Smart Contract

Security Assessment

For NFTULOAN

6 July 2022

Ascendant

@ascendantproj

www.ascendant.finance

Table of Contents

DISCLAIMER

Ascendant Finance ("Ascendant") has conducted an independent audit to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the codes that were provided for the scope of this audit. This audit report does not constitute agreement, acceptance or advocation for the Project that was audited, and users relying on this audit report should not consider this as having any merit for financial advice in any shape, form or nature. The contracts audited do not account for any economic developments that may be pursued by the Project in question, and that the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are completely free of exploits, bugs, vulnerabilities or deprecation of technologies. Further, this audit report shall not be disclosed nor transmitted to any persons or parties on any objective, goal or justification without due written assent, acquiescence or approval by Ascendant.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any persons reading this report should invest their funds without sufficient individual due diligence regardless of the findings presented in this report. Information is provided 'as is', and Ascendant is under no covenant to the completeness, accuracy or solidity of the contracts audited. In no event will Ascendant or its partners, employees, agents or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions and/or actions with regards to the information provided in this audit report.

Cryptocurrencies and any technologies by extension directly or indirectly related to cryptocurrencies are highly volatile and speculative by nature. All reasonable due diligence and safeguards may yet be insufficient, and users should exercise considerable caution when participating in any shape or form in this nascent industry.

The audit report has made all reasonable attempts to provide clear and articulate recommendations to the Project team with respect to the rectification, amendment and/or revision of any highlighted issues, vulnerabilities or exploits within the contracts provided. It is the sole responsibility of the Project team to sufficiently test and perform checks, ensuring that the contracts are functioning as intended, specifically that the functions therein contained within said contracts have the desired intended effects, functionalities and outcomes of the Project team.

Ascendant retains full rights over all intellectual property (including expertise and new attack or exploit vectors) discovered during the audit process. Ascendant is therefore allowed and expected to re-use this knowledge in subsequent audits and to inform existing projects that may have similar vulnerabilities. Ascendant may, at its discretion, claim bug bounties from third-parties while doing so.

Executive Summary

Severity	Found
High	2
Medium	0
Low	5
Informational	53
Total	60

We performed an independent technical audit to identify Smart Contracts uncertainties. This shall protect the code from illegitimate authorization attempts or external & internal threats of any type. This also ensures end-to-end proofing of the contract from frauds. The audit was performed semi-manually. We analyzed the Smart Contracts code line-by-line and used an automation tool to report any suspicious code.

The following tools were used:

- Truffle
- Remix IDE
- Slither

Overview

This report has been prepared for NFTULoan on the Ethereum network. Ascendant provides a user-centered examination of the smart contracts to look for vulnerabilities, logic errors or other issues from both an internal and external perspective.

Summary

Project Name	NFTULoan
Platform	Ethereum
Language	Solidity

Contracts Assessed

Name	Location
ULoanContract.sol	Not deployed
iNFT.sol	In ULoanContract
Ownable.sol	In ULoanContract
IERC721Receiver.sol	In ULoanContract
ReentrancyGuard.sol	In ULoanContract
Context.sol	In ULoanContract

Findings Summary

Severity	Found
High	2
Medium	0
Low	5
Informational	53
Total	60

Classification of Issues

High	Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, or impairment of the contract and its functions. Issues under this classification are recommended to be fixed with utmost urgency.
Medium	Bugs or issues that may be subject to exploit, though their impact is somewhat limited. Issues under this classification are recommended to be fixed as soon as possible.
Low	Effects are minimal in isolation and do not pose a significant danger to the project or its users. Issues under this classification are recommended to be fixed nonetheless.
Informational	Consistency, syntax or style best practices, Generally pose a negligible level of risk, if any.

Manual Review

Issues Checking Status

Issue Description	Checking Status
Compiler errors	PASS
Race conditions and Reentrancy. Cross-function race conditions.	PASS
Possible delays in data delivery.	PASS
Oracle calls.	PASS
Front running.	PASS
Timestamp dependence.	PASS
Integer Overflow and Underflow.	PASS
DoS with Revert.	PASS
DoS with block gas limit.	PASS
Methods execution permissions.	PASS
Economy model of the contract.	PASS
The impact of the exchange rate on the logic.	PASS
Private user data leaks.	PASS
Malicious Event log.	PASS
Scoping and Declarations.	PASS
Uninitialized storage pointers.	PASS

Arithmetic accuracy.	PASS
Design Logic.	PASS
Cross-function race conditions.	PASS
Safe Open Zeppelin contracts implementation and usage.	PASS
Fallback function security.	PASS

Audit Findings

Severity	High
Contract	UContractLoan.sol
Description	Strict equality
Code Snippet	817: require(msg.value == priceToPay, "Bad amount");
Recommendation	Replace "==" with ">=" Strict equalities should be avoided due to rounding errors that can occur when sending a specific amount.
Status	AMENDED

Severity	High
Contract	UContractLoan.sol
Description	Strict Equality
Code Snippet	874: require(msg.value==loan.resalesPrice, "Invalid price");
Recommendation	Replace "==" with ">=" Strict equalities should be avoided due to rounding errors that can occur when sending a specific amount.
Status	AMENDED

Severity	**Low**
Contract	UContractLoan.sol
Description	Lack of Zero Check

Code Snippet	function setAdmin(address newAdmin) public onlyOwner {
	 address previousAdmin = _admin;
	 _admin = newAdmin;
	
	 emit AdminChanged(previousAdmin, newAdmin);
	 }

Recommendation	Add a require statement that requires the new owner address to not be address(0), or the zero address.
Status	AMENDED

Severity	**Low**
Contract	UContractLoan.sol
Description	Lack of Zero Check

Code Snippet	function setServiceWallet(address newAddr)
	external onlyOwner {
	address oldWallet = serviceWallet;
	serviceWallet = newAddr;
	
	emit SetServiceWallet(oldWallet, newAddr);
	}

Recommendation	Add a require statement that requires the new owner address to not be address(0), or the zero address.
Status	AMENDED

Functional Test Status

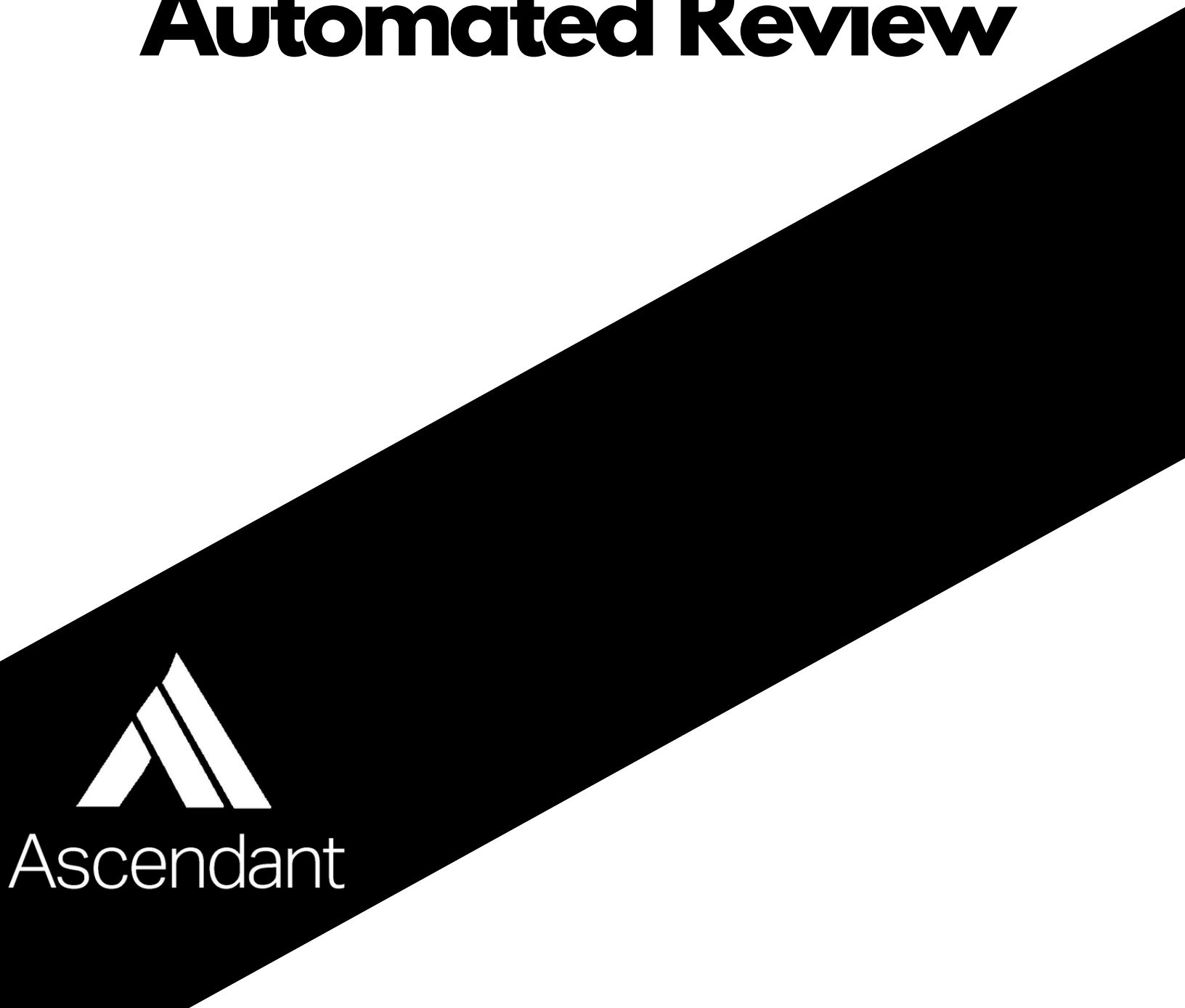
Function Name	Type/Return Type	Score
loanAfterLockingInSec	private	PASS
minLenderLockDurationInSec	private	PASS
minFundingInWei	private	PASS
serviceWallet	private	PASS
signHeader	private	PASS
proposedHashes	private	PASS
lentNfts	private	PASS
marketSalesFeeInM100	read/public	PASS
loanableFund	read/public	PASS
serviceFeePercentInM100	read/public	PASS
interestFeePercentInM100	read/public	PASS
maxLoanablePercentInM100	read/public	PASS
withdrawLockingDuration	read/public	PASS
operators	read/public	PASS
registeredOperators	read/public	PASS
registeredNftCollections	read/public	PASS
nftCollections	read/public	PASS
registeredLenders	read/public	PASS

operatorList	read/public	PASS
lenderList	read/public	PASS
lenderFundList	read/public	PASS
withdrawList	read/public	PASS
loanList	read/public	PASS
loanFunds	read/public	PASS
tagFundsForLoanCreation	internal	PASS
setNftCollectionEx	internal	PASS
createLenderAndAddFund	payable/public	PASS
addFund	payable/public	PASS
closeLoan	payable/public	PASS
saleLoanOnMarket	payable/public	PASS
listOperators	read/public	PASS
getOperatorCount	read/public	PASS
setServiceWallet	write/public	PASS
setFee	write/public	PASS
setResaleFee	write/public	PASS
getMaxLoanablePercent	read/public	PASS
setMaxLoanablePercent	write/public	PASS
askToUnlockFund	write/public	PASS

withdrawFund	write/public	PASS
getWithdrawCount	read/public	PASS
getWithdrawLockingDuration	read/public	PASS
setWithdrawLockingDuration	write/public	PASS
setLenderAutoLoaningMode	write/public	PASS
lenderExists	read/public	PASS
getLenderIndexByAddress	read/public	PASS
getLenderIndexesByAddress	read/public	PASS
getLendersFundCount	read/public	PASS
isAmountLoanable	read/public	PASS
createLoan	write/public	PASS
setNftCollections	write/public	PASS
isNftCollectionEnabled	read/public	PASS
sendBackUnlentNft	write/public	PASS
setLoanAsAHotdeal	write/public	PASS
changeLoanPriceOnMarket	write/public	PASS
getLender	read/public	PASS
getLenderCount	read/public	PASS
listLendersByIndexes	read/public	PASS
listLendersFunds	read/public	PASS
listFunds	read/public	PASS

getLoan	read/public	PASS
getLoanCount	read/public	PASS
listLoans	read/public	PASS
getWithdraw	read/public	PASS
listWithdraws	read/public	PASS
getEthBalance	read/public	PASS
getLoanableFund	write/public	PASS
getMinFundinginWei	write/public	PASS
setOperator	write/public	PASS
checkIfNftAlreadyTransfered	read/public	PASS
TLender	read/public	PASS
TFund	read/public	PASS
TWithdraw	read/public	PASS
TLoanLenderFund	read/public	PASS
TLoan	read/public	PASS
TNftCollection	read/public	PASS
TOperator	read/public	PASS

Automated Review



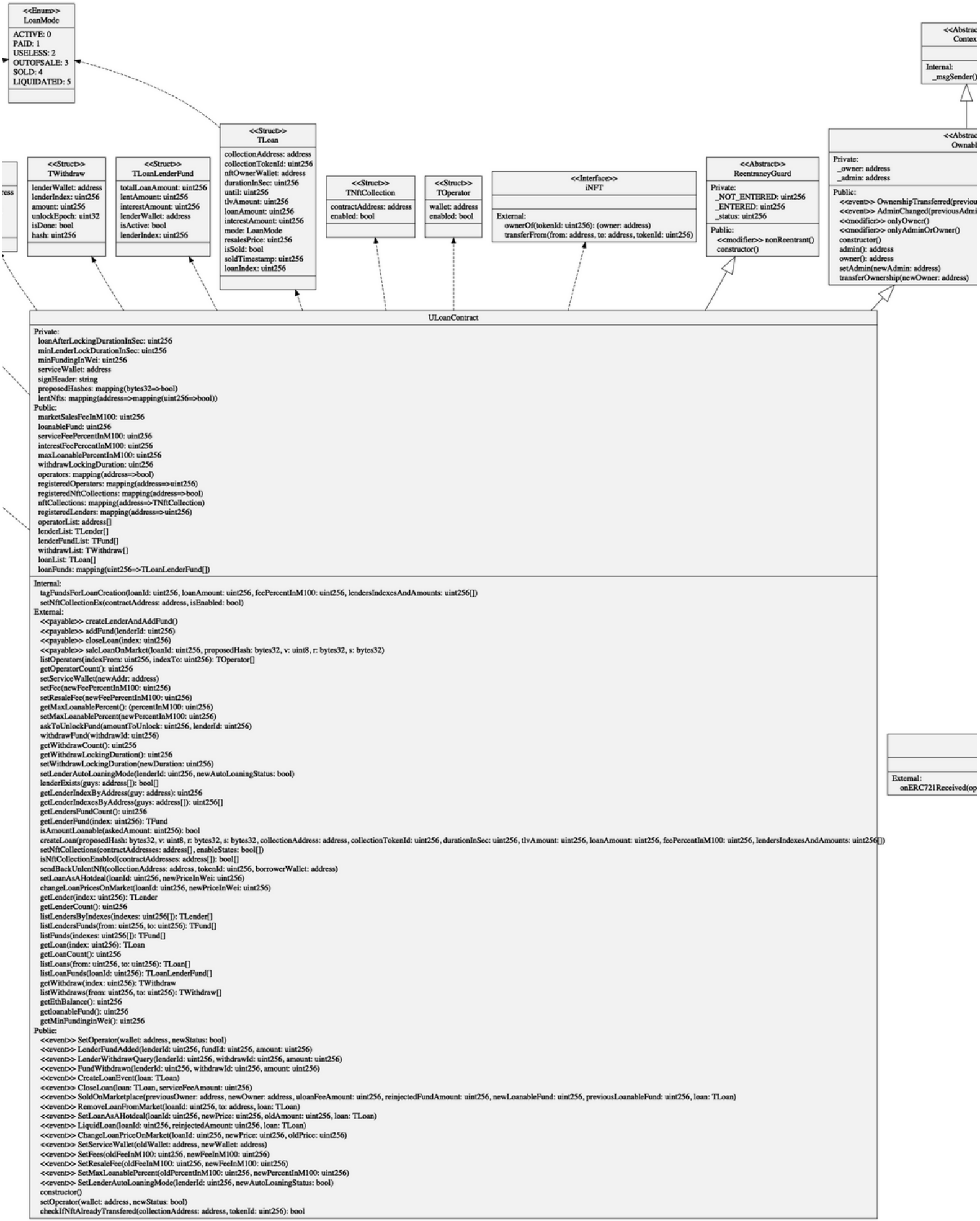
Ascendant

Solidity Static Analysis

Issue	Severity
Check-effects-interaction: **NOTE:** *All flags for checks-effects-interactions have been downgraded from MEDIUM to LOW due to the utilization of Reentrancy Guard.* Potential violation of Checks-Effects-Interaction pattern in ULoanContract.withdrawFund(uint256): Could potentially lead to re-entrancy vulnerability. Pos: 478:8:	Low
Check-effects-interaction: NOTE: All flags for checks-effects-interactions have been downgraded from MEDIUM to LOW due to the utilization of Reentrancy Guard. Potential violation of Checks-Effects-Interaction pattern in ULoanContract.createLoan(bytes32,uint8,bytes32,bytes32,address,uint256,uint256,uint256,uint256,uint256,uint256[]): Could potentially lead to re-entrancy vulnerability. Pos: 610:8:	Low
Check-effects-interaction: **NOTE:** *All flags for checks-effects-interactions have been downgraded from MEDIUM to LOW due to the utilization of Reentrancy Guard.* Potential violation of Checks-Effects-Interaction pattern in ULoanContract.closeLoan(uint256): Could potentially lead to re-entrancy vulnerability. Pos: 717:8:	Low

Check-effects-interaction: **NOTE:** *All flags for checks-effects-interactions have been downgraded from MEDIUM to LOW due to the utilization of Reentrancy Guard.* Potential violation of Checks-Effects-Interaction pattern in ULoanContract.saleLoanOnMarket(uint256,bytes32,uint8,bytes32,bytes32): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis. Pos: 845:8:	Low
For loop over dynamic array: Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.	Low
Block.timestamp Dangerous Comparisons: Any function that utilizes block.timestamp is vulnerable to miner attack but usually only if block.timestamp is used to affect the output of the function (e.g. randomness). The following functions utilize block.timestamp: • getLenderFund • closeLoan • saleLoanOnMarket • setLoanAsAHotdeal • changeLoanPricesOnMarket • getLender • listLendersByIndexes • listFunds • getLoan • listLoans • getWithdraw • listWithdraws	Informational

Unified Modeling Language(UML)



Conclusion

The smart contracts reviewed in this audit contain no critical severity issues and all High to Medium issues have either been corrected or acknowledged.

Please check the disclaimer above and note, the audit makes no statements or warranties on business model, investment attractiveness or code sustainability. The report is provided for the only contract mentioned in the report and does not include any other potential contracts deployed by Owner.

Ascendant

Ascendant

@ascendantproj

www.ascendant.finance

Ascendant