Smart Contract Security Assessment

December 16, 2022

Project Name:
Nau Finance

Prepared by:
HYDN

Contents

- Successfully remediated
- Acknowledged but no further action taken

HYDN Seal

Introduction

About HYDN

HYDN is an industry leader in blockchain security and smart contract audits. Founded by Warren Mercer, a world renowned cybersecurity and blockchain expert, who has previously held senior roles at NYSE, Cisco, and Alert Logic. Warren has been a guest speaker at cybersecurity conferences all over the globe, including giving talks on Bitcoin at Microsoft DCC, NorthSec, Kaspersky SAS, VB, and many more.

Having been involved in cryptocurrency for over 10 years, Warren is dedicated to making the blockchain ecosystem as secure as it can be for everyone. Warren serves as the CEO for HYDN and heads up the delivery team to ensure that work is carried out to the highest standard. The HYDN delivery team has over 10 years combined experience in blockchain, having performed smart contract audits and built security systems for a large range of protocols and companies.
HYDN works closely with Nau Finances to consider their unique business needs, objectives, and concerns. Our mission is to ensure the blockchain supports secure business operations for all and he has built the team at HYDN from the ground up to meet this very personal objective.

To keep up to date with our latest news and announcements, check out our website https://hydnsec.com/ or follow @hydnsecurity on Twitter.

About Nau Finance

Nau Finance is a NFT collateralized lending protocol that uses the crowdfunded, ”peer-to-multi peer" (p2mp) lending model. Nau incentivizes lenders to give high valuations and low-interest rates by using tranches where lenders of all risk tolerances to join in a single loan. NFTs are locked in the smart contract until the end of the loan term, and the borrowers can repay the principal plus interest to reclaim their loan. If a borrower defaults, lenders are repaid in order of their tranch, starting with the lowest interest rate.

Methodology

When tasked with conducting a security assessment, HYDN works through multiple phases of security auditing to ensure smart contracts are audited thoroughly. To begin the process automated tests are carried out, before HYDN then moves onto carrying out a detailed manual review of the smart contracts.

HYDN uses a variety of open-source tools and analyzers as and when they are required and alongside this, HYDN primarily focuses on the following classes of security and reliability issues:

Basic Coding Mistakes

One of the most common causes of critical vulnerabilities is basic coding mistakes. Countless projects have experienced hacks and exploits due to simple, surface level mistakes that could have been flagged and addressed by a simple code review. The HYDN automated audit process which includes model checkers, fuzzers, and theorem provers analyses the smart contract for many of these basic coding mistakes. Once the automated audit has taken place, HYDN then performs a manual review of the code to gain familiarity with the contracts.

Business Logic Errors

HYDN reviews the platform or projects design documents before analysing the code to ensure that the team has a deep understanding of the business logic and goals. Following this, HYDN reviews the smart contracts to ensure that the contract logic is in line with the expected functionality.

HYDN also analyses the code for inconsistencies, flaws, or vulnerabilities which could impact business logic such as Tokenomics errors, arbitrage opportunities, and share pricing.

Complex Integration Risks

Several high-profile exploits have been the result of not any bug within the contract itself, but rather an unintended consequence of its interaction with the broader DeFi ecosystem.
We perform a meticulous review of all of the contract's possible external interactions, and summarise the associated risks; for example: flash loan attacks, oracle price manipulation, MEV/sandwich attacks, etc.

Code Maturity

HYDN reviews the smart contracts for potential improvements in the codebase. These improvements ensure that the code follows industry best practices and guidelines, or code quality standards. Alongside this, HYDN makes suggestions for code optimization items such as gas optimization, upgradeability weaknesses, centralization risks, and more.

Impact Rankings

Once HYDN has completed the security assessment, each issue is assigned an impact rating. This impact rating is based upon both the severity and likelihood of the issue occurring. Each security assessment is different and the business needs and goals, such as project timelines, and Nau Finance threat modelling. are taken into account when the impact rankings are created.

HYDN assigns the following impact rankings: Critical, High, Medium, Low (listed by severity).

Disclaimer

This HYDN security assessment does not provide any warranties on finding all possible issues within the scope and the evaluation results do not guarantee the absence of any issues. **HYDN cannot make any guarantees on any further code which is added or altered after the security assessment has taken place**. As a single security assessment can never be considered fully comprehensive, HYDN always recommends multiple independent assessments paired with a bug bounty program. This security assessment report should not be considered as financial or investment advice.

Summary of Findings

Executive Summary

As part of this smart contract audit, HYDN analysed only the contracts listed below in the Project Summary. In general HYDN found the contracts to be extremely well written with a very mature, clean code base. HYDN only found two Medium severity issues and one Low severity issues.

Project Summary

Platform	Ethereum
Language	Solidity
Scope	CocktailLoan.sol Base.sol ERC721TransfererWithLegacyFallback.sol Renderer.sol WadRayMath.sol utils.sol Deployment and Utility Scripts
Repository	https://github.com/naufinance/nau-core
Initial Commits	82f5c1a40904d93aaacc06b726d167ec3dccf5d6
Latest Commits	83516695fbc6df0ab45570cebd759f3c398b63b1
Deployment address	

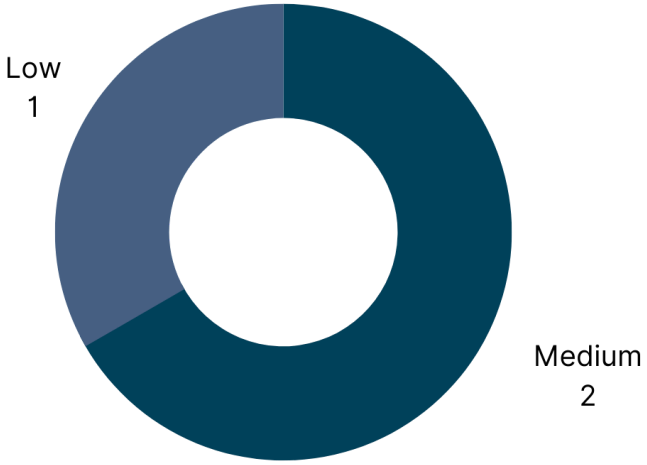
Audit Summary

Delivery Date	November 15th 2022
Audit Method	Static Automated Analysis + Manual Review

List of Checks

- Overflows & Underflows
- Reentrancy Vulnerability
- Replay Vulnerability
- Short Address Vulnerability
- Kill-Switch / Self-Destruct
- Denial of Service
- Arithmetic Accuracy
- Uninitialized Storage Pointers
- Time Based Vulnerability
- Loop Vulnerability
- Ownership Takeover
- Access Control
- Oracle Security
- Deployment process
- Unsafe Type Inference
- Visibility Level Explicity
- Centralization Risks
- Deprecated Issuage
- Standard Compliance
- Code Stability
- Best Practices

Breakdown of Findings



ID	Title	Severity	Status
1	Weak Validation of Collateral NFT Token	Medium	Fixed
2	Weak Validation of Loan Duration	Medium	Fixed
3	Outdated Solidity Compiler Version Used	Low	No further action

Detailed Findings

1 Weak Validation of Collateral NFT Token [Medium]

1.1 Description

When calling *requestNewLoan()* there is no validation at all on the source NFT token deposited as collateral to request a new loan.

```
195   function requestNewLoan(
196     address collateralToken,
197     uint256 collateralTokenID,
198     IERC20Metadata settlementToken,
199     uint256 minAmount,
200     uint256 maxAmount,
201     uint256 minAPRRay,
202     uint256 maxAPRRay,
203     uint40 endDate,
204     uint24 fundingPeriod
205   ) external whenNotPaused returns (uint256) {
```

Extract of CocktailLoan.sol

1.2 Impact

This lack of validation may lead to different unexpected behaviours as the contract interacts with a non-trusted and/or not-verified third party contract.

As an example we can realistically consider a bad actor will sybil attack well known and highly valued NFT collection items. This means that the lender will have a bad valuation and lend to someone with fake collateral that is worth nothing.

A second example would be that any kind of unknown attack may be run through this contract as an attacker can set up a new loan with his own contract in place of a regular NFT contract.

1.3 Recommendations

Similar to the validation used for the settlementToken, maintaining an admin whitelist of NFT collections allowed to be used as collateral.

1.4 Remediation Completed

Issue has been fixed by Nau Finance using the NFT whitelist.

2 Weak Validation of Loan Duration [Medium]

2.1 Description

Currently when calling requestNewLoan() there is no validation at all on the end date of the loan.

```
195   function requestNewLoan(
196     address collateralToken,
197     uint256 collateralTokenID,
198     IERC20Metadata settlementToken,
199     uint256 minAmount,
200     uint256 maxAmount,
201     uint256 minAPRRay,
202     uint256 maxAPRRay,
203     uint40 endDate,
204     uint24 fundingPeriod
205   ) external whenNotPaused returns (uint256) {
```

Extract of CocktailLoan.sol

2.2 Impact

In order to protect the lender, it may be safer to set a maximum duration for a loan (relative to the start date). If a lender misses this detailed information they may lend to someone for an infinite duration and never get any interest or any funds back.

2.3 Recommendations

Similar to the validation used for the fundingPeriod and the minFundingPeriod, make sure startDate to endDate duration stay within a predefined maximum range.

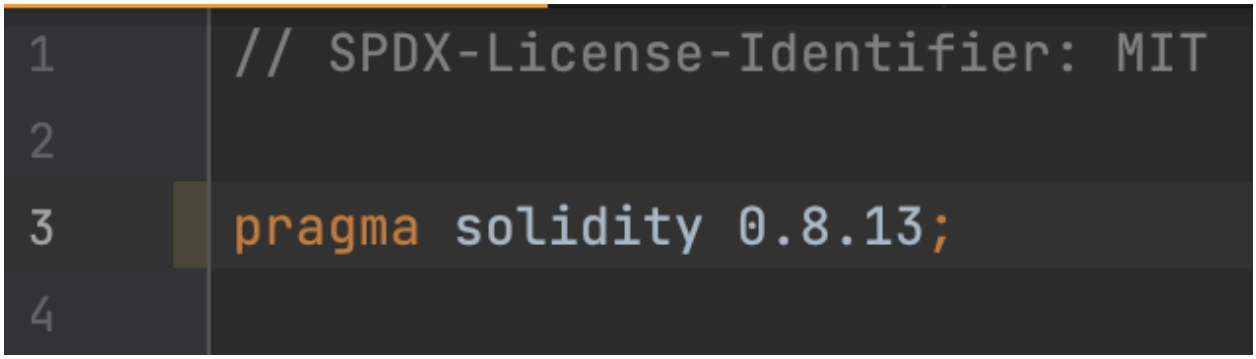
2.4 Remediation Completed

Issue has been fixed by Nau Finance with the implementation of maxLoanDurationDays.

3 Outdated Solidity Compiler Version Used [Low]

3.1 Description

Currently an outdated Solidity version has been used on the contracts.



Extract of CocktailLoan.sol

3.2 Impact

Compiler issues can be caused by using outdated versions of Solidity. Alongside this, using different compiler versions will produce arbitrary outputs and lead to unexpected issues.

3.3 Recommendations

To remediate this issue, it is recommended to use the latest version of the Solidity compiler

3.4 Remediation Completed

Issue has been acknowledged by the client who deemed no further action was required.

Remediations Summary

Successfully remediated

- Weak Validation of Collateral NFT Token (Medium)
- Weak Validation of Loan Duration (Medium)

Acknowledged but no further action taken

- Outdated Solidity Compiler Version Used (Low)

HYDN Seal

Upon successful completion of a Smart Contract Audit from HYDN, a project is provided with a HYDN Seal.

Each HYDN Seal is an ERC-1155 token that contains key metadata about the completed audit, including the hash of the contract bytecode, date, and link to the report.

As each HYDN Seal is stored on-chain, it allows for real-time monitoring of a project's status. The image on the Seal is served dynamically. We will notify you if the contract changes so you can validate if it has been changed by nefarious activity.

The HYDN Seal Token ID assigned to the contracts covered by this report: *TBC*

The Token Factory contract can be found at: *TBC*