMint`

calls). `hashey` 24MYS0FinanceSecurityAssessment PUBLIC

When an NFT is minted, it should behave similarly to safeTransferFrom and properly handle the minting of new tokens to a smart contract. This will prevent a loss of assets.

The burning of a token clears its approvals. If there is a burning function, it should clear the token's previous approvals. [hashey 25 MYSO Finance Security Assessment PUBLIC](#)

6. Fix Review Results On April 10, 2023, [hashey](#) reviewed the fixes and mitigations implemented by the MYSO Finance team for the issues identified in this report. We reviewed each of the fixes to determine their effectiveness in resolving the associated issues. For additional information, see the Detailed Fix Log.

ID	Title	Severity	Status	
1	Incorrect constants	Low	Resolved	
2	Tokens can be stolen through arrangerFee	High	Resolved	
3	Arranger can front-run acceptLoanTerms	Medium	Resolved	
4	Risk of misconfigured loans	Informational	Resolved	
5	Loan rollback can transfer funds to caller	Medium	Resolved	
6	LPToken price can be manipulated	Undetermined	Resolved	
7	Missing important Chainlink price feeds	safety checks	Medium	Resolved
8	Risk of denial of service attack via unlockCollateral	Medium	Resolved	
9	Insufficient event generation	Low	Resolved	
10	Lack of zero checks on function arguments	Low	Resolved	
11	Documentation can be improved	Informational	Partially Resolved	

[hashey 26 MYSO Finance Security Assessment PUBLIC](#)

12 Insufficient protection on sensitive owner private keys Informational Resolved

13 Use of chain ID validation allows re-using signature across forks Informational Resolved

14 Off-chain signature scheme may be a target for phishing High Resolved

15 Project dependencies contain vulnerabilities Medium Resolved

16 Insufficient protection on cross-vaults signature reuse Undetermined Resolved

Detailed Fix Log

TOB-MYSO-1: Incorrect constants Resolved in PR#148. Solidity time units are now employed in constant declarations to ensure the accurate assignment of values.

TOB-MYSO-2: Tokens can be stolen through arrangerFee Resolved in PR#148. A new constant variable, `MAX_ARRANGER_FEE`, has been introduced and assigned a value of 50%. Additionally, a validation has been incorporated into the loan proposal's initialization function to confirm that the arranger fee remains below this threshold.

TOB-MYSO-3: Arranger can front-run acceptLoanTerms Resolved in PR#148. A `LOAN_TERMS_UPDATE_COOL_OFF_PERIOD` constant has been added and set to a duration of one hour. The function `ProposeLoanTerms` now registers the `lastLoanTermsUpdateTime` with the current block timestamp. Subsequently, the `acceptLoanTerms` function ensures that the elapsed time between updates is equal to or greater than the one-hour constant. This enhancement effectively prohibits lenders from altering loan terms more frequently than once per hour.

TOB-MYSO-4: Risk of misconfigured loans Resolved in PR#148. The grace period between reconfigurations has been revised from 30 minutes to a full day. Also, the `MIN_TIME_BETWEEN_DUE_DATES` constant has been extended from one day to a week. Furthermore, documentation regarding loan configuration options has been enhanced for lenders, borrowers, and arrangers. [hashey 27 MYSO Finance Security Assessment PUBLIC](#)

TOB-MYSO-5: Loan rollback can transfer funds to caller Resolved in PR#148. The `safeTransfer` function is now accepting the `_loanTerms.borrower` address as a parameter, rather than the `msg.sender` address, enhancing the overall security of the transaction.

TOB-MYSO-6: LPToken price can be manipulated Resolved in PR#159, PR#160, PR#157. An LPToken's price is now calculated using the "fair reserve" concept. This improves accuracy and reduces potential manipulation. Instead of multiplying token reserves by price, the constant value `k` is derived from the pool's token reserves and computed against the price. Additional documentation and unit tests have also been included.

TOB-MYSO-7: Missing important Chainlink price feeds safety checks Resolved in PR#147, PR#156. Data returned by Chainlink price feeds is now being sanity checked to ensure the oracle is returning accurate and up-to-date information.

TOB-MYSO-8: Risk of denial of service attack via unlockCollateral Resolved in PR#138. A line has been added to verify that `msg.sender` is the `_owner` when the `autoWithdraw` flag is set to true. Additionally, if a zero-length `_loanIds` array is passed in, the `unlockCollateral` operation will now revert.

TOB-MYSO-9: Insufficient event generation Resolved in PR#138, PR#162. Three events— `Withdraw`, `QuoteProcessed`, `OffChainQuoteUsed`, and `OnChainQuoteUsed`—have been introduced and are now being emitted at every location recommended in our suggestions.

TOB-MYSO-10: Lack of zero checks on function arguments Resolved in PR#161. A `NaN` spec comment has been incorporated to clarify that the protocol fee can be set to zero. Consequently, this is why there is no minimum value check for the `_newFee` argument within the `setProtocolFee` function.

TOB-MYSO-11: Documentation can be improved Partially resolved in multiple PRs. Most of the implemented fixes have involved enhancing in-line documentation. However, additional improvements to documentation, extending beyond code comments to visual aids and system diagrams, would benefit both users and developers seeking to interact with the protocol. MYSO provided the following insight regarding this finding and its status: MYSO is committed to fully and accurately document all parts of the system and codebase before the system is deployed for wide use. [hashey 28 MYSO Finance Security Assessment PUBLIC](#)

TOB-MYSO-12:Insufficientprotectiononsensitiveownerprivatekeys
ResolvedinPR#163,PR#165.MYSOhasremovedhard-codedAPIkeysforInfuraand
Alchemyservices,andtheyhaveremovedthe console.log statementthatwasprinting process.env.PRIVATE_KEY
to stdout . TOB-MYSO-13:Useof chainID validationallowsre-usingsignatureacrossforks
ResolvedinPR#165.MYSOhasaddedadecodecommentin AddressRegistry ,
documentingthatthetokenallowlistiscapableofdelistingalltokensandbrickingthe
protocolinthetheoreticalcasethatanimposterforkemergesthatsharesthesame chainId
astheEthereumMainnet. TOB-MYSO-14:Off-chainsignatureschemamaybeatargetforphishing
ResolvedinPR#163.MYSOhasacknowledgedourrecommendationsregardingthe
potentialphishingattacktargetingtheoff-chainsignatureschema.Becauseadirect
solutiondoesnotexist,theyhaveaddedinlinedocumentationtowarnbothdevelopers
andendusersabouttheassociatedrisks. TOB-MYSO-15:Projectdependenciescontainvulnerabilities
Resolved.Theprojectdependencieshavebeenupdated,and npmaudit nolongerreports anycriticalorhigh-
levelseverityissues.However,itstillreportssixmoderate-severity vulnerabilities. TOB-MYSO-
16:Insufficientprotectiononcross-vaultsignaturereuse ResolvedinPR#138.The lenderVault
addresshasbeenincorporatedintothehash schemaforsigningoff-
chainquotes.Thisimplementationwillhelppreventsignaturesfrom
beingreusedforvaultswithsimilarconfigurationsbelongingtothesamelenderonthe samechain. hashey
29MYSOFinanceSecurityAssessment PUBLIC