HALBORN

NFTfi - DirectLoanFixedOffer Redeployment

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **October 31st, 2022 - November 7th, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	11/7/2022	Miguel Jalon
0.2	Draft Review	11/10/2022	Kubilay Onur Gungor
0.3	Draft Review	11/10/2022	Gabi Urrutia
1.0	Remediation Plan	11/14/2022	Miguel Jalon
1.1	Remediation Plan Review	01/12/2022	Roberto Reigada
1.2	Remediation Plan Review	01/12/2022	Piotr Cielas
1.3	Remediation Plan Review	01/12/2022	Gabi Urrutia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
Miguel Jalon	Halborn	Miguel.Jalon@halborn.com
Roberto Reigada	Halborn	Roberto.Reigada@halborn.com
Piotr Cielas	Halborn	Piotr.Cielas@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

NFTfi engaged Halborn to conduct a security audit on their smart contracts beginning on October 31st, 2022 and ending on November 7th, 2022. The security assessment was scoped to the smart contracts provided to the Halborn team.

1.2 AUDIT SUMMARY

The team at Halborn was provided one week for the engagement and assigned a full-time security engineer to audit the security of the smart contract. The security engineer is a blockchain and smart-contract security expert with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The purpose of this audit is to:

- Ensure that smart contract functions operate as intended
- Identify potential security issues with the smart contracts

In summary, Halborn identified some security risks that were mostly addressed by the NFTfi team.

1.3 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of this audit. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of the bridge code and can quickly identify items that do not follow security best practices. The following phases and associated tools were used throughout the term of the audit:

- Research into architecture and purpose
- Smart contract manual code review and walkthrough
- Graphing out functionality and contract logic/connectivity/functions (solgraph)
- Manual assessment of use and safety for the critical Solidity variables and functions in scope to identify any arithmetic related vulnerability classes
- Manual testing by custom scripts
- Scanning of solidity files for vulnerabilities, security hotspots or bugs. (MythX)
- Static Analysis of security for scoped contract, and imported functions. (Slither)
- Testnet deployment (Brownie, Remix IDE)

RISK METHODOLOGY:

Vulnerabilities or issues observed by Halborn are ranked based on the risk assessment methodology by measuring the **LIKELIHOOD** of a security incident and the **IMPACT** should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. The quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that were used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

RISK SCALE - LIKELIHOOD

5 - Almost certain an incident will occur.
4 - High probability of an incident occurring.
3 - Potential of a security incident in the long term.
2 - Low probability of an incident occurring.
1 - Very unlikely issue will cause an incident.

RISK SCALE - IMPACT

5 - May cause devastating and unrecoverable impact or loss.
4 - May cause a significant level of impact or loss.

3 - May cause a partial impact or loss to many.
2 - May cause temporary impact or loss.
1 - May cause minimal or un-noticeable impact.

The risk level is then calculated using a sum of these two values, creating a value of 10 to 1 with 10 being the highest level of security risk.

10 - CRITICAL
9 - 8 - HIGH
7 - 6 - MEDIUM
5 - 4 - LOW
3 - 1 - VERY LOW AND INFORMATIONAL

1.4 SCOPE

IN-SCOPE:
The security assessment was scoped to the following smart contracts:

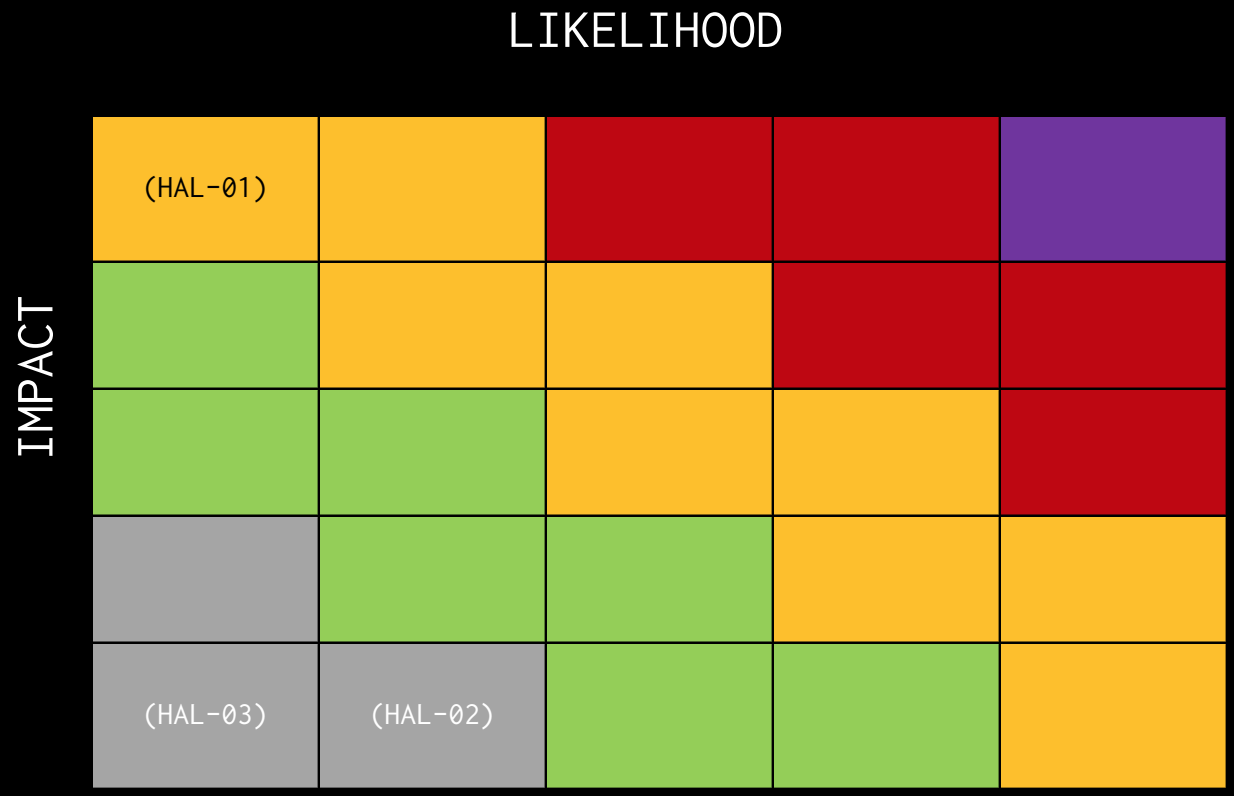
- DirectLoanFixedOfferRedeploy.sol

So then, they were also reviewed:
- DirectLoanFixedOffer.sol
- DirectLoanBaseMinimal.sol
- DirectLoanBase.sol
- PermittedERC20s.sol
- BaseLoan.sol
- NftReceiver.sol
- LoanData.sol

Commit ID: 0e20d31354e394d35f1550becbd0990f85023dbd

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	1	0	2



EXECUTIVE OVERVIEW

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL01 - POSSIBLE LOSS OF OWNERSHIP	Medium	RISK ACCEPTED
HAL02 - ZERO ADDRESS NOT CHECKED	Informational	ACKNOWLEDGED
HAL03 - USE I++ INSTEAD OF ++I IN LOOPS FOR GAS OPTIMIZATION	Informational	ACKNOWLEDGED

FINDINGS & TECH DETAILS

3.1 (HAL-01) POSSIBLE LOSS OF OWNERSHIP - MEDIUM

Description:

When transferring ownership of the protocol, no checks are performed on whether the new address is valid and active. In case there is a mistake when transferring the ownership, the whole protocol is locked out of its permissioned functionalities.

Code Location:

Listing 1: Ownable.sol

```
48     function transferOwnership(address _newOwner) public virtual
↳ onlyOwner {
49         require(_newOwner != address(0), "Ownable: new owner is
↳ the zero address");
50         _setOwner(_newOwner);
51     }
```

Risk Level:

Likelihood - 1
Impact - 5

Recommendation:

The transfer of ownership process should be split into two different transactions, the first one calling the requestTransferOwnership function which proposes a new owner for the protocol, and the second one, the new owner accepts the proposal by calling acceptsTransferOwnership function.

Remediation Plan:

RISK ACCEPTED: The NFTfi team accepted the risk of this finding. The team most likely will implement the remediation when they upgrade the platform.

3.2 (HAL-02) ZERO ADDRESS NOT CHECKED - INFORMATIONAL

Description:

In the constructor of the DirectLoanFixedOfferRedeploy.sol contract, admin and nfthub contract address variables are not being checked to avoid pointing to the zero address, extending this issue to the parent contracts.

Code Location:

Listing 2: DirectLoanFixedOfferRedeploy.sol

```
72     constructor(
73         address _admin,
74         address _nftfiHub,
75         address[] memory _permittedErc20s
76     ) DirectLoanFixedOffer(_admin, _nftfiHub, _permittedErc20s) {
77         // solhint-disable-previous-line no-empty-blocks
78     }
```

Risk Level:

Likelihood - 2
Impact - 1

Recommendation:

When setting an address variable, always make sure the value is not zero.

Remediation Plan:

ACKNOWLEDGED: The NFTfi team acknowledged this issue.

3.3 (HAL-03) USE I++ INSTEAD OF ++I IN LOOPS FOR GAS OPTIMIZATION - INFORMATIONAL

Description:

In the setERC20Permits function, within the loop, the variable i is incremented using i++. It is known that, in loops, using ++i costs less gas per iteration than i++. This also affects variables incremented inside the loop code block.

Code Location:

Listing 3: DirectLoanBaseMinimal.sol (Line 374)

```
371     function setERC20Permits(address[] memory _erc20s, bool[]
 ↳ memory _permits) external onlyOwner {
372         require(_erc20s.length == _permits.length, "
 ↳ setERC20Permits function information arity mismatch");
373
374         for (uint256 i = 0; i < _erc20s.length; i++) {
375             _setERC20Permit(_erc20s[i], _permits[i]);
376         }
377     }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to use ++i instead of i++ to increment the value of a uint variable inside a loop for gas saving.

Remediation Plan:

ACKNOWLEDGED: The NFTfi team acknowledged this issue.

MANUAL TESTING

In the manual testing phase, the following scenarios were simulated. The scenarios listed below were selected based on the severity of the vulnerabilities Halborn was testing the program for.

4.1 SCENARIOS TESTED

- Test 1: Lending - Borrow - normal repay procedure
- Test 2: Lending - Borrow and repay Out Of Time (revert expected)
- Test 3: Lending - Borrow - Out Of Time - normal liquidation procedure
- Test 4: Lending - Borrow and liquidate ahead of schedule (revert expected)
- Test 5: Lending - Borrow - Out of time - Renegotiate - normal repay procedure
- Test 6: Attack: Borrower tries to renegotiate their own loan to steal the NFT (revert expected)

Script The following test environment was set up for the purposes of executing the above scenarios:

Listing 4: NFTFi.t.sol

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity 0.8.4;

import "forge-std/Test.sol";
import "../src/contracts/NftfiHub.sol";
import "../src/contracts/loans/direct/loanTypes/
↳ DirectLoanFixedOfferRedeploy.sol";
import "@openzeppelin/contracts/utils/cryptography/ECDSA.sol";
import "../src/contracts/mocks/NFT.sol";
import "../src/contracts/mocks/NFTWrapper.sol";
import "../src/contracts/loans/direct/DirectLoanCoordinator.sol";
import "../src/contracts/mocks/SimpleERC20.sol";
import "../src/contracts/permittedLists/
↳ PermittedNFTsAndTypeRegistry.sol";
import "../src/contracts/loans/direct/loanTypes/LoanData.sol";

contract NftfiTest is Test {
```

```
16      using ECDSA for bytes32;
17
18      NFT internal nftContract;
19      NFTWrapper internal nftWrapper;
20      SmartNft internal nftPromissoryNote;
21      SmartNft internal nftObligationReceipt;
22      SimpleToken internal token;
23      NftfiHub internal nftFiHub;
24      DirectLoanCoordinator internal directLoanCoordinator;
25      DirectLoanFixedOfferRedeploy internal
↳ directLoanFixedOfferRedeploy;
26      PermittedNFTsAndTypeRegistry internal
↳ permittedNFTsAndTypeRegistry;
27
28      address internal owner;
29      address internal admin;
30      address internal alice;
31      address internal bobby;
32      address internal carla;
33      address internal edgar;
34      address internal zeroo;
35      uint256 internal verifyingSignerPrivateKey;
36      address internal verifyingSigner;
37      address internal dappIdentifier1;
38      address internal sender;
39      address[] internal permittedErc20s;
40      string[] internal contractKeys;
41      address[] internal contractAddresses;
42      string[] internal definedNftTypes;
43      address[] internal definedNftWrappers;
44      address[] internal permittedNftContracts;
45      string[] internal permittedNftTypes;
46      uint256 internal timeNow = block.timestamp;
47      uint256 internal day = 86400;
48      string[] internal loanTypes;
49      address[] internal loanContracts;
50      bool internal liquidated;
51
52
53
54      function setUp() public {
55
56          /* *************** */
57          /* ADDRESSES SETUP */
```

```
58          /* *************** */
59
60          // ADDRESSES DECLARATION
61          owner = vm.addr(0xAA);
62          alice = vm.addr(0xAB);
63          bobby = vm.addr(0xAC);
64          carla = vm.addr(0xAD);
65          edgar = vm.addr(0xAE);
66          zeroo = address(0);
67          admin = owner;
68
69          // 100 ETHER PER ADDRESS
70          vm.deal(owner, 100 ether);
71          vm.deal(alice, 100 ether);
72          vm.deal(bobby, 100 ether);
73          vm.deal(carla, 100 ether);
74          vm.deal(edgar, 100 ether);
75
76          // LABELING ADDRESSES
77          vm.label(owner, "owner");
78          vm.label(alice, "alice");
79          vm.label(bobby, "bobby");
80          vm.label(carla, "carla");
81          vm.label(edgar, "edgar");
82
83          /* ***************** */
84          /* ENVIRONMENT SETUP */
85          /* ***************** */
86
87          // DEPLOYING NFTWRAPPER
88          vm.prank(admin);
89          nftWrapper = new NFTWrapper();
90
91          // DEPLOYING NFT CONTRACT
92          vm.prank(admin);
93          nftContract = new NFT(address(nftWrapper));
94
95          // MINT 5 NFTS TO ALICE
96          vm.prank(alice);
97          nftContract.mintNFT(5);
98
99          // DEPLOYING AND MINTING TOKEN
100         vm.prank(admin);
```

```
101         token = new SimpleToken("token", "TKN", 1000000
 ↳ _000000000000000000);
102         vm.prank(admin);
103         token.transfer(alice, 50_000000000000000000);
104         vm.prank(admin);
105         token.transfer(bobby, 1000_000000000000000000);
106         vm.prank(admin);
107         token.transfer(carla, 250_000000000000000000);
108 
109         // LOGS
110         console.log("****** SETTING ENVIRONMENT ******");
111         console.log("****** STATE 1 *******");
112         console.log("****** BALANCES ******");
113         console.log("Balance Of Admin        ---> " ,token.
 ↳ balanceOf(admin));
114         console.log("Balance Of Alice        ---> " ,token.
 ↳ balanceOf(alice));
115         console.log("Balance Of Bobby        ---> " ,token.
 ↳ balanceOf(bobby));
116         console.log("Balance Of Carla        ---> " ,token.
 ↳ balanceOf(carla));
117         console.log("Owner of the NFT        ---> " ,nftContract.
 ↳ ownerOf(1));
118         console.log(" ");
119         console.log(" ");
120 
121         // PUSHING KEYS AND ADDRESSES OF NFT AND TOKEN CONTRACTS
122         contractKeys.push('PERMITTED_NFTS');
123         contractAddresses.push(address(nftContract));
124         permittedErc20s.push(address(token));
125 
126         // DEPLOYING NFT HUB
127         nftFiHub = new NftfiHub(admin, contractKeys,
 ↳ contractAddresses);
128         address nftFiHubAddr = address(nftFiHub);
129 
130         // DEPLOYING PROMISORY NOTES AND OBLIGATION RECEPT
 ↳ CONTRACTS
131         nftObligationReceipt = new SmartNft(admin, address(
 ↳ nftFiHub), address(directLoanCoordinator), "nftObligationReceipt",
 ↳  "NOR", "customURI");
132         nftPromissoryNote = new SmartNft(admin, address(nftFiHub),
 ↳  address(directLoanCoordinator), "nftPromissoryNote", "NOR", "
 ↳ customURI");
```

```
133         vm.prank(admin);
134         nftObligationReceipt.setLoanCoordinator(address(
 ↳ directLoanCoordinator));
135         vm.prank(admin);
136         nftPromissoryNote.setLoanCoordinator(address(
 ↳ directLoanCoordinator));
137 
138         // DEPLOYING DIRECT_LOAN_FIXED_OFFER_REDEPLOY
139         directLoanFixedOfferRedeploy = new
 ↳ DirectLoanFixedOfferRedeploy(admin, nftFiHubAddr, permittedErc20s)
 ↳ ;
140         address directLoanFixedOfferRedeployAddr = address(
 ↳ directLoanFixedOfferRedeploy);
141 
142         // DEPLOYING DIRECT LOAN COORDINATOR
143         loanTypes.push("DIRECT_LOAN_FIXED_REDEPLOY");
144         loanContracts.push(address(directLoanFixedOfferRedeploy));
145         directLoanCoordinator = new DirectLoanCoordinator(address(
 ↳ nftFiHub), admin, loanTypes, loanContracts);
146 
147         // INITIALIZING DIRECT LOAN COORDINATOR
148         directLoanCoordinator.initialize(address(nftPromissoryNote
 ↳ ), address(nftObligationReceipt));
149         vm.prank(admin);
150         nftObligationReceipt.setLoanCoordinator(address(
 ↳ directLoanCoordinator));
151         vm.prank(admin);
152         nftPromissoryNote.setLoanCoordinator(address(
 ↳ directLoanCoordinator));
153 
154         // SETTING CONTRACTKEYS AND CONTRACTADDRRESSES
155         contractKeys.push('DIRECT_LOAN_COORDINATOR');
156         contractAddresses.push(address(directLoanCoordinator));
157         vm.prank(admin);
158         nftFiHub.setContract('DIRECT_LOAN_COORDINATOR', address(
 ↳ directLoanCoordinator));
159 
160         // REGISTRATION OF PERMITTED NFTS AND TYPES
161         definedNftTypes.push('ERC721');
162         definedNftWrappers.push(address(nftWrapper));
163         permittedNftContracts.push(address(nftContract));
164         permittedNftTypes.push('ERC721');
165         permittedNFTsAndTypeRegistry = new
 ↳ PermittedNFTsAndTypeRegistry(admin, nftFiHubAddr, definedNftTypes,
```

```
 ↳  definedNftWrappers, permittedNftContracts, permittedNftTypes);
166
167         // APPROVALS FOR PUT NFTS AS COLATERAL
168         vm.prank(alice);
169         nftContract.setApprovalForAll(address(
 ↳ directLoanFixedOfferRedeployAddr), true);
170     }
```

TEST 1:

Script

Listing 5: NFTFi.t.sol

```
174 function test_1() public {
175
176     /* ****************************************** */
177     /* TEST 1: LENDING BORROWING NORMAL PROCEDURE */
178     /* ****************************************** */
179
180     console.log("******************************");
181     console.log("**********  TEST 1  **********");
182     console.log("******************************");
183     console.log(" ");
184
185     // DECLARING OFFER
186     LoanData.Offer memory offer = LoanData.Offer({
187         loanPrincipalAmount: 10_000000000000000000,
188         maximumRepaymentAmount: 12_000000000000000000,
189         nftCollateralId: 1,
190         nftCollateralContract: address(nftContract),
191         loanDuration: 10 days,
192         loanAdminFeeInBasisPoints: 500,
193         loanERC20Denomination: address(token),
194         referrer: zeroo
195     });
196
197     // TOKEN APPROVAL
198     vm.prank(bobby);
199     token.approve(address(directLoanFixedOfferRedeploy), 10
 ↳ _000000000000000000);
200
201     // GETTING CHAIN ID
202     uint256 id;
203     assembly {
204         id := chainid()
205     }
206
207     // PREPARING SIGNATURE STRUCT
208     LoanData.Signature memory signature = LoanData.Signature({
209         nonce: 1,
210         expiry: timeNow + 10 days,
211         signer: bobby,
212         signature: hex"1c"
```

```
213     });
214
215     // GETTING THE MESSAGE HASH
216     bytes32 message = keccak256(
217         abi.encodePacked(getEncodedOffer(offer),
 ↳ getEncodedSignature(signature), address(
 ↳ directLoanFixedOfferRedeploy), id)
218     );
219
220     // EIP712 STANDARD
221     bytes32 signedMessage = ECDSA.toEthSignedMessageHash(message);
222
223     // GETTING THE V, R, S OF THE SIGNED MESSAGE
224     (uint8 v, bytes32 r, bytes32 s) = vm.sign(0xAC, signedMessage)
 ↳ ;
225     bytes memory v_bytes;
226     if(v == 27){v_bytes = hex"1b";} else {v_bytes = hex"1c";}
227     bytes memory signaturesf = bytes.concat(r, s, v_bytes);
228
229     // BOBBY SIGNES
230     LoanData.Signature memory signaturefi = LoanData.Signature({
231         nonce: 1,
232         expiry: timeNow + 10 days,
233         signer: bobby,
234         signature: signaturesf
235     });
236
237     // ALICE ACCEPTS BOBBY'S OFFER
238     LoanData.BorrowerSettings memory borrowerSettings;
239     vm.prank(alice);
240     directLoanFixedOfferRedeploy.acceptOffer(offer, signaturefi,
 ↳ borrowerSettings);
241
242     // CHECKING THAT THE STATE IS AS EXPECTED
243     (uint256 loanPrincipalAmount, uint256 maximumRepaymentAmount,
 ↳ uint256 nftCollateralId, address loanERC20Denomination, uint32
 ↳ loanDuration, uint32 loanInterestRateForDuration, uint16
 ↳ loanAdminFeeInBasisPoints, address nftCollateralWrapper, uint64
 ↳ loanStartTime, address nftCollateralContract, address borrower) =
 ↳ directLoanFixedOfferRedeploy.loanIdToLoan(1);
244     liquidated = directLoanFixedOfferRedeploy.
 ↳ loanRepaidOrLiquidated(1);
245
246     // LOGS
```

```
247     console.log("TX: ALICE ---> ACCEPT BOBBY'S OFFER");
248     console.log(" ");
249     console.log("****** STATE 1 *******");
250     console.log("****** BALANCES ******");
251     console.log("Balance Of Admin        ---> " ,token.balanceOf(
 ↳ admin));
252     console.log("Balance Of Alice        ---> " ,token.balanceOf(
 ↳ alice));
253     console.log("Balance Of Bobby        ---> " ,token.balanceOf(
 ↳ bobby));
254     console.log("Balance Of Carla        ---> " ,token.balanceOf(
 ↳ carla));
255     console.log("Owner of the NFT        ---> " ,nftContract.
 ↳ ownerOf(1));
256     console.log(" ");
257     console.log("****** LOAN DATA ******");
258     console.log("loanPrincipalAmount        ---> " ,
 ↳ loanPrincipalAmount);
259     console.log("maximumRepaymentAmount     ---> " ,
 ↳ maximumRepaymentAmount);
260     console.log("nftCollateralId            ---> " ,
 ↳ nftCollateralId);
261     console.log("loanERC20Denomination      ---> " ,
 ↳ loanERC20Denomination);
262     console.log("loanDuration               ---> " ,loanDuration);
263     console.log("interestRateForDuration    ---> " ,
 ↳ loanInterestRateForDuration);
264     console.log("loanAdminFeeInBasisPoints  ---> " ,
 ↳ loanAdminFeeInBasisPoints);
265     console.log("nftCollateralWrapper       ---> " ,
 ↳ nftCollateralWrapper);
266     console.log("loanStartTime              ---> " ,loanStartTime)
 ↳ ;
267     console.log("nftCollateralContract      ---> " ,
 ↳ nftCollateralContract);
268     console.log("borrower                   ---> " ,borrower);
269     console.log("LOAN LIQUIDATED / REPAYED  ---> " ,liquidated);
270     console.log(" ");
271
272     // 5 DAYS LATER
273     console.log("5 DAYS LATER...");
274     vm.warp(5 * day);
275
276     // ALICE PAY THE MONEY
```

```
277     vm.prank(alice);
278     token.approve(address(directLoanFixedOfferRedeploy), 12
 ↳ _000000000000000000);
279     vm.prank(alice);
280     directLoanFixedOfferRedeploy.payBackLoan(1);
281
282
283     // CHECKING THAT THE STATE IS AS EXPECTED
284     (uint256 a, uint256 b, uint256 c, address d, uint32 e, uint32
 ↳ f, uint16 g, address h, uint64 i, address j, address k) =
 ↳ directLoanFixedOfferRedeploy.loanIdToLoan(1);
285     liquidated = directLoanFixedOfferRedeploy.
 ↳ loanRepaidOrLiquidated(1);
286     console.log("TX: ALICE ---> PAY BACK LOAN");
287     console.log(" ");
288     console.log("****** STATE 3 *******");
289     console.log("****** BALANCES ******");
290     console.log("Balance Of Admin        ---> " ,token.balanceOf(
 ↳ admin));
291     console.log("Balance Of Alice        ---> " ,token.balanceOf(
 ↳ alice));
292     console.log("Balance Of Bobby        ---> " ,token.balanceOf(
 ↳ bobby));
293     console.log("Balance Of Carla        ---> " ,token.balanceOf(
 ↳ carla));
294     console.log("Owner of the NFT        ---> " ,nftContract.
 ↳ ownerOf(1));
295     console.log(" ");
296     console.log("****** LOAN DATA ******");
297     console.log("loanPrincipalAmount        ---> " ,a);
298     console.log("maximumRepaymentAmount     ---> " ,b);
299     console.log("nftCollateralId            ---> " ,c);
300     console.log("loanERC20Denomination      ---> " ,d);
301     console.log("loanDuration               ---> " ,e);
302     console.log("interestRateForDuration    ---> " ,f);
303     console.log("loanAdminFeeInBasisPoints  ---> " ,g);
304     console.log("nftCollateralWrapper       ---> " ,h);
305     console.log("loanStartTime              ---> " ,i);
306     console.log("nftCollateralContract      ---> " ,j);
307     console.log("borrower                   ---> " ,k);
308     console.log("LOAN LIQUIDATED / REPAYED  ---> " ,liquidated);
309     console.log(" ");
310 }
```

MANUAL TESTING

Output

```
[PASS] test_1() (gas: 588236)
Logs:
  ****** SETTING ENVIRONMENT ******
  ****** STATE 1 *******
  ****** BALANCES ******
  Balance Of Admin        --->  99870000000000000000000
  Balance Of Alice        --->  5000000000000000000
  Balance Of Bobby        --->  10000000000000000000000
  Balance Of Carla        --->  25000000000000000000
  Owner of the NFT        --->  0x07748403082b29a45abD6C124A37E6B14e6B1803


  ****************************
  **********  TEST 1  **********
  ****************************

  TX: ALICE ---> ACCEPT BOBBY'S OFFER

  ****** STATE 1 *******
  ****** BALANCES ******
  Balance Of Admin        --->  99870000000000000000000
  Balance Of Alice        --->  6000000000000000000
  Balance Of Bobby        --->  99000000000000000000
  Balance Of Carla        --->  25000000000000000000
  Owner of the NFT        --->  0xB9bc102f443F848dD14Fe4aDfdF987a73e5F594D

  ****** LOAN DATA ******
  loanPrincipalAmount        --->  1000000000000000000
  maximumRepaymentAmount     --->  1200000000000000000
  nftCollateralId            --->  1
  loanERC20Denomination      --->  0x16F2DAC52c23a99c6Ff692b5C9b0bA6f9E46f9cc
  loanDuration               --->  864000
  interestRateForDuration    --->  0
  loanAdminFeeInBasisPoints  --->  500
  nftCollateralWrapper       --->  0xFb5b63865c5AaF1C0C1C6a6053894626883A0782
  loanStartTime              --->  1
  nftCollateralContract      --->  0xec6f254a0C543E30697C04d1B6B6Bd77Fe8b6799
  borrower                   --->  0x07748403082b29a45abD6C124A37E6B14e6B1803
  LOAN LIQUIDATED / REPAYED  --->  false

  5 DAYS LATER...
  TX: ALICE ---> PAY BACK LOAN

  ****** STATE 3 *******
  ****** BALANCES ******
  Balance Of Admin        --->  99870010000000000000000
  Balance Of Alice        --->  4800000000000000000
  Balance Of Bobby        --->  10019000000000000000000
  Balance Of Carla        --->  25000000000000000000
  Owner of the NFT        --->  0x07748403082b29a45abD6C124A37E6B14e6B1803

  ****** LOAN DATA ******
  loanPrincipalAmount        --->  0
  maximumRepaymentAmount     --->  0
  nftCollateralId            --->  0
  loanERC20Denomination      --->  0x0000000000000000000000000000000000000000
  loanDuration               --->  0
  interestRateForDuration    --->  0
  loanAdminFeeInBasisPoints  --->  0
  nftCollateralWrapper       --->  0x0000000000000000000000000000000000000000
  loanStartTime              --->  0
  nftCollateralContract      --->  0x0000000000000000000000000000000000000000
  borrower                   --->  0x0000000000000000000000000000000000000000
  LOAN LIQUIDATED / REPAYED  --->  true
```

TEST 2:

Script

Listing 6: NFTFi.t.sol

```
312 function test_2() public {
313
314     /* ****************************************************** */
315     /* TEST 2: LENDING BORROWING NORMAL PROCEDURE OUT OF TIME */
316     /* ****************************************************** */
317
318     console.log("******************************");
319     console.log("**********  TEST 2  **********");
320     console.log("******************************");
321     console.log(" ");
322
323     // DECLARING OFFER
324     LoanData.Offer memory offer = LoanData.Offer({
325         loanPrincipalAmount: 10_000000000000000000,
326         maximumRepaymentAmount: 12_000000000000000000,
327         nftCollateralId: 1,
328         nftCollateralContract: address(nftContract),
329         loanDuration: 10 days,
330         loanAdminFeeInBasisPoints: 500,
331         loanERC20Denomination: address(token),
332         referrer: zeroo
333     });
334
335     // TOKEN APPROVAL
336     vm.prank(bobby);
337     token.approve(address(directLoanFixedOfferRedeploy), 10
 ↳ _000000000000000000);
338
339     // GETTING CHAIN ID
340     uint256 id;
341     assembly {
342         id := chainid()
343     }
344
345     // PREPARING SIGNATURE STRUCT
346     LoanData.Signature memory signature = LoanData.Signature({
347         nonce: 1,
348         expiry: timeNow + 10 days,
349         signer: bobby,
350         signature: hex"1c"
```

```
351     });
352
353     // GETTING THE MESSAGE HASH
354     bytes32 message = keccak256(
355         abi.encodePacked(getEncodedOffer(offer),
 ↳ getEncodedSignature(signature), address(
 ↳ directLoanFixedOfferRedeploy), id)
356     );
357
358     // EIP712 STANDARD
359     bytes32 signedMessage = ECDSA.toEthSignedMessageHash(message);
360
361     // GETTING THE V, R, S OF THE SIGNED MESSAGE
362     (uint8 v, bytes32 r, bytes32 s) = vm.sign(0xAC, signedMessage)
 ↳ ;
363     bytes memory v_bytes;
364     if(v == 27){v_bytes = hex"1b";} else {v_bytes = hex"1c";}
365     bytes memory signaturesf = bytes.concat(r, s, v_bytes);
366
367     // BOBBY SIGNES
368     LoanData.Signature memory signaturefi = LoanData.Signature({
369         nonce: 1,
370         expiry: timeNow + 10 days,
371         signer: bobby,
372         signature: signaturesf
373     });
374
375     // ALICE ACCEPTS BOBBY'S OFFER
376     LoanData.BorrowerSettings memory borrowerSettings;
377     vm.prank(alice);
378     directLoanFixedOfferRedeploy.acceptOffer(offer, signaturefi,
 ↳ borrowerSettings);
379
380     // CHECKING THAT THE STATE IS AS EXPECTED
381     (uint256 loanPrincipalAmount, uint256 maximumRepaymentAmount,
 ↳ uint256 nftCollateralId, address loanERC20Denomination, uint32
 ↳ loanDuration, uint32 loanInterestRateForDuration, uint16
 ↳ loanAdminFeeInBasisPoints, address nftCollateralWrapper, uint64
 ↳ loanStartTime, address nftCollateralContract, address borrower) =
 ↳ directLoanFixedOfferRedeploy.loanIdToLoan(1);
382     liquidated = directLoanFixedOfferRedeploy.
 ↳ loanRepaidOrLiquidated(1);
383
384     // LOGS
```

```
385     console.log("TX: ALICE ---> ACCEPT BOBBY'S OFFER");
386     console.log(" ");
387     console.log("****** STATE 1 *******");
388     console.log("****** BALANCES ******");
389     console.log("Balance Of Admin        ---> " ,token.balanceOf(
 ↳ admin));
390     console.log("Balance Of Alice        ---> " ,token.balanceOf(
 ↳ alice));
391     console.log("Balance Of Bobby        ---> " ,token.balanceOf(
 ↳ bobby));
392     console.log("Balance Of Carla        ---> " ,token.balanceOf(
 ↳ carla));
393     console.log("Owner of the NFT        ---> " ,nftContract.
 ↳ ownerOf(1));
394     console.log(" ");
395     console.log("****** LOAN DATA ******");
396     console.log("loanPrincipalAmount        ---> " ,
 ↳ loanPrincipalAmount);
397     console.log("maximumRepaymentAmount     ---> " ,
 ↳ maximumRepaymentAmount);
398     console.log("nftCollateralId            ---> " ,
 ↳ nftCollateralId);
399     console.log("loanERC20Denomination      ---> " ,
 ↳ loanERC20Denomination);
400     console.log("loanDuration               ---> " ,loanDuration);
401     console.log("interestRateForDuration    ---> " ,
 ↳ loanInterestRateForDuration);
402     console.log("loanAdminFeeInBasisPoints  ---> " ,
 ↳ loanAdminFeeInBasisPoints);
403     console.log("nftCollateralWrapper       ---> " ,
 ↳ nftCollateralWrapper);
404     console.log("loanStartTime              ---> " ,loanStartTime)
 ↳ ;
405     console.log("nftCollateralContract      ---> " ,
 ↳ nftCollateralContract);
406     console.log("borrower                   ---> " ,borrower);
407     console.log("LOAN LIQUIDATED / REPAYED  ---> " ,liquidated);
408     console.log(" ");
409
410     // 15 DAYS LATER
411     console.log("15 DAYS LATER...");
412     vm.warp(15 * day);
413
414     // ALICE PAY THE MONEY
```

```
415     vm.prank(alice);
416     token.approve(address(directLoanFixedOfferRedeploy), 12
 ↳ _000000000000000000);
417
418     vm.prank(alice);
419     vm.expectRevert();
420     directLoanFixedOfferRedeploy.payBackLoan(1);
421
422
423     // CHECKING THAT THE STATE IS AS EXPECTED
424     (uint256 a, uint256 b, uint256 c, address d, uint32 e, uint32
 ↳ f, uint16 g, address h, uint64 i, address j, address k) =
 ↳ directLoanFixedOfferRedeploy.loanIdToLoan(1);
425     liquidated = directLoanFixedOfferRedeploy.
 ↳ loanRepaidOrLiquidated(1);
426     console.log("TX: ALICE ---> PAY BACK LOAN");
427     console.log(" ");
428     console.log("****** STATE 3 *******");
429     console.log("****** BALANCES ******");
430     console.log("Balance Of Admin        ---> " ,token.balanceOf(
 ↳ admin));
431     console.log("Balance Of Alice        ---> " ,token.balanceOf(
 ↳ alice));
432     console.log("Balance Of Bobby        ---> " ,token.balanceOf(
 ↳ bobby));
433     console.log("Balance Of Carla        ---> " ,token.balanceOf(
 ↳ carla));
434     console.log("Owner of the NFT        ---> " ,nftContract.
 ↳ ownerOf(1));
435     console.log(" ");
436     console.log("****** LOAN DATA ******");
437     console.log("loanPrincipalAmount        ---> " ,a);
438     console.log("maximumRepaymentAmount     ---> " ,b);
439     console.log("nftCollateralId            ---> " ,c);
440     console.log("loanERC20Denomination      ---> " ,d);
441     console.log("loanDuration               ---> " ,e);
442     console.log("interestRateForDuration    ---> " ,f);
443     console.log("loanAdminFeeInBasisPoints  ---> " ,g);
444     console.log("nftCollateralWrapper       ---> " ,h);
445     console.log("loanStartTime              ---> " ,i);
446     console.log("nftCollateralContract      ---> " ,j);
447     console.log("borrower                   ---> " ,k);
448     console.log("LOAN LIQUIDATED / REPAYED  ---> " ,liquidated);
449     console.log(" ");
```

```
450 }
```

Output

```
[PASS] test_2() (gas: 616101)
Logs:
  ****** SETTING ENVIRONMENT ******
  ****** STATE 1 *******
  ****** BALANCES ******
  Balance Of Admin        --->  99870000000000000000000
  Balance Of Alice        --->  5000000000000000000
  Balance Of Bobby        --->  10000000000000000000000
  Balance Of Carla        --->  25000000000000000000000
  Owner of the NFT        --->  0x07748403082b29a45abD6C124A37E6B14e6B1803


  ******************************
  **********  TEST 2  **********
  ******************************

  TX: ALICE ---> ACCEPT BOBBY'S OFFER

  ****** STATE 1 *******
  ****** BALANCES ******
  Balance Of Admin        --->  99870000000000000000000
  Balance Of Alice        --->  6000000000000000000
  Balance Of Bobby        --->  99000000000000000000000
  Balance Of Carla        --->  25000000000000000000000
  Owner of the NFT        --->  0xB9bc102f443F848dD14Fe4aDfdF987a73e5F594D

  ****** LOAN DATA ******
  loanPrincipalAmount        --->  1000000000000000000
  maximumRepaymentAmount     --->  1200000000000000000
  nftCollateralId            --->  1
  loanERC20Denomination      --->  0x16F2DAC52c23a99c6Ff692b5C9b0bA6f9E46f9cc
  loanDuration               --->  864000
  interestRateForDuration    --->  0
  loanAdminFeeInBasisPoints  --->  500
  nftCollateralWrapper       --->  0xFb5b63865c5AaF1C0C1C6a6053894626883A0782
  loanStartTime              --->  1
  nftCollateralContract      --->  0xec6f254a0C543E30697C04d1B6B6Bd77Fe8b6799
  borrower                   --->  0x07748403082b29a45abD6C124A37E6B14e6B1803
  LOAN LIQUIDATED / REPAYED  --->  false

  15 DAYS LATER...
  TX: ALICE ---> PAY BACK LOAN

  ****** STATE 3 *******
  ****** BALANCES ******
  Balance Of Admin        --->  99870000000000000000000
  Balance Of Alice        --->  6000000000000000000
  Balance Of Bobby        --->  99000000000000000000000
  Balance Of Carla        --->  25000000000000000000000
  Owner of the NFT        --->  0xB9bc102f443F848dD14Fe4aDfdF987a73e5F594D

  ****** LOAN DATA ******
  loanPrincipalAmount        --->  1000000000000000000
  maximumRepaymentAmount     --->  1200000000000000000
  nftCollateralId            --->  1
  loanERC20Denomination      --->  0x16F2DAC52c23a99c6Ff692b5C9b0bA6f9E46f9cc
  loanDuration               --->  864000
  interestRateForDuration    --->  0
  loanAdminFeeInBasisPoints  --->  500
  nftCollateralWrapper       --->  0xFb5b63865c5AaF1C0C1C6a6053894626883A0782
  loanStartTime              --->  1
  nftCollateralContract      --->  0xec6f254a0C543E30697C04d1B6B6Bd77Fe8b6799
  borrower                   --->  0x07748403082b29a45abD6C124A37E6B14e6B1803
  LOAN LIQUIDATED / REPAYED  --->  false
```

TEST 3:

Script

Listing 7: NFTFi.t.sol

```
174 function test_1() public {
175
176     /* ****************************************** */
177     /* TEST 1: LENDING BORROWING NORMAL PROCEDURE */
178     /* ****************************************** */
179
180     console.log("******************************");
181     console.log("**********  TEST 1  **********");
182     console.log("******************************");
183     console.log(" ");
184
185     // DECLARING OFFER
186     LoanData.Offer memory offer = LoanData.Offer({
187         loanPrincipalAmount: 10_000000000000000000,
188         maximumRepaymentAmount: 12_000000000000000000,
189         nftCollateralId: 1,
190         nftCollateralContract: address(nftContract),
191         loanDuration: 10 days,
192         loanAdminFeeInBasisPoints: 500,
193         loanERC20Denomination: address(token),
194         referrer: zeroo
195     });
196
197     // TOKEN APPROVAL
198     vm.prank(bobby);
199     token.approve(address(directLoanFixedOfferRedeploy), 10
 ↳ _000000000000000000);
200
201     // GETTING CHAIN ID
202     uint256 id;
203     assembly {
204         id := chainid()
205     }
206
207     // PREPARING SIGNATURE STRUCT
208     LoanData.Signature memory signature = LoanData.Signature({
209         nonce: 1,
210         expiry: timeNow + 10 days,
211         signer: bobby,
212         signature: hex"1c"
```

```
213     });
214
215     // GETTING THE MESSAGE HASH
216     bytes32 message = keccak256(
217         abi.encodePacked(getEncodedOffer(offer),
 ↳ getEncodedSignature(signature), address(
 ↳ directLoanFixedOfferRedeploy), id)
218     );
219
220     // EIP712 STANDARD
221     bytes32 signedMessage = ECDSA.toEthSignedMessageHash(message);
222
223     // GETTING THE V, R, S OF THE SIGNED MESSAGE
224     (uint8 v, bytes32 r, bytes32 s) = vm.sign(0xAC, signedMessage)
 ↳ ;
225     bytes memory v_bytes;
226     if(v == 27){v_bytes = hex"1b";} else {v_bytes = hex"1c";}
227     bytes memory signaturesf = bytes.concat(r, s, v_bytes);
228
229     // BOBBY SIGNES
230     LoanData.Signature memory signaturefi = LoanData.Signature({
231         nonce: 1,
232         expiry: timeNow + 10 days,
233         signer: bobby,
234         signature: signaturesf
235     });
236
237     // ALICE ACCEPTS BOBBY'S OFFER
238     LoanData.BorrowerSettings memory borrowerSettings;
239     vm.prank(alice);
240     directLoanFixedOfferRedeploy.acceptOffer(offer, signaturefi,
 ↳ borrowerSettings);
241
242     // CHECKING THAT THE STATE IS AS EXPECTED
243     (uint256 loanPrincipalAmount, uint256 maximumRepaymentAmount,
 ↳ uint256 nftCollateralId, address loanERC20Denomination, uint32
 ↳ loanDuration, uint32 loanInterestRateForDuration, uint16
 ↳ loanAdminFeeInBasisPoints, address nftCollateralWrapper, uint64
 ↳ loanStartTime, address nftCollateralContract, address borrower) =
 ↳ directLoanFixedOfferRedeploy.loanIdToLoan(1);
244     liquidated = directLoanFixedOfferRedeploy.
 ↳ loanRepaidOrLiquidated(1);
245
246     // LOGS
```

```
247     console.log("TX: ALICE ---> ACCEPT BOBBY'S OFFER");
248     console.log(" ");
249     console.log("****** STATE 1 *******");
250     console.log("****** BALANCES ******");
251     console.log("Balance Of Admin        ---> " ,token.balanceOf(
 ↳ admin));
252     console.log("Balance Of Alice        ---> " ,token.balanceOf(
 ↳ alice));
253     console.log("Balance Of Bobby        ---> " ,token.balanceOf(
 ↳ bobby));
254     console.log("Balance Of Carla        ---> " ,token.balanceOf(
 ↳ carla));
255     console.log("Owner of the NFT        ---> " ,nftContract.
 ↳ ownerOf(1));
256     console.log(" ");
257     console.log("****** LOAN DATA ******");
258     console.log("loanPrincipalAmount        ---> " ,
 ↳ loanPrincipalAmount);
259     console.log("maximumRepaymentAmount     ---> " ,
 ↳ maximumRepaymentAmount);
260     console.log("nftCollateralId            ---> " ,
 ↳ nftCollateralId);
261     console.log("loanERC20Denomination      ---> " ,
 ↳ loanERC20Denomination);
262     console.log("loanDuration               ---> " ,loanDuration);
263     console.log("interestRateForDuration    ---> " ,
 ↳ loanInterestRateForDuration);
264     console.log("loanAdminFeeInBasisPoints  ---> " ,
 ↳ loanAdminFeeInBasisPoints);
265     console.log("nftCollateralWrapper       ---> " ,
 ↳ nftCollateralWrapper);
266     console.log("loanStartTime              ---> " ,loanStartTime)
 ↳ ;
267     console.log("nftCollateralContract      ---> " ,
 ↳ nftCollateralContract);
268     console.log("borrower                   ---> " ,borrower);
269     console.log("LOAN LIQUIDATED / REPAYED  ---> " ,liquidated);
270     console.log(" ");
271
272     // 5 DAYS LATER
273     console.log("5 DAYS LATER...");
274     vm.warp(5 * day);
275
276     // ALICE PAY THE MONEY
```

```
277     vm.prank(alice);
278     token.approve(address(directLoanFixedOfferRedeploy), 12
 ↳ _000000000000000000);
279     vm.prank(alice);
280     directLoanFixedOfferRedeploy.payBackLoan(1);
281
282
283     // CHECKING THAT THE STATE IS AS EXPECTED
284     (uint256 a, uint256 b, uint256 c, address d, uint32 e, uint32
 ↳ f, uint16 g, address h, uint64 i, address j, address k) =
 ↳ directLoanFixedOfferRedeploy.loanIdToLoan(1);
285     liquidated = directLoanFixedOfferRedeploy.
 ↳ loanRepaidOrLiquidated(1);
286     console.log("TX: ALICE ---> PAY BACK LOAN");
287     console.log(" ");
288     console.log("****** STATE 3 *******");
289     console.log("****** BALANCES ******");
290     console.log("Balance Of Admin        ---> " ,token.balanceOf(
 ↳ admin));
291     console.log("Balance Of Alice        ---> " ,token.balanceOf(
 ↳ alice));
292     console.log("Balance Of Bobby        ---> " ,token.balanceOf(
 ↳ bobby));
293     console.log("Balance Of Carla        ---> " ,token.balanceOf(
 ↳ carla));
294     console.log("Owner of the NFT        ---> " ,nftContract.
 ↳ ownerOf(1));
295     console.log(" ");
296     console.log("****** LOAN DATA ******");
297     console.log("loanPrincipalAmount        ---> " ,a);
298     console.log("maximumRepaymentAmount     ---> " ,b);
299     console.log("nftCollateralId            ---> " ,c);
300     console.log("loanERC20Denomination      ---> " ,d);
301     console.log("loanDuration               ---> " ,e);
302     console.log("interestRateForDuration    ---> " ,f);
303     console.log("loanAdminFeeInBasisPoints  ---> " ,g);
304     console.log("nftCollateralWrapper       ---> " ,h);
305     console.log("loanStartTime              ---> " ,i);
306     console.log("nftCollateralContract      ---> " ,j);
307     console.log("borrower                   ---> " ,k);
308     console.log("LOAN LIQUIDATED / REPAYED  ---> " ,liquidated);
309     console.log(" ");
310 }
```

MANUAL TESTING

Output

```
[PASS] test_3() (gas: 564732)
Logs:
  ****** SETTING ENVIRONMENT ******
  ****** STATE 1 *******
  ****** BALANCES ******
  Balance Of Admin        --->  99870000000000000000000
  Balance Of Alice        --->  5000000000000000000
  Balance Of Bobby        --->  10000000000000000000000
  Balance Of Carla        --->  25000000000000000000
  Owner of the NFT        --->  0x07748403082b29a45abD6C124A37E6B14e6B1803


  ****************************
  **********  TEST 3  **********
  ****************************

  TX: ALICE ---> ACCEPT BOBBY'S OFFER

  ****** STATE 1 *******
  ****** BALANCES ******
  Balance Of Admin        --->  99870000000000000000000
  Balance Of Alice        --->  6000000000000000000
  Balance Of Bobby        --->  99000000000000000000
  Balance Of Carla        --->  25000000000000000000
  Owner of the NFT        --->  0xB9bc102f443F848dD14Fe4aDfdF987a73e5F594D

  ****** LOAN DATA ******
  loanPrincipalAmount        --->  1000000000000000000
  maximumRepaymentAmount     --->  1200000000000000000
  nftCollateralId            --->  1
  loanERC20Denomination      --->  0x16F2DAC52c23a99c6Ff692b5C9b0bA6f9E46f9cc
  loanDuration               --->  864000
  interestRateForDuration    --->  0
  loanAdminFeeInBasisPoints  --->  500
  nftCollateralWrapper       --->  0xFb5b63865c5AaF1C0C1C6a6053894626883A0782
  loanStartTime              --->  1
  nftCollateralContract      --->  0xec6f254a0C543E30697C04d1B6B6Bd77Fe8b6799
  borrower                   --->  0x07748403082b29a45abD6C124A37E6B14e6B1803
  LOAN LIQUIDATED / REPAYED  --->  false

  11 DAYS LATER...
  TX: BOBBY ---> LIQUIDATE LOAN

  ****** STATE 3 *******
  ****** BALANCES ******
  Balance Of Admin        --->  99870000000000000000000
  Balance Of Alice        --->  6000000000000000000
  Balance Of Bobby        --->  99000000000000000000
  Balance Of Carla        --->  25000000000000000000
  Owner of the NFT        --->  0x63486b70d804464766Cfd096bba5552c4BcdAC30

  ****** LOAN DATA ******
  loanPrincipalAmount        --->  0
  maximumRepaymentAmount     --->  0
  nftCollateralId            --->  0
  loanERC20Denomination      --->  0x0000000000000000000000000000000000000000
  loanDuration               --->  0
  interestRateForDuration    --->  0
  loanAdminFeeInBasisPoints  --->  0
  nftCollateralWrapper       --->  0x0000000000000000000000000000000000000000
  loanStartTime              --->  0
  nftCollateralContract      --->  0x0000000000000000000000000000000000000000
  borrower                   --->  0x0000000000000000000000000000000000000000
  LOAN LIQUIDATED / REPAYED  --->  true
```

TEST 4:

Script

Listing 8: NFTFi.t.sol

```
589     function test_4() public {
590
591     /* ************************************************** */
592     /* TEST 4: LENDING + LIQUIDATION BEFORE THAN LOAN FINISH */
593     /* ************************************************** */
594
595     console.log("******************************");
596     console.log("**********  TEST 4  **********");
597     console.log("******************************");
598     console.log(" ");
599
600     // DECLARING OFFER
601     LoanData.Offer memory offer = LoanData.Offer({
602         loanPrincipalAmount: 10_000000000000000000,
603         maximumRepaymentAmount: 12_000000000000000000,
604         nftCollateralId: 1,
605         nftCollateralContract: address(nftContract),
606         loanDuration: 10 days,
607         loanAdminFeeInBasisPoints: 500,
608         loanERC20Denomination: address(token),
609         referrer: zeroo
610     });
611
612     // TOKEN APPROVAL
613     vm.prank(bobby);
614     token.approve(address(directLoanFixedOfferRedeploy), 10
 ↳ _000000000000000000);
615
616     // GETTING CHAIN ID
617     uint256 id;
618     assembly {
619         id := chainid()
620     }
621
622     // PREPARING SIGNATURE STRUCT
623     LoanData.Signature memory signature = LoanData.Signature({
624         nonce: 1,
625         expiry: timeNow + 10 days,
626         signer: bobby,
627         signature: hex"1c"
```

```
628     });
629
630     // GETTING THE MESSAGE HASH
631     bytes32 message = keccak256(
632         abi.encodePacked(getEncodedOffer(offer),
 ↳ getEncodedSignature(signature), address(
 ↳ directLoanFixedOfferRedeploy), id)
633     );
634
635     // EIP712 STANDARD
636     bytes32 signedMessage = ECDSA.toEthSignedMessageHash(message);
637
638     // GETTING THE V, R, S OF THE SIGNED MESSAGE
639     (uint8 v, bytes32 r, bytes32 s) = vm.sign(0xAC, signedMessage)
 ↳ ;
640     bytes memory v_bytes;
641     if(v == 27){v_bytes = hex"1b";} else {v_bytes = hex"1c";}
642     bytes memory signaturesf = bytes.concat(r, s, v_bytes);
643
644     // BOBBY SIGNES
645     LoanData.Signature memory signaturefi = LoanData.Signature({
646         nonce: 1,
647         expiry: timeNow + 10 days,
648         signer: bobby,
649         signature: signaturesf
650     });
651
652     // ALICE ACCEPTS BOBBY'S OFFER
653     LoanData.BorrowerSettings memory borrowerSettings;
654     vm.prank(alice);
655     directLoanFixedOfferRedeploy.acceptOffer(offer, signaturefi,
 ↳ borrowerSettings);
656
657     // CHECKING THAT THE STATE IS AS EXPECTED
658     (uint256 loanPrincipalAmount, uint256 maximumRepaymentAmount,
 ↳ uint256 nftCollateralId, address loanERC20Denomination, uint32
 ↳ loanDuration, uint32 loanInterestRateForDuration, uint16
 ↳ loanAdminFeeInBasisPoints, address nftCollateralWrapper, uint64
 ↳ loanStartTime, address nftCollateralContract, address borrower) =
 ↳ directLoanFixedOfferRedeploy.loanIdToLoan(1);
659     liquidated = directLoanFixedOfferRedeploy.
 ↳ loanRepaidOrLiquidated(1);
660
661     // LOGS
```

```
662     console.log("TX: ALICE ---> ACCEPT BOBBY'S OFFER");
663     console.log(" ");
664     console.log("****** STATE 1 *******");
665     console.log("****** BALANCES ******");
666     console.log("Balance Of Admin        ---> " ,token.balanceOf(
 ↳ admin));
667     console.log("Balance Of Alice        ---> " ,token.balanceOf(
 ↳ alice));
668     console.log("Balance Of Bobby        ---> " ,token.balanceOf(
 ↳ bobby));
669     console.log("Balance Of Carla        ---> " ,token.balanceOf(
 ↳ carla));
670     console.log("Owner of the NFT        ---> " ,nftContract.
 ↳ ownerOf(1));
671     console.log(" ");
672     console.log("****** LOAN DATA ******");
673     console.log("loanPrincipalAmount        ---> " ,
 ↳ loanPrincipalAmount);
674     console.log("maximumRepaymentAmount     ---> " ,
 ↳ maximumRepaymentAmount);
675     console.log("nftCollateralId            ---> " ,
 ↳ nftCollateralId);
676     console.log("loanERC20Denomination      ---> " ,
 ↳ loanERC20Denomination);
677     console.log("loanDuration               ---> " ,loanDuration);
678     console.log("interestRateForDuration    ---> " ,
 ↳ loanInterestRateForDuration);
679     console.log("loanAdminFeeInBasisPoints  ---> " ,
 ↳ loanAdminFeeInBasisPoints);
680     console.log("nftCollateralWrapper       ---> " ,
 ↳ nftCollateralWrapper);
681     console.log("loanStartTime              ---> " ,loanStartTime)
 ↳ ;
682     console.log("nftCollateralContract      ---> " ,
 ↳ nftCollateralContract);
683     console.log("borrower                   ---> " ,borrower);
684     console.log("LOAN LIQUIDATED / REPAYED  ---> " ,liquidated);
685     console.log(" ");
686
687     // 3 DAYS LATER
688     vm.warp(3 * day);
689     console.log("3 DAYS LATER...");
690
691     // BOBBY LIQUIDATES THE LOAN
```

```
692     vm.prank(bobby);
693     vm.expectRevert();
694     directLoanFixedOfferRedeploy.liquidateOverdueLoan(1);
695
696
697     // CHECKING THAT THE STATE IS AS EXPECTED
698     (uint256 a, uint256 b, uint256 c, address d, uint32 e, uint32
 ↳ f, uint16 g, address h, uint64 i, address j, address k) =
 ↳ directLoanFixedOfferRedeploy.loanIdToLoan(1);
699     liquidated = directLoanFixedOfferRedeploy.
 ↳ loanRepaidOrLiquidated(1);
700     console.log("TX: BOBBY ---> LIQUIDATE LOAN");
701     console.log(" ");
702     console.log("****** STATE 3 *******");
703     console.log("****** BALANCES ******");
704     console.log("Balance Of Admin        ---> " ,token.balanceOf(
 ↳ admin));
705     console.log("Balance Of Alice        ---> " ,token.balanceOf(
 ↳ alice));
706     console.log("Balance Of Bobby        ---> " ,token.balanceOf(
 ↳ bobby));
707     console.log("Balance Of Carla        ---> " ,token.balanceOf(
 ↳ carla));
708     console.log("Owner of the NFT        ---> " ,nftContract.
 ↳ ownerOf(1));
709     console.log(" ");
710     console.log("****** LOAN DATA ******");
711     console.log("loanPrincipalAmount        ---> " ,a);
712     console.log("maximumRepaymentAmount     ---> " ,b);
713     console.log("nftCollateralId            ---> " ,c);
714     console.log("loanERC20Denomination      ---> " ,d);
715     console.log("loanDuration               ---> " ,e);
716     console.log("interestRateForDuration    ---> " ,f);
717     console.log("loanAdminFeeInBasisPoints  ---> " ,g);
718     console.log("nftCollateralWrapper       ---> " ,h);
719     console.log("loanStartTime              ---> " ,i);
720     console.log("nftCollateralContract      ---> " ,j);
721     console.log("borrower                   ---> " ,k);
722     console.log("LOAN LIQUIDATED / REPAYED  ---> " ,liquidated);
723     console.log(" ");
724 }
725
726 function firstStep_test5() internal {
727
```

```
728     // DECLARING OFFER
729     LoanData.Offer memory offer = LoanData.Offer({
730         loanPrincipalAmount: 10_000000000000000000,
731         maximumRepaymentAmount: 12_000000000000000000,
732         nftCollateralId: 1,
733         nftCollateralContract: address(nftContract),
734         loanDuration: 10 days,
735         loanAdminFeeInBasisPoints: 500,
736         loanERC20Denomination: address(token),
737         referrer: zeroo
738     });
739 
740     // TOKEN APPROVAL
741     vm.prank(bobby);
742     token.approve(address(directLoanFixedOfferRedeploy), 10
 ↳ _000000000000000000);
743 
744     // GETTING CHAIN ID
745     uint256 id;
746     assembly {
747         id := chainid()
748     }
749 
750     // PREPARING SIGNATURE STRUCT
751     LoanData.Signature memory signature = LoanData.Signature({
752         nonce: 1,
753         expiry: timeNow + 10 days,
754         signer: bobby,
755         signature: hex"1c"
756     });
757 
758     // GETTING THE MESSAGE HASH
759     bytes32 message = keccak256(
760         abi.encodePacked(getEncodedOffer(offer),
 ↳ getEncodedSignature(signature), address(
 ↳ directLoanFixedOfferRedeploy), id)
761     );
762 
763     // EIP712 STANDARD
764     bytes32 signedMessage = ECDSA.toEthSignedMessageHash(message);
765 
766     // GETTING THE V, R, S OF THE SIGNED MESSAGE
767     (uint8 v, bytes32 r, bytes32 s) = vm.sign(0xAC, signedMessage)
 ↳ ;
```

```
768     bytes memory v_bytes;
769     if(v == 27){v_bytes = hex"1b";} else {v_bytes = hex"1c";}
770     bytes memory signaturesf = bytes.concat(r, s, v_bytes);
771 
772     // BOBBY SIGNES
773     LoanData.Signature memory signaturefi = LoanData.Signature({
774         nonce: 1,
775         expiry: timeNow + 10 days,
776         signer: bobby,
777         signature: signaturesf
778     });
779 
780     // ALICE ACCEPTS BOBBY'S OFFER
781     LoanData.BorrowerSettings memory borrowerSettings;
782     vm.prank(alice);
783     directLoanFixedOfferRedeploy.acceptOffer(offer, signaturefi,
 ↳ borrowerSettings);
784 
785     // CHECKING THAT THE STATE IS AS EXPECTED
786     (uint256 loanPrincipalAmount, uint256 maximumRepaymentAmount,
 ↳ uint256 nftCollateralId, address loanERC20Denomination, uint32
 ↳ loanDuration, uint32 loanInterestRateForDuration, uint16
 ↳ loanAdminFeeInBasisPoints, address nftCollateralWrapper, uint64
 ↳ loanStartTime, address nftCollateralContract, address borrower) =
 ↳ directLoanFixedOfferRedeploy.loanIdToLoan(1);
787     liquidated = directLoanFixedOfferRedeploy.
 ↳ loanRepaidOrLiquidated(1);
788 
789     // LOGS
790     console.log("TX: ALICE ---> ACCEPT BOBBY'S OFFER");
791     console.log(" ");
792     console.log("****** STATE 1 *******");
793     console.log("****** BALANCES ******");
794     console.log("Balance Of Admin        ---> " ,token.balanceOf(
 ↳ admin));
795     console.log("Balance Of Alice        ---> " ,token.balanceOf(
 ↳ alice));
796     console.log("Balance Of Bobby        ---> " ,token.balanceOf(
 ↳ bobby));
797     console.log("Balance Of Carla        ---> " ,token.balanceOf(
 ↳ carla));
798     console.log("Owner of the NFT        ---> " ,nftContract.
 ↳ ownerOf(1));
799     console.log(" ");
```

```
800     console.log("****** LOAN DATA ******");
801     console.log("loanPrincipalAmount        ---> " ,
 ↳ loanPrincipalAmount);
802     console.log("maximumRepaymentAmount     ---> " ,
 ↳ maximumRepaymentAmount);
803     console.log("nftCollateralId            ---> " ,
 ↳ nftCollateralId);
804     console.log("loanERC20Denomination      ---> " ,
 ↳ loanERC20Denomination);
805     console.log("loanDuration               ---> " ,loanDuration);
806     console.log("interestRateForDuration    ---> " ,
 ↳ loanInterestRateForDuration);
807     console.log("loanAdminFeeInBasisPoints  ---> " ,
 ↳ loanAdminFeeInBasisPoints);
808     console.log("nftCollateralWrapper       ---> " ,
 ↳ nftCollateralWrapper);
809     console.log("loanStartTime              ---> " ,loanStartTime)
 ↳ ;
810     console.log("nftCollateralContract      ---> " ,
 ↳ nftCollateralContract);
811     console.log("borrower                   ---> " ,borrower);
812     console.log("LOAN LIQUIDATED / REPAYED  ---> " ,liquidated);
813     console.log(" ");
814 }
```

Output

```
[PASS] test_4() (gas: 593005)
Logs:
  ****** SETTING ENVIRONMENT ******
  ****** STATE 1 *******
  ****** BALANCES ******
  Balance Of Admin        --->  99870000000000000000000
  Balance Of Alice        --->  5000000000000000000
  Balance Of Bobby        --->  10000000000000000000000
  Balance Of Carla        --->  25000000000000000000000
  Owner of the NFT        --->  0x07748403082b29a45abD6C124A37E6B14e6B1803


  ****************************
  **********  TEST 4  **********
  ****************************

  TX: ALICE ---> ACCEPT BOBBY'S OFFER

  ****** STATE 1 *******
  ****** BALANCES ******
  Balance Of Admin        --->  99870000000000000000000
  Balance Of Alice        --->  6000000000000000000
  Balance Of Bobby        --->  99000000000000000000000
  Balance Of Carla        --->  25000000000000000000000
  Owner of the NFT        --->  0xB9bc102f443F848dD14Fe4aDfdF987a73e5F594D

  ****** LOAN DATA ******
  loanPrincipalAmount        --->  1000000000000000000
  maximumRepaymentAmount     --->  1200000000000000000
  nftCollateralId            --->  1
  loanERC20Denomination      --->  0x16F2DAC52c23a99c6Ff692b5C9b0bA6f9E46f9cc
  loanDuration               --->  864000
  interestRateForDuration    --->  0
  loanAdminFeeInBasisPoints  --->  500
  nftCollateralWrapper       --->  0xFb5b63865c5AaF1C0C1C6a60538946268 83A0782
  loanStartTime              --->  1
  nftCollateralContract      --->  0xec6f254a0C543E30697C04d1B6B6Bd77Fe8b6799
  borrower                   --->  0x07748403082b29a45abD6C124A37E6B14e6B1803
  LOAN LIQUIDATED / REPAYED  --->  false

  3 DAYS LATER...
  TX: BOBBY ---> LIQUIDATE LOAN

  ****** STATE 3 *******
  ****** BALANCES ******
  Balance Of Admin        --->  99870000000000000000000
  Balance Of Alice        --->  6000000000000000000
  Balance Of Bobby        --->  99000000000000000000000
  Balance Of Carla        --->  25000000000000000000000
  Owner of the NFT        --->  0xB9bc102f443F848dD14Fe4aDfdF987a73e5F594D

  ****** LOAN DATA ******
  loanPrincipalAmount        --->  1000000000000000000
  maximumRepaymentAmount     --->  1200000000000000000
  nftCollateralId            --->  1
  loanERC20Denomination      --->  0x16F2DAC52c23a99c6Ff692b5C9b0bA6f9E46f9cc
  loanDuration               --->  864000
  interestRateForDuration    --->  0
  loanAdminFeeInBasisPoints  --->  500
  nftCollateralWrapper       --->  0xFb5b63865c5AaF1C0C1C6a60538946268 83A0782
  loanStartTime              --->  1
  nftCollateralContract      --->  0xec6f254a0C543E30697C04d1B6B6Bd77Fe8b6799
  borrower                   --->  0x07748403082b29a45abD6C124A37E6B14e6B1803
  LOAN LIQUIDATED / REPAYED  --->  false
```

TEST 5:

Script

Listing 9: NFTFi.t.sol

```
816 function test_5() public {
817     /* ****************************************************** */
818     /* TEST 5: BORROWING + WANT TO PAY LATE + RENEGOTIATE LOAN */
819     /* ****************************************************** */
820
821     console.log("******************************");
822     console.log("**********  TEST 5  **********");
823     console.log("******************************");
824     console.log(" ");
825
826     firstStep_test5();
827
828     // 15 DAYS LATER
829     console.log("12 DAYS LATER...");
830     vm.warp(12 days);
831
832     uint256 id2;
833     assembly {
834         id2 := chainid()
835     }
836
837     uint256 _expiry = block.timestamp + 20 days;
838
839     // GETTING THE MESSAGE HASH
840     bytes32 message = keccak256(
841             abi.encodePacked(
842                 uint256(1),
843                 uint32(30 days),
844                 uint256(20_000000000000000000),
845                 uint256(5_000000000000000000),
846                 abi.encodePacked(bobby, uint256(2), _expiry),
847                 address(directLoanFixedOfferRedeploy),
848                 id2
849             )
850         );
851
852     // EIP712 STANDARD
853     bytes32 signedMessage = ECDSA.toEthSignedMessageHash(message);
854
855     // GETTING THE V, R, S OF THE SIGNED MESSAGE
```

```
856     (uint8 v, bytes32 r, bytes32 s) = vm.sign(0xAC, signedMessage)
 ↳ ;
857     bytes memory v_bytes;
858     if(v == 27){v_bytes = hex"1b";} else {v_bytes = hex"1c";}
859     bytes memory _lenderSignature = bytes.concat(r, s, v_bytes);
860
861     // ALICE WANTS TO RENEGOTIATE
862     vm.prank(alice);
863     token.approve(address(directLoanFixedOfferRedeploy), 5
 ↳ _000000000000000000);
864     vm.prank(alice);
865     directLoanFixedOfferRedeploy.renegotiateLoan(1, 30 days, 20
 ↳ _000000000000000000, 5_000000000000000000, 2, _expiry,
 ↳ _lenderSignature);
866
867     // 10 DAYS LATER
868     console.log("10 DAYS LATER...");
869     vm.warp(10 * day);
870
871     // ALICE PAY THE MONEY
872     vm.prank(alice);
873     token.approve(address(directLoanFixedOfferRedeploy), 20
 ↳ _000000000000000000);
874     vm.prank(alice);
875     token.approve(address(directLoanFixedOfferRedeploy), 20
 ↳ _000000000000000000);
876     vm.prank(alice);
877     directLoanFixedOfferRedeploy.payBackLoan(1);
878
879     // CHECKING THAT THE STATE IS AS EXPECTED
880     (uint256 a, uint256 b, uint256 c, address d, uint32 e, uint32
 ↳ f, uint16 g, address h, uint64 i, address j, address k) =
 ↳ directLoanFixedOfferRedeploy.loanIdToLoan(1);
881     liquidated = directLoanFixedOfferRedeploy.
 ↳ loanRepaidOrLiquidated(1);
882     console.log("TX: ALICE ---> PAY BACK LOAN");
883     console.log(" ");
884     console.log("****** STATE 3 *******");
885     console.log("****** BALANCES ******");
886     console.log("Balance Of Admin        ---> " ,token.balanceOf(
 ↳ admin));
887     console.log("Balance Of Alice        ---> " ,token.balanceOf(
 ↳ alice));
```

```
888     console.log("Balance Of Bobby        ---> " ,token.balanceOf(
 ↳ bobby));
889     console.log("Balance Of Carla        ---> " ,token.balanceOf(
 ↳ carla));
890     console.log("Owner of the NFT        ---> " ,nftContract.
 ↳ ownerOf(1));
891     console.log(" ");
892     console.log("****** LOAN DATA ******");
893     console.log("loanPrincipalAmount        ---> " ,a);
894     console.log("maximumRepaymentAmount     ---> " ,b);
895     console.log("nftCollateralId            ---> " ,c);
896     console.log("loanERC20Denomination      ---> " ,d);
897     console.log("loanDuration               ---> " ,e);
898     console.log("interestRateForDuration    ---> " ,f);
899     console.log("loanAdminFeeInBasisPoints  ---> " ,g);
900     console.log("nftCollateralWrapper       ---> " ,h);
901     console.log("loanStartTime              ---> " ,i);
902     console.log("nftCollateralContract      ---> " ,j);
903     console.log("borrower                   ---> " ,k);
904     console.log("LOAN LIQUIDATED / REPAYED  ---> " ,liquidated);
905     console.log(" ");
906 }
```

MANUAL TESTING

Output

```
[PASS] test_5() (gas: 671720)
Logs:
  ****** SETTING ENVIRONMENT ******
  ****** STATE 1 *******
  ****** BALANCES ******
  Balance Of Admin        --->  99870000000000000000000
  Balance Of Alice        --->  5000000000000000000
  Balance Of Bobby        --->  10000000000000000000000
  Balance Of Carla        --->  25000000000000000000
  Owner of the NFT        --->  0x07748403082b29a45abD6C124A37E6B14e6B1803


  ******************************
  **********  TEST 5  **********
  ******************************

  TX: ALICE ---> ACCEPT BOBBY'S OFFER

  ****** STATE 1 *******
  ****** BALANCES ******
  Balance Of Admin        --->  99870000000000000000000
  Balance Of Alice        --->  6000000000000000000
  Balance Of Bobby        --->  99000000000000000000
  Balance Of Carla        --->  25000000000000000000
  Owner of the NFT        --->  0xB9bc102f443F848dD14Fe4aDfdF987a73e5F594D

  ****** LOAN DATA ******
  loanPrincipalAmount        --->  1000000000000000000
  maximumRepaymentAmount     --->  1200000000000000000
  nftCollateralId            --->  1
  loanERC20Denomination      --->  0x16F2DAC52c23a99c6Ff692b5C9b0bA6f9E46f9cc
  loanDuration               --->  864000
  interestRateForDuration    --->  0
  loanAdminFeeInBasisPoints  --->  500
  nftCollateralWrapper       --->  0xFb5b63865c5AaF1C0C1C6a60538946268 83A0782
  loanStartTime              --->  1
  nftCollateralContract      --->  0xec6f254a0C543E30697C04d1B6B6Bd77Fe8b6799
  borrower                   --->  0x07748403082b29a45abD6C124A37E6B14e6B1803
  LOAN LIQUIDATED / REPAYED  --->  false

  12 DAYS LATER...
  10 DAYS LATER...
  TX: ALICE ---> PAY BACK LOAN

  ****** STATE 3 *******
  ****** BALANCES ******
  Balance Of Admin        --->  99870075000000000000000
  Balance Of Alice        --->  3500000000000000000
  Balance Of Bobby        --->  101425000000000000000
  Balance Of Carla        --->  25000000000000000000
  Owner of the NFT        --->  0x07748403082b29a45abD6C124A37E6B14e6B1803

  ****** LOAN DATA ******
  loanPrincipalAmount        --->  0
  maximumRepaymentAmount     --->  0
  nftCollateralId            --->  0
  loanERC20Denomination      --->  0x0000000000000000000000000000000000000000
  loanDuration               --->  0
  interestRateForDuration    --->  0
  loanAdminFeeInBasisPoints  --->  0
  nftCollateralWrapper       --->  0x0000000000000000000000000000000000000000
  loanStartTime              --->  0
  nftCollateralContract      --->  0x0000000000000000000000000000000000000000
  borrower                   --->  0x0000000000000000000000000000000000000000
  LOAN LIQUIDATED / REPAYED  --->  true
```

TEST 6:

Script

Listing 10: NFTFi.t.sol

```
908 function test_6() public {
909     /*
 ↳ ************************************************************
 ↳  */
910     /* TEST 6: ATTACK LENDER WANTS TO STEAL NFT BY RENEGOTIATE
 ↳ LOAN EARLY */
911     /*
 ↳ ************************************************************
 ↳  */
912
913     console.log("******************************");
914     console.log("**********  TEST 6  **********");
915     console.log("******************************");
916     console.log(" ");
917
918     firstStep_test5();
919
920     // 1 DAY LATER
921     console.log("1 DAYS LATER...");
922     vm.warp(1 days);
923
924     // MSG HASH PREPARATION
925     uint256 id2;
926     assembly {
927         id2 := chainid()
928     }
929     uint256 _expiry = block.timestamp + 10; // + 10 seconds
930
931     // GETTING THE MESSAGE HASH
932     bytes32 message = keccak256(
933             abi.encodePacked(
934                 uint256(1),
935                 uint32(0),
936                 uint256(1_00000000000000000),
937                 uint256(0),
938                 abi.encodePacked(bobby, uint256(2), _expiry),
939                 address(directLoanFixedOfferRedeploy),
940                 id2
941             )
942         );
```

```
943
944     // EIP712 STANDARD
945     bytes32 signedMessage = ECDSA.toEthSignedMessageHash(message);
946
947     // GETTING THE V, R, S OF THE SIGNED MESSAGE
948     (uint8 v, bytes32 r, bytes32 s) = vm.sign(0xAC, signedMessage)
 ↳ ;
949     bytes memory v_bytes;
950     if(v == 27){v_bytes = hex"1b";} else {v_bytes = hex"1c";}
951     bytes memory _lenderSignature = bytes.concat(r, s, v_bytes);
952
953     // ALICE WANTS TO RENEGOTIATE
954     vm.prank(bobby);
955     vm.expectRevert();
956     directLoanFixedOfferRedeploy.renegotiateLoan(1, 0, 1
 ↳ _000000000000000000, 0, 2, _expiry, _lenderSignature);
957
958     // BOBBY TRIES TO LIQUIDATE THE LOAN
959     vm.prank(bobby);
960     vm.expectRevert();
961     directLoanFixedOfferRedeploy.liquidateOverdueLoan(1);
962
963     // CHECKING THAT THE STATE IS AS EXPECTED
964     (uint256 a, uint256 b, uint256 c, address d, uint32 e, uint32
 ↳ f, uint16 g, address h, uint64 i, address j, address k) =
 ↳ directLoanFixedOfferRedeploy.loanIdToLoan(1);
965     liquidated = directLoanFixedOfferRedeploy.
 ↳ loanRepaidOrLiquidated(1);
966     console.log("TX: ALICE ---> PAY BACK LOAN");
967     console.log(" ");
968     console.log("****** STATE 3 *******");
969     console.log("****** BALANCES ******");
970     console.log("Balance Of Admin        ---> " ,token.balanceOf(
 ↳ admin));
971     console.log("Balance Of Alice        ---> " ,token.balanceOf(
 ↳ alice));
972     console.log("Balance Of Bobby        ---> " ,token.balanceOf(
 ↳ bobby));
973     console.log("Balance Of Carla        ---> " ,token.balanceOf(
 ↳ carla));
974     console.log("Owner of the NFT        ---> " ,nftContract.
 ↳ ownerOf(1));
975     console.log(" ");
976     console.log("****** LOAN DATA ******");
```

```
977     console.log("loanPrincipalAmount        ---> " ,a);
978     console.log("maximumRepaymentAmount     ---> " ,b);
979     console.log("nftCollateralId            ---> " ,c);
980     console.log("loanERC20Denomination      ---> " ,d);
981     console.log("loanDuration               ---> " ,e);
982     console.log("interestRateForDuration    ---> " ,f);
983     console.log("loanAdminFeeInBasisPoints  ---> " ,g);
984     console.log("nftCollateralWrapper       ---> " ,h);
985     console.log("loanStartTime              ---> " ,i);
986     console.log("nftCollateralContract      ---> " ,j);
987     console.log("borrower                   ---> " ,k);
988     console.log("LOAN LIQUIDATED / REPAYED  ---> " ,liquidated);
989     console.log(" ");
990 }
```

Output

```
[PASS] test_6() (gas: 610594)
Logs:
  ****** SETTING ENVIRONMENT ******
  ****** STATE 1 *******
  ****** BALANCES ******
  Balance Of Admin        --->  99870000000000000000000
  Balance Of Alice        --->  5000000000000000000
  Balance Of Bobby        --->  10000000000000000000000
  Balance Of Carla        --->  25000000000000000000000
  Owner of the NFT        --->  0x07748403082b29a45abD6C124A37E6B14e6B1803


  ******************************
  **********  TEST 6  **********
  ******************************

  TX: ALICE ---> ACCEPT BOBBY'S OFFER

  ****** STATE 1 *******
  ****** BALANCES ******
  Balance Of Admin        --->  99870000000000000000000
  Balance Of Alice        --->  6000000000000000000
  Balance Of Bobby        --->  99000000000000000000000
  Balance Of Carla        --->  25000000000000000000000
  Owner of the NFT        --->  0xB9bc102f443F848dD14Fe4aDfdF987a73e5F594D

  ****** LOAN DATA ******
  loanPrincipalAmount        --->  1000000000000000000
  maximumRepaymentAmount     --->  1200000000000000000
  nftCollateralId            --->  1
  loanERC20Denomination      --->  0x16F2DAC52c23a99c6Ff692b5C9b0bA6f9E46f9cc
  loanDuration               --->  864000
  interestRateForDuration    --->  0
  loanAdminFeeInBasisPoints  --->  500
  nftCollateralWrapper       --->  0xFb5b63865c5AaF1C0C1C6a6053894626883A0782
  loanStartTime              --->  1
  nftCollateralContract      --->  0xec6f254a0C543E30697C04d1B6B6Bd77Fe8b6799
  borrower                   --->  0x07748403082b29a45abD6C124A37E6B14e6B1803
  LOAN LIQUIDATED / REPAYED  --->  false

  1 DAYS LATER...
  TX: ALICE ---> PAY BACK LOAN

  ****** STATE 3 *******
  ****** BALANCES ******
  Balance Of Admin        --->  99870000000000000000000
  Balance Of Alice        --->  6000000000000000000
  Balance Of Bobby        --->  99000000000000000000000
  Balance Of Carla        --->  25000000000000000000000
  Owner of the NFT        --->  0xB9bc102f443F848dD14Fe4aDfdF987a73e5F594D

  ****** LOAN DATA ******
  loanPrincipalAmount        --->  1000000000000000000
  maximumRepaymentAmount     --->  1200000000000000000
  nftCollateralId            --->  1
  loanERC20Denomination      --->  0x16F2DAC52c23a99c6Ff692b5C9b0bA6f9E46f9cc
  loanDuration               --->  864000
  interestRateForDuration    --->  0
  loanAdminFeeInBasisPoints  --->  500
  nftCollateralWrapper       --->  0xFb5b63865c5AaF1C0C1C6a6053894626883A0782
  loanStartTime              --->  1
  nftCollateralContract      --->  0xec6f254a0C543E30697C04d1B6B6Bd77Fe8b6799
  borrower                   --->  0x07748403082b29a45abD6C124A37E6B14e6B1803
  LOAN LIQUIDATED / REPAYED  --->  false
```

AUTOMATED TESTING

5.1 STATIC ANALYSIS REPORT

Description:

Halborn used automated testing techniques to enhance the coverage of certain areas of the scoped contracts. Among the tools used was Slither, a Solidity static analysis framework. After Halborn verified all the contracts in the repository and was able to compile them correctly into their ABI and binary formats, Slither was run on the all-scoped contracts. This tool can statically verify mathematical relationships between Solidity variables to detect invalid or inconsistent usage of the contracts' APIs across the entire code-base.

Slither results:

DirectLoanFixedOfferRedeploy.sol

```
[Reentrancy in DirectLoanBaseMinimal.payBackLoan(uint32) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#490-508):
        External calls:
[       - _payBackLoan(_loanId,borrower,lender,loan) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#499)
[               - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#93)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
[               - IERC20(_loan.loanERC20Denomination).safeTransferFrom(msg.sender,_lender,payoffAmount) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#921)
                - IERC20(_loan.loanERC20Denomination).safeTransferFrom(msg.sender,loanExtras.revenueSharePartner,revenueShare) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#933-937)
                - IERC20(_loan.loanERC20Denomination).safeTransferFrom(msg.sender,owner(),adminFee) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#940)
        - _resolveLoan(_loanId,borrower,loan,loanCoordinator) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#501)
                - Address.functionDelegateCall(_loanTerms.nftCollateralWrapper,abi.encodeWithSelector(INftWrapper(_loanTerms.nftCollateralWrapper).transferNFT.selector,_sender,_recipient,_loanTerms.nftCollateralCon
ransferred) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#885-895)
[               - _loanCoordinator.resolveLoan(_loanId) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#999)
[               - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#191)
        External calls sending eth:
        - _payBackLoan(_loanId,borrower,lender,loan) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#499)
[               - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
[       State variables written after the call(s):
[       - delete loanIdToLoan[_loanId] (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#506)
[       - delete loanIdToLoanExtras[_loanId] (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#507)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities
[
[Reentrancy in DirectLoanBaseMinimal._renegotiateLoan(uint32,uint32,uint256,uint256,uint256,bytes) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#729-796):
[       External calls:
        - IERC20(loan.loanERC20Denomination).safeTransferFrom(borrower,lender,_renegotiationFee - renegotiationAdminFee) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#775-779)
        - IERC20(loan.loanERC20Denomination).safeTransferFrom(borrower,owner(),renegotiationAdminFee) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#781)
        State variables written after the call(s):
        - loan.loanDuration = _newLoanDuration (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#784)
        - loan.maximumRepaymentAmount = _newMaximumRepaymentAmount (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#785)
Reentrancy in DirectLoanBaseMinimal.liquidateOverdueLoan(uint32) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#524-562):
        External calls:
        - _resolveLoan(_loanId,lender,loan,loanCoordinator) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#543)
                - Address.functionDelegateCall(_loanTerms.nftCollateralWrapper,abi.encodeWithSelector(INftWrapper(_loanTerms.nftCollateralWrapper).transferNFT.selector,_sender,_recipient,_loanTerms.nftCollateralCon
ransferred) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#885-895)
                - _loanCoordinator.resolveLoan(_loanId) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#999)
                - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#191)
        State variables written after the call(s):
        - delete loanIdToLoan[_loanId] (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#560)
Reentrancy in DirectLoanBaseMinimal.mintObligationReceipt(uint32) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#417-425):
        External calls:
        - loanCoordinator.mintObligationReceipt(_loanId,borrower) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#422)
        State variables written after the call(s):
        - delete loanIdToLoan[_loanId].borrower (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#424)
Reentrancy in DirectLoanBaseMinimal.wrapCollateral(uint32) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#609-618):
        External calls:
        - (instance,receiverId) = LoanAirdropUtils.wrapCollateral(_loanId,loan,hub) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#616)
        State variables written after the call(s):
        - _escrowTokens[instance][receiverId] += 1 (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#617)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

DirectLoanBaseMinimal._renegotiateLoan(uint32,uint32,uint256,uint256,uint256,uint256,bytes).renegotiationAdminFee (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#762) is a local variable never
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

DirectLoanBaseMinimal._transferNFT(LoanData.LoanTerms,address,address) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#880-896) ignores return value by Address.functionDelegateCall(_loanTerms.n
loanTerms.nftCollateralWrapper).transferNFT.selector,_sender,_recipient,_loanTerms.nftCollateralContract,_loanTerms.nftCollateralId),NFT not successfully transferred) (../NFTFI/contracts/loans/direct/loanTypes/Dire
LoanAirdropUtils._transferNFT(LoanData.LoanTerms,address,address) (../NFTFI/contracts/loans/direct/loanTypes/LoanAirdropUtils.sol#183-199) ignores return value by Address.functionDelegateCall(_loan.nftCollateralWra
ralWrapper).transferNFT.selector,_sender,_recipient,_loan.nftCollateralContract,_loan.nftCollateralId),NFT not successfully transferred) (../NFTFI/contracts/loans/direct/loanTypes/LoanAirdropUtils.sol#188-198)
LoanAirdropUtils._transferNFTtoAirdropReceiver(LoanData.LoanTerms,address,address) (../NFTFI/contracts/loans/direct/loanTypes/LoanAirdropUtils.sol#209-225) ignores return value by Address.functionDelegateCall(_loan
(_loan.nftCollateralWrapper).wrapAirdropReceiver.selector,_airdropReceiverInstance,_loan.nftCollateralContract,_loan.nftCollateralId,_airdropBeneficiary),NFT was not successfully migrated) (../NFTFI/contracts/loans
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

Variable 'ECDSA.tryRecover(bytes32,bytes).r (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#62)' in ECDSA.tryRecover(bytes32,bytes) (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.
d(uint256)(signature + 0x20) (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#79)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

Reentrancy in DirectLoanBaseMinimal._createLoan(bytes32,LoanData.LoanTerms,LoanData.LoanExtras,address,address,address) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#809-822):
        External calls:
        - _transferNFT(_loanTerms,_borrower,address(this)) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#819)
                - Address.functionDelegateCall(_loanTerms.nftCollateralWrapper,abi.encodeWithSelector(INftWrapper(_loanTerms.nftCollateralWrapper).transferNFT.selector,_sender,_recipient,_loanTerms.nftCollateralCon
ransferred) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#885-895)
                - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#191)
        - _createLoanNoNftTransfer(_loanType,_loanTerms,_loanExtras,_borrower,_lender,_referrer) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#821)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#93)
```

```
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
                - IERC20(_loanTerms.loanERC20Denomination).safeTransferFrom(_lender,_referrer,referralFee) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#853)
                - IERC20(_loanTerms.loanERC20Denomination).safeTransferFrom(_lender,_borrower,principalAmount) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#856)
                - loanId = loanCoordinator.registerLoan(_lender,_loanType) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#863)
        External calls sending eth:
        - _createLoanNoNftTransfer(_loanType,_loanTerms,_loanExtras,_borrower,_lender,_referrer) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#821)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
        State variables written after the call(s):
        - _createLoanNoNftTransfer(_loanType,_loanTerms,_loanExtras,_borrower,_lender,_referrer) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#821)
                - loanIdToLoan[loanId] = _loanTerms (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#867)
        - _createLoanNoNftTransfer(_loanType,_loanTerms,_loanExtras,_borrower,_lender,_referrer) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#821)
                - loanIdToLoanExtras[loanId] = _loanExtras (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#868)
Reentrancy in DirectLoanBaseMinimal._createLoanNoNftTransfer(bytes32,LoanData.LoanTerms,LoanData.LoanExtras,address,address,address) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#835-871):
        External calls:
        - IERC20(_loanTerms.loanERC20Denomination).safeTransferFrom(_lender,_referrer,referralFee) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#853)
        - IERC20(_loanTerms.loanERC20Denomination).safeTransferFrom(_lender,_borrower,principalAmount) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#856)
        - loanId = loanCoordinator.registerLoan(_lender,_loanType) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#863)
        State variables written after the call(s):
        - loanIdToLoan[loanId] = _loanTerms (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#867)
        - loanIdToLoanExtras[loanId] = _loanExtras (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#868)
Reentrancy in DirectLoanBaseMinimal.liquidateOverdueLoan(uint32) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#524-562):
        External calls:
        - _resolveLoan(_loanId,lender,loan,loanCoordinator) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#543)
                - Address.functionDelegateCall(_loanTerms.nftCollateralWrapper,abi.encodeWithSelector(INftWrapper(_loanTerms.nftCollateralWrapper).transferNFT.selector,_sender,_recipient,_loanTerms.nftCollateralCor
ransferred) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#885-895)
                - _loanCoordinator.resolveLoan(_loanId) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#999)
                - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#191)
        State variables written after the call(s):
        - delete loanIdToLoanExtras[_loanId] (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#561)
Reentrancy in DirectLoanBaseMinimal.payBackLoan(uint32) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#498-508):
        External calls:
        - _payBackLoan(_loanId,borrower,lender,loan) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#499)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#93)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
                - IERC20(_loan.loanERC20Denomination).safeTransferFrom(msg.sender,_lender,payoffAmount) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#921)
                - IERC20(_loan.loanERC20Denomination).safeTransferFrom(msg.sender,loanExtras.revenueSharePartner,revenueShare) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#933-937)
                - IERC20(_loan.loanERC20Denomination).safeTransferFrom(msg.sender,owner(),adminFee) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#940)
        - _resolveLoan(_loanId,borrower,loan,loanCoordinator) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#501)
                - Address.functionDelegateCall(_loanTerms.nftCollateralWrapper,abi.encodeWithSelector(INftWrapper(_loanTerms.nftCollateralWrapper).transferNFT.selector,_sender,_recipient,_loanTerms.nftCollateralCor
ransferred) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#885-895)
                - _loanCoordinator.resolveLoan(_loanId) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#999)
                - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#191)
        External calls sending eth:
        - _payBackLoan(_loanId,borrower,lender,loan) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#499)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
        State variables written after the call(s):
        - _resolveLoan(_loanId,borrower,loan,loanCoordinator) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#501)
                - loanRepaidOrLiquidated[_loanId] = true (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#994)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in DirectLoanFixedOffer._acceptOffer(LoanData.LoanTerms,LoanData.LoanExtras,LoanData.Offer,LoanData.Signature) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanFixedOffer.sol#150-182):
        External calls:
        - loanId = _createLoan(LOAN_TYPE(),_loanTerms,_loanExtras,msg.sender,_signature.signer,_offer.referrer) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanFixedOffer.sol#171-178)
                - Address.functionDelegateCall(_loanTerms.nftCollateralWrapper,abi.encodeWithSelector(INftWrapper(_loanTerms.nftCollateralWrapper).transferNFT.selector,_sender,_recipient,_loanTerms.nftCollateralCor
ransferred) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#885-895)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#93)
                - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#191)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
                - IERC20(_loanTerms.loanERC20Denomination).safeTransferFrom(_lender,_referrer,referralFee) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#853)
                - IERC20(_loanTerms.loanERC20Denomination).safeTransferFrom(_lender,_borrower,principalAmount) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#856)
                - loanId = loanCoordinator.registerLoan(_lender,_loanType) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#863)
        External calls sending eth:
        - loanId = _createLoan(LOAN_TYPE(),_loanTerms,_loanExtras,msg.sender,_signature.signer,_offer.referrer) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanFixedOffer.sol#171-178)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
        Event emitted after the call(s):
        - LoanStarted(loanId,msg.sender,_signature.signer,_loanTerms,_loanExtras) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanFixedOffer.sol#181)
Reentrancy in DirectLoanBaseMinimal._payBackLoan(uint32,address,address,LoanData.LoanTerms) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#909-956):
        External calls:
        - IERC20(_loan.loanERC20Denomination).safeTransferFrom(msg.sender,_lender,payoffAmount) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#921)
        - IERC20(_loan.loanERC20Denomination).safeTransferFrom(msg.sender,loanExtras.revenueSharePartner,revenueShare) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#933-937)
        - IERC20(_loan.loanERC20Denomination).safeTransferFrom(msg.sender,owner(),adminFee) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#940)
        Event emitted after the call(s):
        - LoanRepaid(_loanId,_borrower,_lender,_loan.loanPrincipalAmount,_loan.nftCollateralId,payoffAmount,adminFee,revenueShare,loanExtras.revenueSharePartner,_loan.nftCollateralContract,_loan.loanERC20Denominati
nBaseMinimal.sol#943-955)
Reentrancy in DirectLoanBaseMinimal._renegotiateLoan(uint32,uint32,uint256,uint256,uint256,uint256,bytes) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#729-796):
        External calls:
        - IERC20(loan.loanERC20Denomination).safeTransferFrom(borrower,lender,_renegotiationFee - renegotiationAdminFee) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#775-779)
        - IERC20(loan.loanERC20Denomination).safeTransferFrom(borrower,owner(),renegotiationAdminFee) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#781)
        Event emitted after the call(s):
        - LoanRenegotiated(_loanId,borrower,lender,_newLoanDuration,_newMaximumRepaymentAmount,_renegotiationFee,renegotiationAdminFee) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#787-795)
Reentrancy in DirectLoanBaseMinimal.liquidateOverdueLoan(uint32) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#524-562):
        External calls:
        - _resolveLoan(_loanId,lender,loan,loanCoordinator) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#543)
                - Address.functionDelegateCall(_loanTerms.nftCollateralWrapper,abi.encodeWithSelector(INftWrapper(_loanTerms.nftCollateralWrapper).transferNFT.selector,_sender,_recipient,_loanTerms.nftCollateralCor
ransferred) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#885-895)
                - _loanCoordinator.resolveLoan(_loanId) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#999)
                - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#191)
        Event emitted after the call(s):
        - LoanLiquidated(_loanId,borrower,lender,loan.loanPrincipalAmount,loan.nftCollateralId,loanMaturityDate,block.timestamp,loan.nftCollateralContract) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseM
Reentrancy in LoanAirdropUtils.pullAirdrop(uint32,LoanData.LoanTerms,address,bytes,address,uint256,bool,uint256,INftfiHub) (../NFTFI/contracts/loans/direct/loanTypes/LoanAirdropUtils.sol#65-132):
        External calls:
        - _transferNFT(_loan,address(this),address(airdropFlashLoan)) (../NFTFI/contracts/loans/direct/loanTypes/LoanAirdropUtils.sol#98)
                - Address.functionDelegateCall(_loan.nftCollateralWrapper,abi.encodeWithSelector(INftWrapper(_loan.nftCollateralWrapper).transferNFT.selector,_sender,_recipient,_loan.nftCollateralContract,_loan.nft
I/contracts/loans/direct/loanTypes/LoanAirdropUtils.sol#188-198)
                - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#191)
        - airdropFlashLoan.pullAirdrop(_loan.nftCollateralContract,_loan.nftCollateralId,_loan.nftCollateralWrapper,_target,_data,_nftAirdrop,_nftAirdropId,_is1155,_nftAirdropAmount,borrower) (../NFTFI/contracts/lo
        Event emitted after the call(s):
        - AirdropPulledFlashloan(_loanId,borrower,_loan.nftCollateralId,_loan.nftCollateralContract,_target,_data) (../NFTFI/contracts/loans/direct/loanTypes/LoanAirdropUtils.sol#124-131)
Reentrancy in LoanAirdropUtils.wrapCollateral(uint32,LoanData.LoanTerms,INftfiHub) (../NFTFI/contracts/loans/direct/loanTypes/LoanAirdropUtils.sol#134-174):
        External calls:
        - (instance,receiverId) = factory.createAirdropReceiver(address(this)) (../NFTFI/contracts/loans/direct/loanTypes/LoanAirdropUtils.sol#157)
        - _transferNFTtoAirdropReceiver(_loan,instance,borrower) (../NFTFI/contracts/loans/direct/loanTypes/LoanAirdropUtils.sol#160)
                - Address.functionDelegateCall(_loan.nftCollateralWrapper,abi.encodeWithSelector(INftWrapper(_loan.nftCollateralWrapper).wrapAirdropReceiver.selector,_airdropReceiverInstance,_loan.nftCollateralCont
not successfully migrated) (../NFTFI/contracts/loans/direct/loanTypes/LoanAirdropUtils.sol#214-224)
                - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#191)
        Event emitted after the call(s):
        - CollateralWrapped(_loanId,borrower,_loan.nftCollateralId,_loan.nftCollateralContract,receiverId,instance) (../NFTFI/contracts/loans/direct/loanTypes/LoanAirdropUtils.sol#162-169)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

DirectLoanBaseMinimal.liquidateOverdueLoan(uint32) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#524-562) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp > loanMaturityDate,Loan is not overdue yet) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#539)
DirectLoanFixedOffer._acceptOffer(LoanData.LoanTerms,LoanData.LoanExtras,LoanData.Offer,LoanData.Signature) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanFixedOffer.sol#150-182) uses timestamp for comparisor
        Dangerous comparisons:
        - require(bool,string)(_loanTerms.nftCollateralContract != bundle,Collateral cannot be bundle) (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanFixedOffer.sol#169)
LoanChecksAndCalculations.payBackChecks(uint32,INftfiHub) (../NFTFI/contracts/loans/direct/loanTypes/LoanChecksAndCalculations.sol#28-42) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp <= (uint256(loanStartTime) + uint256(loanDuration)),Loan is expired) (../NFTFI/contracts/loans/direct/loanTypes/LoanChecksAndCalculations.sol#41)
LoanChecksAndCalculations.renegotiationChecks(LoanData.LoanTerms,uint32,uint32,uint256,uint256,INftfiHub) (../NFTFI/contracts/loans/direct/loanTypes/LoanChecksAndCalculations.sol#96-139) uses timestamp for comparis
        Dangerous comparisons:
        - require(bool,string)(block.timestamp <= (uint256(_loan.loanStartTime) + _newLoanDuration),New duration already expired) (../NFTFI/contracts/loans/direct/loanTypes/LoanChecksAndCalculations.sol#119)
NFTfiSigningUtils.isValidBorrowerSignature(LoanData.ListingTerms,LoanData.Signature,address) (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#102-128) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp <= _signature.expiry,Borrower Signature has expired) (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#107)
NFTfiSigningUtils.isValidBorrowerSignatureBundle(LoanData.ListingTerms,IBundleBuilder.BundleElements,LoanData.Signature,address) (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#202-230) uses timestamp for comparisc
        Dangerous comparisons:
        - require(bool,string)(block.timestamp <= _signature.expiry,Borrower Signature has expired) (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#208)
NFTfiSigningUtils.isValidLenderSignature(LoanData.Offer,LoanData.Signature,address) (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#299-328) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp <= _signature.expiry,Lender Signature has expired) (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#304)
NFTfiSigningUtils.isValidLenderSignatureBundle(LoanData.Offer,IBundleBuilder.BundleElements,LoanData.Signature,address) (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#391-419) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp <= _signature.expiry,Lender Signature has expired) (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#397)
NFTfiSigningUtils.isValidLenderRenegotiationSignature(uint256,uint32,uint256,uint256,LoanData.Signature,address) (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#478-510) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp <= _signature.expiry,Renegotiation Signature has expired) (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#486)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

ContractKeys.getIdFromStringKey(string) (../NFTFI/contracts/utils/ContractKeys.sol#31-38) uses assembly
        - INLINE ASM (../NFTFI/contracts/utils/ContractKeys.sol#35-37)
NFTfiSigningUtils.getChainID() (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#23-30) uses assembly
        - INLINE ASM (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#26-28)
Address.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#201-221) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#213-216)
ECDSA.tryRecover(bytes32,bytes) (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#57-86) uses assembly
```

```
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#67-71)
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#78-81)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity are used:
        - Version used: ['0.8.4', '^0.8.0', '^0.8.1']
        - 0.8.4 (../NFTFI/contracts/airdrop/IAirdropReceiverFactory.sol#3)
        - 0.8.4 (../NFTFI/contracts/interfaces/IAirdropFlashLoan.sol#3)
        - 0.8.4 (../NFTFI/contracts/interfaces/IBundleBuilder.sol#3)
        - 0.8.4 (../NFTFI/contracts/interfaces/IDirectLoanCoordinator.sol#3)
        - 0.8.4 (../NFTFI/contracts/interfaces/INftWrapper.sol#3)
        - 0.8.4 (../NFTFI/contracts/interfaces/INftfiHub.sol#3)
        - 0.8.4 (../NFTFI/contracts/interfaces/IPermittedERC20s.sol#3)
        - 0.8.4 (../NFTFI/contracts/interfaces/IPermittedNFTs.sol#3)
        - 0.8.4 (../NFTFI/contracts/interfaces/IPermittedPartners.sol#3)
        - 0.8.4 (../NFTFI/contracts/loans/BaseLoan.sol#3)
        - 0.8.4 (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#3)
        - 0.8.4 (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanFixedOffer.sol#3)
        - 0.8.4 (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanFixedOfferRedeploy.sol#3)
        - 0.8.4 (../NFTFI/contracts/loans/direct/loanTypes/IDirectLoanBase.sol#5)
        - 0.8.4 (../NFTFI/contracts/loans/direct/loanTypes/LoanAirdropUtils.sol#3)
        - 0.8.4 (../NFTFI/contracts/loans/direct/loanTypes/LoanChecksAndCalculations.sol#3)
        - 0.8.4 (../NFTFI/contracts/loans/direct/loanTypes/LoanData.sol#3)
        - 0.8.4 (../NFTFI/contracts/utils/ContractKeys.sol#3)
        - 0.8.4 (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#3)
        - 0.8.4 (../NFTFI/contracts/utils/NftReceiver.sol#2)
        - 0.8.4 (../NFTFI/contracts/utils/Ownable.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/interfaces/IERC1271.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/security/Pausable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/security/ReentrancyGuard.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155/IERC1155.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155/IERC1155Receiver.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/IERC721.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/IERC721Receiver.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/utils/ERC721Holder.sol#4)
        - ^0.8.1 (node_modules/@openzeppelin/contracts/utils/Address.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Strings.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/cryptography/SignatureChecker.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/IERC165.sol#4)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Address.functionCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#85-87) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#114-120) is never used and should be removed
Address.functionDelegateCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#174-176) is never used and should be removed
Address.functionStaticCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#147-149) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#157-166) is never used and should be removed
Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#60-65) is never used and should be removed
Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol#21-23) is never used and should be removed
ECDSA._throwError(ECDSA.RecoverError) (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#23-35) is never used and should be removed
ECDSA.recover(bytes32,bytes) (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#102-106) is never used and should be removed
ECDSA.recover(bytes32,bytes32,bytes32) (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#130-138) is never used and should be removed
ECDSA.recover(bytes32,uint8,bytes32,bytes32) (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#181-190) is never used and should be removed
ECDSA.toEthSignedMessageHash(bytes) (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#214-216) is never used and should be removed
ECDSA.toTypedDataHash(bytes32,bytes32) (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#227-229) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#45-58) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#69-80) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#60-67) is never used and should be removed
Strings.toHexString(uint256) (node_modules/@openzeppelin/contracts/utils/Strings.sol#40-51) is never used and should be removed
Strings.toHexString(uint256,uint256) (node_modules/@openzeppelin/contracts/utils/Strings.sol#56-66) is never used and should be removed
Strings.toString(uint256) (node_modules/@openzeppelin/contracts/utils/Strings.sol#15-35) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/interfaces/IERC1271.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/security/Pausable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/security/ReentrancyGuard.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155/IERC1155.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155/IERC1155Receiver.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/IERC721.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/IERC721Receiver.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/utils/ERC721Holder.sol#4) allows old versions
Pragma version^0.8.1 (node_modules/@openzeppelin/contracts/utils/Address.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Strings.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/cryptography/SignatureChecker.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/IERC165.sol#4) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#60-65):
        - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts/utils/Address.sol#63)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#128-139):
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
Low level call in Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#157-166):
        - (success,returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#164)
Low level call in Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#184-193):
        - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#191)
Low level call in SignatureChecker.isValidSignatureNow(address,bytes32,bytes) (node_modules/@openzeppelin/contracts/utils/cryptography/SignatureChecker.sol#25-39):
        - (success,result) = signer.staticcall(abi.encodeWithSelector(IERC1271.isValidSignature.selector,hash,signature)) (node_modules/@openzeppelin/contracts/utils/cryptography/SignatureChecker.sol#35-37)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter DirectLoanBaseMinimal.updateMaximumLoanDuration(uint256)._newMaximumLoanDuration (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#321) is not in mixedCase
Parameter DirectLoanBaseMinimal.updateAdminFee(uint16)._newAdminFeeInBasisPoints (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#334) is not in mixedCase
Parameter DirectLoanBaseMinimal.drainERC20Airdrop(address,address)._tokenAddress (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#346) is not in mixedCase
Parameter DirectLoanBaseMinimal.drainERC20Airdrop(address,address)._receiver (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#346) is not in mixedCase
Parameter DirectLoanBaseMinimal.setERC20Permit(address,bool)._erc20 (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#360) is not in mixedCase
Parameter DirectLoanBaseMinimal.setERC20Permit(address,bool)._permit (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#360) is not in mixedCase
Parameter DirectLoanBaseMinimal.setERC20Permits(address[],bool[])._erc20s (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#371) is not in mixedCase
Parameter DirectLoanBaseMinimal.setERC20Permits(address[],bool[])._permits (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#371) is not in mixedCase
Parameter DirectLoanBaseMinimal.drainERC721Airdrop(address,uint256,address)._tokenAddress (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#388) is not in mixedCase
Parameter DirectLoanBaseMinimal.drainERC721Airdrop(address,uint256,address)._tokenId (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#389) is not in mixedCase
Parameter DirectLoanBaseMinimal.drainERC721Airdrop(address,uint256,address)._receiver (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#390) is not in mixedCase
Parameter DirectLoanBaseMinimal.drainERC1155Airdrop(address,uint256,address)._tokenAddress (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#406) is not in mixedCase
Parameter DirectLoanBaseMinimal.drainERC1155Airdrop(address,uint256,address)._tokenId (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#407) is not in mixedCase
Parameter DirectLoanBaseMinimal.drainERC1155Airdrop(address,uint256,address)._receiver (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#408) is not in mixedCase
Parameter DirectLoanBaseMinimal.mintObligationReceipt(uint32)._loanId (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#417) is not in mixedCase
Parameter DirectLoanBaseMinimal.renegotiateLoan(uint32,uint32,uint256,uint256,uint256,uint256,bytes)._loanId (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#460) is not in mixedCase
Parameter DirectLoanBaseMinimal.renegotiateLoan(uint32,uint32,uint256,uint256,uint256,uint256,bytes)._newLoanDuration (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#461) is not in mixedCase
Parameter DirectLoanBaseMinimal.renegotiateLoan(uint32,uint32,uint256,uint256,uint256,uint256,bytes)._newMaximumRepaymentAmount (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#462) is not in mi
Parameter DirectLoanBaseMinimal.renegotiateLoan(uint32,uint32,uint256,uint256,uint256,uint256,bytes)._renegotiationFee (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#463) is not in mixedCase
Parameter DirectLoanBaseMinimal.renegotiateLoan(uint32,uint32,uint256,uint256,uint256,uint256,bytes)._lenderNonce (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#464) is not in mixedCase
Parameter DirectLoanBaseMinimal.renegotiateLoan(uint32,uint32,uint256,uint256,uint256,uint256,bytes)._expiry (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#465) is not in mixedCase
Parameter DirectLoanBaseMinimal.renegotiateLoan(uint32,uint32,uint256,uint256,uint256,uint256,bytes)._lenderSignature (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#466) is not in mixedCase
Parameter DirectLoanBaseMinimal.payBackLoan(uint32)._loanId (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#490) is not in mixedCase
Parameter DirectLoanBaseMinimal.liquidateOverdueLoan(uint32)._loanId (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#526) is not in mixedCase
Parameter DirectLoanBaseMinimal.pullAirdrop(uint32,address,bytes,address,uint256,bool,uint256)._loanId (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#577) is not in mixedCase
Parameter DirectLoanBaseMinimal.pullAirdrop(uint32,address,bytes,address,uint256,bool,uint256)._target (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#578) is not in mixedCase
Parameter DirectLoanBaseMinimal.pullAirdrop(uint32,address,bytes,address,uint256,bool,uint256)._data (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#579) is not in mixedCase
Parameter DirectLoanBaseMinimal.pullAirdrop(uint32,address,bytes,address,uint256,bool,uint256)._nftAirdrop (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#580) is not in mixedCase
Parameter DirectLoanBaseMinimal.pullAirdrop(uint32,address,bytes,address,uint256,bool,uint256)._nftAirdropId (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#581) is not in mixedCase
Parameter DirectLoanBaseMinimal.pullAirdrop(uint32,address,bytes,address,uint256,bool,uint256)._is1155 (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#582) is not in mixedCase
Parameter DirectLoanBaseMinimal.pullAirdrop(uint32,address,bytes,address,uint256,bool,uint256)._nftAirdropAmount (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#583) is not in mixedCase
Parameter DirectLoanBaseMinimal.wrapCollateral(uint32)._loanId (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#609) is not in mixedCase
Parameter DirectLoanBaseMinimal.cancelLoanCommitmentBeforeLoanHasBegun(uint256)._nonce (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#636) is not in mixedCase
Parameter DirectLoanBaseMinimal.getWhetherNonceHasBeenUsedForUser(address,uint256)._user (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#673) is not in mixedCase
Parameter DirectLoanBaseMinimal.getWhetherNonceHasBeenUsedForUser(address,uint256)._nonce (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#673) is not in mixedCase
Parameter DirectLoanBaseMinimal.getERC20Permit(address)._erc20 (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#684) is not in mixedCase
Variable DirectLoanBaseMinimal.LOAN_COORDINATOR (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#89) is not in mixedCase
Variable DirectLoanBaseMinimal._nonceHasBeenUsedForUser (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#136) is not in mixedCase
Parameter DirectLoanFixedOffer.acceptOffer(LoanData.Offer,LoanData.Signature,LoanData.BorrowerSettings)._offer (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanFixedOffer.sol#96) is not in mixedCase
Parameter DirectLoanFixedOffer.acceptOffer(LoanData.Offer,LoanData.Signature,LoanData.BorrowerSettings)._signature (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanFixedOffer.sol#97) is not in mixedCase
Parameter DirectLoanFixedOffer.acceptOffer(LoanData.Offer,LoanData.Signature,LoanData.BorrowerSettings)._borrowerSettings (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanFixedOffer.sol#98) is not in mixedCase
Function DirectLoanFixedOffer.LOAN_TYPE() (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanFixedOffer.sol#120-122) is not in mixedCase
Parameter DirectLoanFixedOffer.getPayoffAmount(uint32)._loanId (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanFixedOffer.sol#133) is not in mixedCase
Function DirectLoanFixedOfferRedeploy.LOAN_TYPE() (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanFixedOfferRedeploy.sol#90-92) is not in mixedCase
Function IDirectLoanBase.LOAN_COORDINATOR() (../NFTFI/contracts/loans/direct/loanTypes/IDirectLoanBase.sol#23) is not in mixedCase
Parameter LoanAirdropUtils.pullAirdrop(uint32,LoanData.LoanTerms,address,bytes,address,uint256,bool,uint256,INftfiHub)._loanId (../NFTFI/contracts/loans/direct/loanTypes/LoanAirdropUtils.sol#66) is not in mixedCas
Parameter LoanAirdropUtils.pullAirdrop(uint32,LoanData.LoanTerms,address,bytes,address,uint256,bool,uint256,INftfiHub)._loan (../NFTFI/contracts/loans/direct/loanTypes/LoanAirdropUtils.sol#67) is not in mixedCase
Parameter LoanAirdropUtils.pullAirdrop(uint32,LoanData.LoanTerms,address,bytes,address,uint256,bool,uint256,INftfiHub)._target (../NFTFI/contracts/loans/direct/loanTypes/LoanAirdropUtils.sol#68) is not in mixedCas
Parameter LoanAirdropUtils.pullAirdrop(uint32,LoanData.LoanTerms,address,bytes,address,uint256,bool,uint256,INftfiHub)._data (../NFTFI/contracts/loans/direct/loanTypes/LoanAirdropUtils.sol#69) is not in mixedCase
Parameter LoanAirdropUtils.pullAirdrop(uint32,LoanData.LoanTerms,address,bytes,address,uint256,bool,uint256,INftfiHub)._nftAirdrop (../NFTFI/contracts/loans/direct/loanTypes/LoanAirdropUtils.sol#70) is not in mixe
```

```
Parameter LoanAirdropUtils.pullAirdrop(uint32,LoanData.LoanTerms,address,bytes,address,uint256,bool,uint256,INftfiHub)._nftAirdropId (../NFTFI/contracts/loans/direct/loanTypes/LoanAirdropUtils.sol#71) is not in mix
Parameter LoanAirdropUtils.pullAirdrop(uint32,LoanData.LoanTerms,address,bytes,address,uint256,bool,uint256,INftfiHub)._isiLSS (../NFTFI/contracts/loans/direct/loanTypes/LoanAirdropUtils.sol#72) is not in mixedCase
Parameter LoanAirdropUtils.pullAirdrop(uint32,LoanData.LoanTerms,address,bytes,address,uint256,bool,uint256,INftfiHub)._nftAirdropAmount (../NFTFI/contracts/loans/direct/loanTypes/LoanAirdropUtils.sol#73) is not in
Parameter LoanAirdropUtils.pullAirdrop(uint32,LoanData.LoanTerms,address,bytes,address,uint256,bool,uint256,INftfiHub)._hub (../NFTFI/contracts/loans/direct/loanTypes/LoanAirdropUtils.sol#74) is not in mixedCase
Parameter LoanAirdropUtils.wrapCollateral(uint32,LoanData.LoanTerms,INftfiHub)._loanId (../NFTFI/contracts/loans/direct/loanTypes/LoanAirdropUtils.sol#135) is not in mixedCase
Parameter LoanAirdropUtils.wrapCollateral(uint32,LoanData.LoanTerms,INftfiHub)._loan (../NFTFI/contracts/loans/direct/loanTypes/LoanAirdropUtils.sol#136) is not in mixedCase
Parameter LoanAirdropUtils.wrapCollateral(uint32,LoanData.LoanTerms,INftfiHub)._hub (../NFTFI/contracts/loans/direct/loanTypes/LoanAirdropUtils.sol#137) is not in mixedCase
Parameter LoanChecksAndCalculations.payBackChecks(uint32,INftfiHub)._loanId (../NFTFI/contracts/loans/direct/loanTypes/LoanChecksAndCalculations.sol#28) is not in mixedCase
Parameter LoanChecksAndCalculations.payBackChecks(uint32,INftfiHub)._hub (../NFTFI/contracts/loans/direct/loanTypes/LoanChecksAndCalculations.sol#28) is not in mixedCase
Parameter LoanChecksAndCalculations.checkLoanIdValidity(uint32,INftfiHub)._loanId (../NFTFI/contracts/loans/direct/loanTypes/LoanChecksAndCalculations.sol#44) is not in mixedCase
Parameter LoanChecksAndCalculations.checkLoanIdValidity(uint32,INftfiHub)._hub (../NFTFI/contracts/loans/direct/loanTypes/LoanChecksAndCalculations.sol#44) is not in mixedCase
Parameter LoanChecksAndCalculations.getRevenueSharePercent(address,INftfiHub)._revenueSharePartner (../NFTFI/contracts/loans/direct/loanTypes/LoanChecksAndCalculations.sol#61) is not in mixedCase
Parameter LoanChecksAndCalculations.getRevenueSharePercent(address,INftfiHub)._hub (../NFTFI/contracts/loans/direct/loanTypes/LoanChecksAndCalculations.sol#61) is not in mixedCase
Parameter LoanChecksAndCalculations.renegotiationChecks(LoanData.LoanTerms,uint32,uint32,uint256,uint256,INftfiHub)._loan (../NFTFI/contracts/loans/direct/loanTypes/LoanChecksAndCalculations.sol#97) is not in mixed
Parameter LoanChecksAndCalculations.renegotiationChecks(LoanData.LoanTerms,uint32,uint32,uint256,uint256,INftfiHub)._loanId (../NFTFI/contracts/loans/direct/loanTypes/LoanChecksAndCalculations.sol#98) is not in mix
Parameter LoanChecksAndCalculations.renegotiationChecks(LoanData.LoanTerms,uint32,uint32,uint256,uint256,INftfiHub)._newLoanDuration (../NFTFI/contracts/loans/direct/loanTypes/LoanChecksAndCalculations.sol#99) is n
Parameter LoanChecksAndCalculations.renegotiationChecks(LoanData.LoanTerms,uint32,uint32,uint256,uint256,INftfiHub)._newMaximumRepaymentAmount (../NFTFI/contracts/loans/direct/loanTypes/LoanChecksAndCalculations.so
Parameter LoanChecksAndCalculations.renegotiationChecks(LoanData.LoanTerms,uint32,uint32,uint256,uint256,INftfiHub)._lenderNonce (../NFTFI/contracts/loans/direct/loanTypes/LoanChecksAndCalculations.sol#101) is not
Parameter LoanChecksAndCalculations.renegotiationChecks(LoanData.LoanTerms,uint32,uint32,uint256,uint256,INftfiHub)._hub (../NFTFI/contracts/loans/direct/loanTypes/LoanChecksAndCalculations.sol#102) is not in mixed
Parameter LoanChecksAndCalculations.computeRevenueShare(uint256,uint256)._adminFee (../NFTFI/contracts/loans/direct/loanTypes/LoanChecksAndCalculations.sol#153) is not in mixedCase
Parameter LoanChecksAndCalculations.computeRevenueShare(uint256,uint256)._revenueShareInBasisPoints (../NFTFI/contracts/loans/direct/loanTypes/LoanChecksAndCalculations.sol#153) is not in mixedCase
Parameter LoanChecksAndCalculations.computeAdminFee(uint256,uint256)._interestDue (../NFTFI/contracts/loans/direct/loanTypes/LoanChecksAndCalculations.sol#174) is not in mixedCase
Parameter LoanChecksAndCalculations.computeAdminFee(uint256,uint256)._adminFeeInBasisPoints (../NFTFI/contracts/loans/direct/loanTypes/LoanChecksAndCalculations.sol#174) is not in mixedCase
Parameter LoanChecksAndCalculations.computeReferralFee(uint256,uint256,address)._loanPrincipalAmount (../NFTFI/contracts/loans/direct/loanTypes/LoanChecksAndCalculations.sol#193) is not in mixedCase
Parameter LoanChecksAndCalculations.computeReferralFee(uint256,uint256,address)._referralFeeInBasisPoints (../NFTFI/contracts/loans/direct/loanTypes/LoanChecksAndCalculations.sol#194) is not in mixedCase
Parameter LoanChecksAndCalculations.computeReferralFee(uint256,uint256,address)._referrer (../NFTFI/contracts/loans/direct/loanTypes/LoanChecksAndCalculations.sol#195) is not in mixedCase
Parameter ContractKeys.getIdFromStringKey(string)._key (../NFTFI/contracts/utils/ContractKeys.sol#31) is not in mixedCase
Parameter NFTfiSigningUtils.isValidBorrowerSignature(LoanData.ListingTerms,LoanData.Signature)._listingTerms (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#90) is not in mixedCase
Parameter NFTfiSigningUtils.isValidBorrowerSignature(LoanData.ListingTerms,LoanData.Signature)._signature (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#90) is not in mixedCase
Parameter NFTfiSigningUtils.isValidBorrowerSignature(LoanData.ListingTerms,LoanData.Signature,address)._listingTerms (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#103) is not in mixedCase
Parameter NFTfiSigningUtils.isValidBorrowerSignature(LoanData.ListingTerms,LoanData.Signature,address)._signature (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#104) is not in mixedCase
Parameter NFTfiSigningUtils.isValidBorrowerSignature(LoanData.ListingTerms,LoanData.Signature,address)._loanContract (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#105) is not in mixedCase
Parameter NFTfiSigningUtils.isValidBorrowerSignatureBundle(LoanData.ListingTerms,IBundleBuilder.BundleElements,LoanData.Signature)._listingTerms (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#191) is not in mixedC
Parameter NFTfiSigningUtils.isValidBorrowerSignatureBundle(LoanData.ListingTerms,IBundleBuilder.BundleElements,LoanData.Signature)._bundleElements (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#192) is not in mixe
Parameter NFTfiSigningUtils.isValidBorrowerSignatureBundle(LoanData.ListingTerms,IBundleBuilder.BundleElements,LoanData.Signature)._signature (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#193) is not in mixedCase
Parameter NFTfiSigningUtils.isValidBorrowerSignatureBundle(LoanData.ListingTerms,IBundleBuilder.BundleElements,LoanData.Signature,address)._listingTerms (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#203) is not i
Parameter NFTfiSigningUtils.isValidBorrowerSignatureBundle(LoanData.ListingTerms,IBundleBuilder.BundleElements,LoanData.Signature,address)._bundleElements (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#204) is not
Parameter NFTfiSigningUtils.isValidBorrowerSignatureBundle(LoanData.ListingTerms,IBundleBuilder.BundleElements,LoanData.Signature,address)._signature (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#205) is not in m
Parameter NFTfiSigningUtils.isValidBorrowerSignatureBundle(LoanData.ListingTerms,IBundleBuilder.BundleElements,LoanData.Signature,address)._loanContract (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#206) is not i
Parameter NFTfiSigningUtils.isValidLenderSignature(LoanData.Offer,LoanData.Signature)._offer (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#287) is not in mixedCase
Parameter NFTfiSigningUtils.isValidLenderSignature(LoanData.Offer,LoanData.Signature)._signature (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#287) is not in mixedCase
Parameter NFTfiSigningUtils.isValidLenderSignature(LoanData.Offer,LoanData.Signature,address)._offer (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#300) is not in mixedCase
Parameter NFTfiSigningUtils.isValidLenderSignature(LoanData.Offer,LoanData.Signature,address)._signature (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#301) is not in mixedCase
Parameter NFTfiSigningUtils.isValidLenderSignature(LoanData.Offer,LoanData.Signature,address)._loanContract (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#302) is not in mixedCase
Parameter NFTfiSigningUtils.isValidLenderSignatureBundle(LoanData.Offer,IBundleBuilder.BundleElements,LoanData.Signature)._offer (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#380) is not in mixedCase
Parameter NFTfiSigningUtils.isValidLenderSignatureBundle(LoanData.Offer,IBundleBuilder.BundleElements,LoanData.Signature)._bundleElements (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#381) is not in mixedCase
Parameter NFTfiSigningUtils.isValidLenderSignatureBundle(LoanData.Offer,IBundleBuilder.BundleElements,LoanData.Signature)._signature (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#382) is not in mixedCase
Parameter NFTfiSigningUtils.isValidLenderSignatureBundle(LoanData.Offer,IBundleBuilder.BundleElements,LoanData.Signature,address)._offer (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#392) is not in mixedCase
Parameter NFTfiSigningUtils.isValidLenderSignatureBundle(LoanData.Offer,IBundleBuilder.BundleElements,LoanData.Signature,address)._bundleElements (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#393) is not in mixed
Parameter NFTfiSigningUtils.isValidLenderSignatureBundle(LoanData.Offer,IBundleBuilder.BundleElements,LoanData.Signature,address)._signature (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#394) is not in mixedCase
Parameter NFTfiSigningUtils.isValidLenderSignatureBundle(LoanData.Offer,IBundleBuilder.BundleElements,LoanData.Signature,address)._loanContract (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#395) is not in mixedCa
Parameter NFTfiSigningUtils.isValidLenderRenegotiationSignature(uint256,uint32,uint256,uint256,LoanData.Signature)._loanId (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#457) is not in mixedCase
Parameter NFTfiSigningUtils.isValidLenderRenegotiationSignature(uint256,uint32,uint256,uint256,LoanData.Signature)._newLoanDuration (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#458) is not in mixedCase
Parameter NFTfiSigningUtils.isValidLenderRenegotiationSignature(uint256,uint32,uint256,uint256,LoanData.Signature)._newMaximumRepaymentAmount (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#459) is not in mixedCase
Parameter NFTfiSigningUtils.isValidLenderRenegotiationSignature(uint256,uint32,uint256,uint256,LoanData.Signature)._renegotiationFee (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#460) is not in mixedCase
Parameter NFTfiSigningUtils.isValidLenderRenegotiationSignature(uint256,uint32,uint256,uint256,LoanData.Signature)._signature (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#461) is not in mixedCase
Parameter NFTfiSigningUtils.isValidLenderRenegotiationSignature(uint256,uint32,uint256,uint256,LoanData.Signature,address)._loanId (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#479) is not in mixedCase
Parameter NFTfiSigningUtils.isValidLenderRenegotiationSignature(uint256,uint32,uint256,uint256,LoanData.Signature,address)._newLoanDuration (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#480) is not in mixedCase
Parameter NFTfiSigningUtils.isValidLenderRenegotiationSignature(uint256,uint32,uint256,uint256,LoanData.Signature,address)._newMaximumRepaymentAmount (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#481) is not in m
Parameter NFTfiSigningUtils.isValidLenderRenegotiationSignature(uint256,uint32,uint256,uint256,LoanData.Signature,address)._renegotiationFee (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#482) is not in mixedCase
Parameter NFTfiSigningUtils.isValidLenderRenegotiationSignature(uint256,uint32,uint256,uint256,LoanData.Signature,address)._signature (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#483) is not in mixedCase
Parameter NFTfiSigningUtils.isValidLenderRenegotiationSignature(uint256,uint32,uint256,uint256,LoanData.Signature,address)._loanContract (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#484) is not in mixedCase
Parameter NFTfiSigningUtils.getEncodedListing(LoanData.ListingTerms)._listingTerms (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#515) is not in mixedCase
Parameter NFTfiSigningUtils.getEncodedOffer(LoanData.Offer)._offer (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#534) is not in mixedCase
Parameter NFTfiSigningUtils.getEncodedSignature(LoanData.Signature)._signature (../NFTFI/contracts/utils/NFTfiSigningUtils.sol#551) is not in mixedCase
Parameter NftReceiver.supportsInterface(bytes4)._interfaceId (../NFTFI/contracts/utils/NftReceiver.sol#48) is not in mixedCase
Parameter Ownable.transferOwnership(address)._newOwner (../NFTFI/contracts/utils/Ownable.sol#48) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Variable DirectLoanBaseMinimal.LOAN_COORDINATOR (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#89) is too similar to DirectLoanBaseMinimal._resolveLoan(uint32,address,LoanData.LoanTerms,IDirec
oans/direct/loanTypes/DirectLoanBaseMinimal.sol#971)
Variable DirectLoanBaseMinimal.LOAN_COORDINATOR (../NFTFI/contracts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#89) is too similar to DirectLoanBaseMinimal._resolveLoanNoNftTransfer(uint32,LoanData.LoanTerms,I
cts/loans/direct/loanTypes/DirectLoanBaseMinimal.sol#990)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

supportsInterface(bytes4) should be declared external:
        - NftReceiver.supportsInterface(bytes4) (../NFTFI/contracts/utils/NftReceiver.sol#48-53)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (../NFTFI/contracts/utils/Ownable.sol#48-51)
onERC721Received(address,address,uint256,bytes) should be declared external:
        - ERC721Holder.onERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/utils/ERC721Holder.sol#20-27)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

- As a result of the tests carried out with the Slither tool, some results were obtained and reviewed by Halborn. Based on the results reviewed, some vulnerabilities were determined to be false positives.

5.2 AUTOMATED SECURITY SCAN

Description:

Halborn used automated security scanners to assist with detection of well-known security issues, and to identify low-hanging fruits on the targets for this engagement. Among the tools used was MythX, a security analysis service for Ethereum smart contracts. MythX performed a scan on all the contracts and sent the compiled results to the analyzers to locate any vulnerabilities.

MythX results:

DirectLoanFixedOfferRedeploy.sol

Line	SWC Title	Severity	Short Description
305	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
306	(SWC-110) Assert Violation	Unknown	Out of bounds array access
374	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
375	(SWC-110) Assert Violation	Unknown	Out of bounds array access
538	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
615	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered
617	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
778	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
843	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
850	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
931	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered
996	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered

Report for contracts/loans/direct/loanTypes/DirectLoanFixedOffer.sol
https://dashboard.mythx.io/#/console/analyses/b0b00975-013d-4a15-a860-f3cba2c6b0c1

Line	SWC Title	Severity	Short Description
218	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
223	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered

Report for contracts/loans/direct/loanTypes/LoanChecksAndCalculations.sol
https://dashboard.mythx.io/#/console/analyses/b0b00975-013d-4a15-a860-f3cba2c6b0c1

Line	SWC Title	Severity	Short Description
41	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
119	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
158	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
158	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
175	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
175	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
200	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
200	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered

Report for node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol
https://dashboard.mythx.io/#/console/analyses/b0b00975-013d-4a15-a860-f3cba2c6b0c1

Line	SWC Title	Severity	Short Description
65	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
77	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered

Report for node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol
https://dashboard.mythx.io/#/console/analyses/b0b00975-013d-4a15-a860-f3cba2c6b0c1

Line	SWC Title	Severity	Short Description
65	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
77	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered

Report for node_modules/@openzeppelin/contracts/utils/Strings.sol
https://dashboard.mythx.io/#/console/analyses/b0b00975-013d-4a15-a860-f3cba2c6b0c1

Line	SWC Title	Severity	Short Description
25	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
26	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/=" discovered
30	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-=" discovered
31	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "%" discovered
31	(SWC-110) Assert Violation	Unknown	Out of bounds array access
31	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
32	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/=" discovered
47	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
57	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
57	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
58	(SWC-110) Assert Violation	Unknown	Out of bounds array access
59	(SWC-110) Assert Violation	Unknown	Out of bounds array access
60	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "--" discovered
60	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
60	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
61	(SWC-110) Assert Violation	Unknown	Out of bounds array access

Report for node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol
https://dashboard.mythx.io/#/console/analyses/b0b00975-013d-4a15-a860-f3cba2c6b0c1

Line	SWC Title	Severity	Short Description
121	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered

- No major issues found by Mythx. The reentrancy issue flagged by MythX is a false positive as the function is already protected against reentrancy attacks by using the nonreentrant modifer.

THANK YOU FOR CHOOSING

// HALBORN