// HALBORN

NFTfi - Native Punk Wrapper

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **January 17th, 2023 - February 10th, 2023**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY 3
CONTACTS 4
1 EXECUTIVE OVERVIEW 5
1.1 INTRODUCTION 6
1.2 AUDIT SUMMARY 6
1.3 TEST APPROACH & METHODOLOGY 6
RISK METHODOLOGY 7
1.4 SCOPE 9
2 ASSESSMENT SUMMARY & FINDINGS OVERVIEW 10
3 FINDINGS & TECH DETAILS 11
3.1 (HAL-01) AIRDROP RECEIVER FUNCTIONALITY DENIED - CRITICAL 13
Description 13
Code Location 13
Risk Level 14
Recommendation 14
Remediation Plan 14
4 MANUAL TESTING 16
4.1 SCENARIOS TESTED 17
TEST 1 27
TEST 1.1 30
TEST 2 33
TEST 3 36
TEST 4 38
TEST 5 40
TEST 6 44

TEST 7 47
TEST 8 50
TEST 8.1 53
TEST 9 56
5 AUTOMATED TESTING 60
5.1 STATIC ANALYSIS REPORT 61
Description 61
Slither results 62
5.2 AUTOMATED SECURITY SCAN 66
Description 66
MythX results 66

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	01/10/2023	Miguel Jalon
0.2	Document Updates	02/13/2023	Miguel Jalon
0.3	Final Draft	02/13/2023	Miguel Jalon
0.4	Draft Review	02/13/2023	Roberto Reigada
0.5	Draft Review	02/14/2023	Piotr Cielas
0.6	Draft Review	02/14/2023	Gabi Urrutia
1.0	Remediation Plan	02/27/2023	Miguel Jalon
1.1	Remediation Plan Review	02/28/2023	Grzegorz Trawinski
1.2	Remediation Plan Review	02/28/2023	Piotr Cielas
1.3	Remediation Plan Review	02/28/2023	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
Grzegorz Trawinski	Halborn	Grzegorz.Trawinski@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Native Punk Wrapper introduces support for Native Punk NFTs as collateral in the NFTfi protocol.

NFTfi engaged Halborn to conduct a security audit on their smart contracts beginning on January 17th, 2023 and ending on February 10th, 2023. The security assessment was scoped to the smart contracts provided to the Halborn team.

1.2 AUDIT SUMMARY

The team at Halborn was provided three weeks 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 a security risk that was 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.

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL

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:

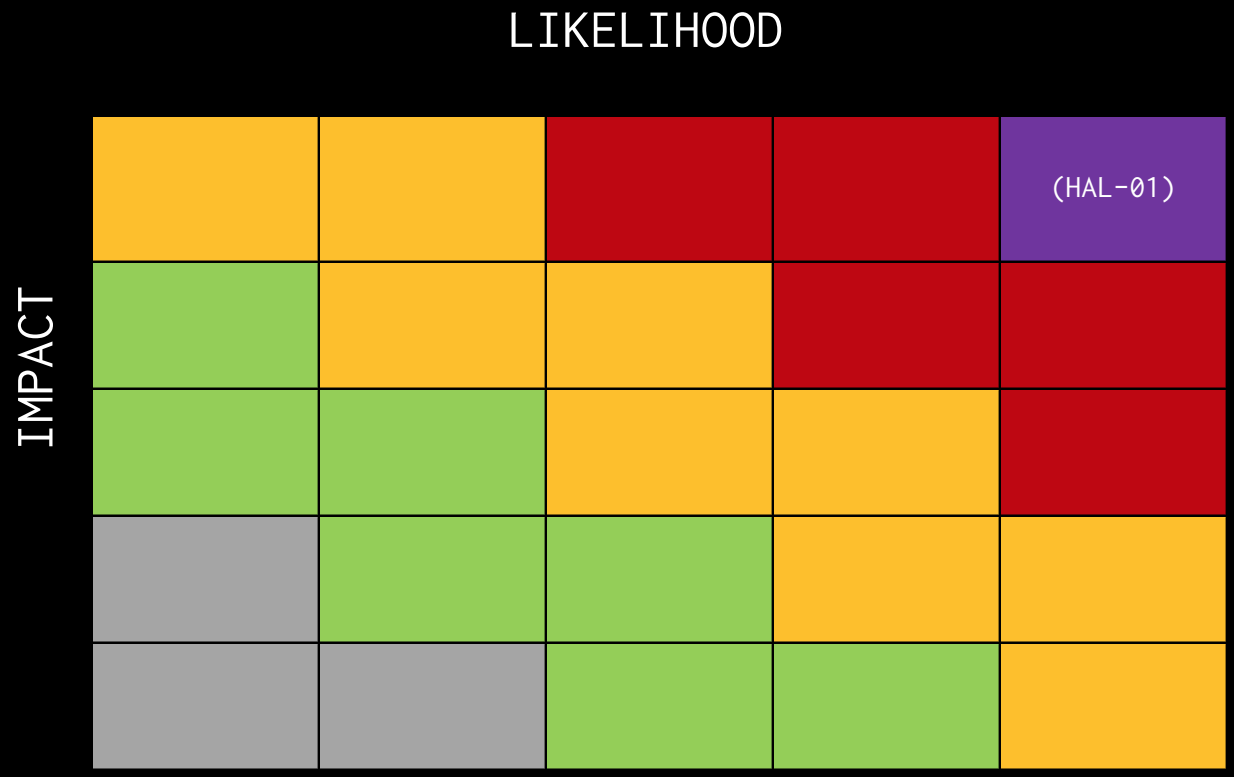
- PunkWrapper.sol

Commit ID:
- da20627039858015c2ef00dc76b68512f9befb4f

Fixed commit ID:
- 529a1d25aced6fc747e579ca900a4628b9c7cb16

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
1	0	0	0	0



SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL01 - WRAPAIRDROPRECEIVER FUNCTION UNUSABLE	Critical	SOLVED - 02/17/2023

FINDINGS & TECH DETAILS

3.1 (HAL-01) AIRDROP RECEIVER FUNCTIONALITY DENIED - CRITICAL

Description:

The borrower can never recover the collateral from the Escrow if the collateral is wrap to use the airdrop functionality. The AirdropReceiver functionality in the PunkWrapper contract is unusable, as it reverts to the transferNFT() function when it is called by the _resolveLoan() function in the contract of the offer used. The issue is that the airdropReceiver instance has to be sent to the borrower so then, being the owner, can drain all the airdrops inside the contract. However, punkWrapper is only prepared to transfer Punks, and then the AirdropReceiver that has to be sent to the borrower to claim the airdrop, reverts in the transaction because it is not prepared to transfer ERC721 Airdrop Functionality for the Punks is disabled and the borrower can never recover the collateral. The Proof of Concept is the test number 9 AirdropReceiver.

Code Location:

Listing 1: PunkWrapper.sol (Lines 30-35)

```
24     function transferNFT(
25         address _sender,
26         address _recipient,
27         address _nftContract,
28         uint256 _nftId
29     ) external override returns (bool) {
30         if (address(this) == _sender) {
31             IPunks(_nftContract).transferPunk(_recipient, _nftId);
32         } else {
33             require(isOwner(_sender, _nftContract, _nftId), "
↳ PunkWrapper:sender must be owner");
34             IPunks(_nftContract).buyPunk(_nftId);
35         }
36         return true;
37     }
```

Risk Level:

Likelihood - 5
Impact - 5

Recommendation:

The remediation is adding the functionality for the PunkWrapper to send not only Native Punks, but also ERC721 NFTs.

Remediation Plan:

SOLVED: The NFTfi team fixed the issue in commit 529a1d25 by adding a try-catch to the transferNFT function.

Listing 2: PunkWrapper.sol (Lines 35-38)

```
26     function transferNFT(
27         address _sender,
28         address _recipient,
29         address _nftContract,
30         uint256 _nftId
31     ) external override returns (bool) {
32         // we have to check if we are wrapped -
33         // ideally should be fixed in loan airdrop utils:
34         // should update collateralWrapper, but that would need a
↳ loans contract re-deploy
35         try IERC721(_nftContract).supportsInterface(type(IERC721).
↳ interfaceId) {
36             IERC721(_nftContract).transferFrom(_sender, _recipient
↳ , _nftId);
37             return true;
38         } catch {
39             if (address(this) == _sender) {
40                 IPunks(_nftContract).transferPunk(_recipient,
↳ _nftId);
41             } else {
42                 require(isOwner(_sender, _nftContract, _nftId), "
↳ PunkWrapper:sender must be owner");
43                 IPunks(_nftContract).buyPunk(_nftId);
44             }
45             return true;
```

```
46          }
47      }
```

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: General repayment cycle (Borrower - Escrow - Borrower)
- Test 1.1: Attack vector: External user repays loan to steal the NFT (revert expected)
- Test 2: General repayment cycle out of time (revert expected)
- Test 3: General liquidation cycle (Borrower - Escrow - Lender)
- Test 4: General liquidation cycle before agreement (revert expected)
- Test 5: General repayment cycle out of time with renegotiation
- Test 6: Attack vector: Borrower renegotiate their loan to steal the Punk (revert expected)
- Test 7: General repayment cycle with promissory note transfer
- Test 8: General repayment cycle with obligation receipt transfer
- Test 8.2: Attack vector: Obligation receipt transfer and old owner of OR repay loan.
- Test 9: General airdrop functionality (Airdrop receiver - Wrap - Unwrap)

Script

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

Listing 3: PunkTest.t.sol

```
1     // SPDX-License-Identifier: UNLICENSED
2 pragma solidity 0.8.4;
3
4 import "forge-std/Test.sol";
5 import "../src/NftfiHub.sol";
6 import "../src/loans/direct/loanTypes/DirectLoanFixedOfferRedeploy
↳ .sol";
7 import "@openzeppelin/contracts/utils/cryptography/ECDSA.sol";
8 import "../src/mocks/NFT.sol";
```

```
 9 import "../src/mocks/NFTWrapper.sol";
10 import "../src/test/DummyPunks.sol";
11 import "../src/nftTypeRegistry/nftTypes/PunkWrapper.sol";
12 import "../src/loans/direct/DirectLoanCoordinator.sol";
13 import "../src/mocks/SimpleERC20.sol";
14 import "../src/permittedLists/PermittedAirdrops.sol";
15 import "../src/permittedLists/PermittedNFTsAndTypeRegistry.sol";
16 import "../src/loans/direct/loanTypes/LoanData.sol";
17 import "../src/airdrop/AirdropReceiverFactory.sol";
18
19 contract punkTest is Test {
20     using ECDSA for bytes32;
21
22     NFTWrapper internal nftWrapper;
23     PunkWrapper internal punkWrapper;
24     NFT internal nftContract;
25     DummyPunks internal punkContract;
26     SimpleToken internal token;
27     SmartNft internal nftPromissoryNote;
28     SmartNft internal nftObligationReceipt;
29     NftfiHub internal nftFiHub;
30     DirectLoanCoordinator internal directLoanCoordinator;
31     DirectLoanFixedOfferRedeploy internal
↳ directLoanFixedOfferRedeploy;
32     PermittedNFTsAndTypeRegistry internal
↳ permittedNFTsAndTypeRegistry;
33     PermittedAirdrops internal permittedAirdrop;
34     AirdropReceiver internal airdropReceiverToClone;
35     AirdropReceiver internal airdropReceiver;
36     AirdropReceiverFactory internal airdropReceiverFactory;
37
38     address internal owner;
39     address internal admin;
40     address internal alice;
41     address internal bobby;
42     address internal carla;
43     address internal edgar;
44     address internal zeroo;
45     uint256 internal verifyingSignerPrivateKey;
46     address internal verifyingSigner;
47     address internal sender;
48     bytes4[] internal selectors;
49     address[] internal airdopContracts;
50     address[] internal permittedErc20s;
```

```
51      string[] internal contractKeys;
52      address[] internal contractAddresses;
53      string[] internal definedNftTypes;
54      address[] internal definedNftWrappers;
55      address[] internal permittedNftContracts;
56      string[] internal permittedNftTypes;
57      string[] internal loanTypes;
58      address[] internal loanContracts;
59      bool internal liquidated;
60      uint256 internal timeNow;
61
62      function setUp() public {
63          /* *************** */
64          /* ADDRESSES SETUP */
65          /* *************** */
66
67          // ADDRESSES DECLARATION
68          admin = vm.addr(0x60DDD);
69          alice = vm.addr(0xA71CE);
70          bobby = vm.addr(0xB0BB1);
71          carla = vm.addr(0xCA47A);
72          edgar = vm.addr(0xED6A4);
73          zeroo = address(0);
74
75          // 100 ETHER PER ADDRESS
76          vm.deal(admin, 100 ether);
77          vm.deal(alice, 100 ether);
78          vm.deal(bobby, 100 ether);
79          vm.deal(carla, 100 ether);
80          vm.deal(edgar, 100 ether);
81
82          // LABELING ADDRESSES
83          vm.label(admin, "admin");
84          vm.label(alice, "alice");
85          vm.label(bobby, "bobby");
86          vm.label(carla, "carla");
87          vm.label(edgar, "edgar");
88
89          /* ***************** */
90          /* ENVIRONMENT SETUP */
91          /* ***************** */
92
93          // DEPLYING NFTWRAPPER
94          vm.prank(admin);
```

```
95          punkWrapper = new PunkWrapper();
96          vm.prank(admin);
97          nftWrapper = new NFTWrapper();
98
99          // DEPLOYING NFT CONTRACT
100         nftContract = new NFT(address(nftWrapper));
101         punkContract = new DummyPunks{value: 30000000 gwei}(
 ↳ address(punkWrapper));
102
103         // MINT A PUNK TO ALICE
104         vm.prank(alice);
105         punkContract.mintPunk(alice, 0);
106         vm.prank(carla);
107         nftContract.mintNFT(10);
108
109         // DEPLOYING AND MINTING TOKEN
110         vm.startPrank(admin);
111         token = new SimpleToken("token", "TKN", 1000000
 ↳ _000000000000000000);
112         token.transfer(alice, 50_000000000000000000);
113         token.transfer(bobby, 1000_000000000000000000);
114         token.transfer(carla, 50_000000000000000000);
115
116         contractKeys.push('PERMITTED_NFTS');
117         contractKeys.push('PERMITTED_NFTS');
118         contractAddresses.push(address(nftContract));
119         contractAddresses.push(address(punkContract));
120         permittedErc20s.push(address(token));
121
122         // DEPLOYING NFTFI HUB
123         nftFiHub = new NftfiHub(admin, contractKeys,
 ↳ contractAddresses);
124         address nftFiHubAddr = address(nftFiHub);
125
126         // DEPLOYING PROMISORY NOTES AND OBLIGATION RECEPT
 ↳ CONTRACTS
127         nftObligationReceipt = new SmartNft(admin, address(
 ↳ nftFiHub), address(directLoanCoordinator), "nftObligationReceipt",
 ↳  "NOR", "customURI");
128         nftPromissoryNote = new SmartNft(admin, address(nftFiHub),
 ↳  address(directLoanCoordinator), "nftPromissoryNote", "NOR", "
 ↳ customURI");
129         nftObligationReceipt.setLoanCoordinator(address(
 ↳ directLoanCoordinator));
```

```
130         nftPromissoryNote.setLoanCoordinator(address(
 ↳ directLoanCoordinator));
131
132         // DEPLOYING DIRECT_LOAN_FIXED_OFFER_REDEPLOY
133         directLoanFixedOfferRedeploy = new
 ↳ DirectLoanFixedOfferRedeploy(admin, nftFiHubAddr, permittedErc20s)
 ↳ ;
134         address directLoanFixedOfferRedeployAddr = address(
 ↳ directLoanFixedOfferRedeploy);
135
136         // DEPLOYING DIRECT LOAN COORDINATOR
137         loanTypes.push("DIRECT_LOAN_FIXED_REDEPLOY");
138         loanContracts.push(address(directLoanFixedOfferRedeploy));
139         directLoanCoordinator = new DirectLoanCoordinator(address(
 ↳ nftFiHub), admin, loanTypes, loanContracts);
140
141         // INITIALIZING DIRECT LOAN COORDINATOR
142         directLoanCoordinator.initialize(address(nftPromissoryNote
 ↳ ), address(nftObligationReceipt));
143         nftObligationReceipt.setLoanCoordinator(address(
 ↳ directLoanCoordinator));
144         nftPromissoryNote.setLoanCoordinator(address(
 ↳ directLoanCoordinator));
145
146         // SETTING CONTRACTKEYS AND CONTRACTADDRRESSES
147         contractKeys.push('DIRECT_LOAN_COORDINATOR');
148         contractAddresses.push(address(directLoanCoordinator));
149         nftFiHub.setContract('DIRECT_LOAN_COORDINATOR', address(
 ↳ directLoanCoordinator));
150
151         // REGISTRATION OF PERMITTED NFTS AND TYPES
152         definedNftTypes.push('NFT');
153         definedNftTypes.push('PUNKS');
154         definedNftWrappers.push(address(nftWrapper));
155         definedNftWrappers.push(address(punkWrapper));
156         permittedNftContracts.push(address(nftContract));
157         permittedNftContracts.push(address(punkContract));
158         permittedNftTypes.push('NFT');
159         permittedNftTypes.push('PUNKS');
160         permittedNFTsAndTypeRegistry = new
 ↳ PermittedNFTsAndTypeRegistry(admin, nftFiHubAddr, definedNftTypes,
 ↳  definedNftWrappers, permittedNftContracts, permittedNftTypes);
161         vm.stopPrank();
162
```

```
163         // PERMITTED AIRDROPS SETTINGS
164         selectors.push(bytes4(keccak256(bytes("mintNFT(uint256)"))
 ↳ ));
165         airdopContracts.push(address(nftContract));
166         permittedAirdrop = new PermittedAirdrops(admin,
 ↳ airdopContracts, selectors);
167 
168         // AIRDROP RECEIVER (alt 2)
169         vm.startPrank(admin);
170         airdropReceiverFactory = new AirdropReceiverFactory(
 ↳ address(admin), address(nftFiHub));
171         airdropReceiverToClone = new AirdropReceiver(address(
 ↳ nftFiHub));
172         permittedNFTsAndTypeRegistry.setNftType('AirdropWrapper',
 ↳ address(airdropReceiverToClone));
173 
174         // SETTING CONTRACTS AND PERMISIONS
175         permittedNftContracts.push(address(airdropReceiverToClone)
 ↳ );
176         permittedNftTypes.push('AIRDROP_RECEIVER');
177         nftFiHub.setContract('AIRDROP_RECEIVER', address(
 ↳ airdropReceiverToClone));
178         nftFiHub.setContract('AIRDROP_FACTORY', address(
 ↳ airdropReceiverFactory));
179         nftFiHub.setContract('PERMITTED_NFTS', address(
 ↳ permittedNFTsAndTypeRegistry));
180         nftFiHub.setContract('PERMITTED_AIRDROPS', address(
 ↳ permittedAirdrop));
181 
182         // time update
183         timeNow = block.timestamp;
184         vm.stopPrank();
185     }
```

The following internal functions have been used for the complete execution of the tests:

Listing 4: PunkTest.t.sol

```
 1     function getBobbyOfferSignature() internal returns (LoanData.
 ↳ Signature memory) {
 2 
 3         // DECLARING OFFER
```

```
4           LoanData.Offer memory offer = declareOffer();
5
6           // GETTING CHAIN ID
7           uint256 id;
8           assembly {
9               id := chainid()
10          }
11
12          // PREPARING SIGNATURE STRUCT
13          LoanData.Signature memory signature = LoanData.Signature({
14              nonce: 0,
15              expiry: timeNow + 10 days,
16              signer: bobby,
17              signature: hex"1c"
18          });
19
20          // GETTING THE MESSAGE HASH
21          bytes32 message = keccak256(
22              abi.encodePacked(getEncodedOffer(offer),
↳ getEncodedSignature(signature), address(
↳ directLoanFixedOfferRedeploy), id)
23          );
24
25          // EIP712 STANDARD
26          bytes32 signedMessage = ECDSA.toEthSignedMessageHash(
↳ message);
27
28          // GETTING THE V, R, S OF THE SIGNED MESSAGE
29          (uint8 v, bytes32 r, bytes32 s) = vm.sign(0xB0BB1,
↳ signedMessage);
30          bytes memory v_bytes;
31          if(v == 27){v_bytes = hex"1b";} else {v_bytes = hex"1c";}
32          bytes memory signaturesf = bytes.concat(r, s, v_bytes);
33
34          // BOBBY SIGNES
35          LoanData.Signature memory signaturefi = LoanData.Signature
↳ ({
36              nonce: 0,
37              expiry: timeNow + 10 days,
38              signer: bobby,
39              signature: signaturesf
40          });
41
42          return signaturefi;
```

```
43      }
44
45      function declareOffer() internal returns (LoanData.Offer
↳ memory) {
46
47          // DECLARING OFFER
48          LoanData.Offer memory offer = LoanData.Offer({
49              loanPrincipalAmount: 10_000000000000000000,
50              maximumRepaymentAmount: 12_000000000000000000,
51              nftCollateralId: 0,
52              nftCollateralContract: address(punkContract),
53              loanDuration: 10 days,
54              loanAdminFeeInBasisPoints: 500,
55              loanERC20Denomination: address(token),
56              referrer: zeroo
57          });
58
59          return offer;
60      }
61
62      function getBalances(uint32 loanId) internal {
63          (uint256 loanPrincipalAmount, uint256
↳ maximumRepaymentAmount, uint256 nftCollateralId, address
↳ loanERC20Denomination, uint32 loanDuration, uint32
↳ loanInterestRateForDuration, uint16 loanAdminFeeInBasisPoints,
↳ address nftCollateralWrapper, uint64 loanStartTime, address
↳ nftCollateralContract, address borrower) =
↳ directLoanFixedOfferRedeploy.loanIdToLoan(loanId);
64          liquidated = directLoanFixedOfferRedeploy.
↳ loanRepaidOrLiquidated(loanId);
65
66          console.log("****** BALANCES ******");
67          console.log("Balance Of Admin        ---> " ,token.
↳ balanceOf(admin));
68          console.log("Balance Of Alice        ---> " ,token.
↳ balanceOf(alice));
69          console.log("Balance Of Bobby        ---> " ,token.
↳ balanceOf(bobby));
70          console.log("Balance Of Carla        ---> " ,token.
↳ balanceOf(carla));
71          console.log("Owner of the NFT        ---> " ,punkContract.
↳ punkIndexToAddress(0));
72          console.log(" ");
73          console.log("****** LOAN DATA ******");
```

```
74          console.log("loanPrincipalAmount        ---> " ,
↳ loanPrincipalAmount);
75          console.log("maximumRepaymentAmount     ---> " ,
↳ maximumRepaymentAmount);
76          console.log("nftCollateralId            ---> " ,
↳ nftCollateralId);
77          console.log("loanERC20Denomination      ---> " ,
↳ loanERC20Denomination);
78          console.log("loanDuration               ---> " ,
↳ loanDuration);
79          console.log("interestRateForDuration    ---> " ,
↳ loanInterestRateForDuration);
80          console.log("loanAdminFeeInBasisPoints  ---> " ,
↳ loanAdminFeeInBasisPoints);
81          console.log("nftCollateralWrapper       ---> " ,
↳ nftCollateralWrapper);
82          console.log("loanStartTime              ---> " ,
↳ loanStartTime);
83          console.log("nftCollateralContract      ---> " ,
↳ nftCollateralContract);
84          console.log("borrower                   ---> " ,borrower);
85          console.log("LOAN LIQUIDATED / REPAYED  ---> " ,liquidated
↳ );
86          console.log(" ");
87      }
88
89      function getEncodedSignature(LoanData.Signature memory
↳ _signature) internal pure returns (bytes memory) {
90          return abi.encodePacked(_signature.signer, _signature.
↳ nonce, _signature.expiry);
91      }
92
93      function getEncodedOffer(LoanData.Offer memory _offer)
↳ internal pure returns (bytes memory) {
94          return
95              abi.encodePacked(
96                  _offer.loanERC20Denomination,
97                  _offer.loanPrincipalAmount,
98                  _offer.maximumRepaymentAmount,
99                  _offer.nftCollateralContract,
100                 _offer.nftCollateralId,
101                 _offer.referrer,
102                 _offer.loanDuration,
103                 _offer.loanAdminFeeInBasisPoints
```

```
104                 );
105         }
106 }
107
```

TEST 1:

Script

Listing 5: NFTFi.t.sol

```
174     function test_1() public {
175
176         // INITIAL STATE LOGS
177         console.log("****** STATE 0 (ENV) *******");
178         console.log(" ");
179         getBalances();
180
181         vm.prank(alice);
182         punkContract.offerPunkForSaleToAddress(0, 0, address(
 ↳ directLoanFixedOfferRedeploy));
183
184         // TOKEN APPROVAL
185         vm.prank(bobby);
186         token.approve(address(directLoanFixedOfferRedeploy), 10
 ↳ _000000000000000000);
187
188         // PREPARING SINGNATURE
189         LoanData.Offer memory offer = declareOffer();
190         LoanData.Signature memory signaturefi =
 ↳ getBobbyOfferSignature();
191
192         // ALICE ACCEPTS BOBBY'S OFFER
193         LoanData.BorrowerSettings memory borrowerSettings;
194         vm.prank(alice);
195         directLoanFixedOfferRedeploy.acceptOffer(offer,
 ↳ signaturefi, borrowerSettings);
196
197         // LOGS
198         console.log("****** STATE 1 *******");
199         console.log("TX: ALICE ---> ACCEPT BOBBY'S OFFER");
200         console.log(" ");
201         getBalances(1);
202
203         // 5 DAYS LATER
204         console.log("---------------");
205         console.log("5 DAYS LATER...");
206         console.log("---------------");
207         console.log(" ");
208         vm.warp(5 days);
209
```

```
210         // ALICE PAY THE MONEY
211         vm.prank(alice);
212         token.approve(address(directLoanFixedOfferRedeploy), 12
 ↳ _000000000000000000);
213         vm.prank(alice);
214         directLoanFixedOfferRedeploy.payBackLoan(1);
215 
216         // LOGS
217         console.log("****** STATE 2 *******");
218         console.log("TX: ALICE ---> PAY BACK LOAN");
219         console.log(" ");
220         getBalances(1);
221     }
```

MANUAL TESTING

Output

```
[PASS] test_1() (gas: 643950)
Logs:
  ********************************************************************************
  **********  TEST 1: General repayment cycle (Borrower - Escrow - Borrower) **********
  ********************************************************************************

  ****** STATE 0 (ENV) *******

  ****** BALANCES ******
  Balance Of Admin        --->  99890000000000000000000
  Balance Of Alice        --->  5000000000000000000
  Balance Of Bobby        --->  10000000000000000000000
  Balance Of Carla        --->  5000000000000000000
  Owner of the NFT        --->  0x61A1D7fD8C9bbd82932D99DFD47bD2581C23b08c

  ****** STATE 1 *******
  TX: ALICE ---> ACCEPT BOBBY'S OFFER

  ****** BALANCES ******
  Balance Of Admin        --->  99890000000000000000000
  Balance Of Alice        --->  6000000000000000000
  Balance Of Bobby        --->  99000000000000000000
  Balance Of Carla        --->  5000000000000000000
  Owner of the NFT        --->  0x5f67D76CDa93c064D8f8386fdd5Bd0caE49957A3

  ****** LOAN DATA ******
  loanPrincipalAmount        --->  1000000000000000000
  maximumRepaymentAmount     --->  1200000000000000000
  nftCollateralId            --->  0
  loanERC20Denomination      --->  0xe6CCdc781deb32A0463fb1d81D0fF86d2E184941
  loanDuration               --->  864000
  interestRateForDuration    --->  0
  loanAdminFeeInBasisPoints  --->  500
  nftCollateralWrapper       --->  0x24a683C0D64AC3B75d1e1aE6D9EC4d5bcA11E387
  loanStartTime              --->  1
  nftCollateralContract      --->  0x7D28001937fe8e131F76DaE9E9947adEDbD0abdE
  borrower                   --->  0x61A1D7fD8C9bbd82932D99DFD47bD2581C23b08c
  LOAN LIQUIDATED / REPAYED  --->  false

  ---------------
  5 DAYS LATER...
  ---------------

  ****** STATE 2 *******
  TX: ALICE ---> PAY BACK LOAN

  ****** BALANCES ******
  Balance Of Admin        --->  99890010000000000000000
  Balance Of Alice        --->  4800000000000000000
  Balance Of Bobby        --->  10019000000000000000000
  Balance Of Carla        --->  5000000000000000000
  Owner of the NFT        --->  0x61A1D7fD8C9bbd82932D99DFD47bD2581C23b08c

  ****** 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 1.1:

Script

Listing 6

```
function test_13() public {

        // INITIAL STATE LOGS
        console.log("****** STATE 0 (ENV) *******");
        console.log(" ");
        getBalances();

        vm.prank(alice);
        punkContract.offerPunkForSaleToAddress(0, 0, address(
↳ directLoanFixedOfferRedeploy));

        // TOKEN APPROVAL
        vm.prank(bobby);
        token.approve(address(directLoanFixedOfferRedeploy), 10
↳ _000000000000000000);

        // PREPARING SINGNATURE
        LoanData.Offer memory offer = declareOffer();
        LoanData.Signature memory signaturefi =
↳ getBobbyOfferSignature();

        // ALICE ACCEPTS BOBBY'S OFFER
        LoanData.BorrowerSettings memory borrowerSettings;
        vm.prank(alice);
        directLoanFixedOfferRedeploy.acceptOffer(offer,
↳ signaturefi, borrowerSettings);

        // LOGS
        console.log("****** STATE 1 *******");
        console.log("TX: ALICE ---> ACCEPT BOBBY'S OFFER");
        console.log(" ");
        getBalances(1);

        // 5 DAYS LATER
        console.log("---------------");
        console.log("5 DAYS LATER...");
        console.log("---------------");
        vm.warp(5 days);

        // ALICE PAY THE MONEY
```

```
37          vm.prank(carla);
38          token.approve(address(directLoanFixedOfferRedeploy), 12
↳ _000000000000000000);
39          vm.prank(carla);
40          directLoanFixedOfferRedeploy.payBackLoan(1);
41
42          // LOGS
43          console.log("****** STATE 2 *******");
44          console.log("TX: CARLA ---> PAY BACK LOAN");
45          console.log(" ");
46          getBalances(1);
47
48          punkContract.punkIndexToAddress(0);
49      }
```

Output

```
[PASS] test_13() (gas: 646696)
Logs:
  ********************************************************************************************************************
  **********  TEST 1.3: Attack vector: External user repays loan to steal the NFT (revert expected)  **********
  ********************************************************************************************************************

  ****** STATE 0 (ENV) *******

  ****** BALANCES ******
  Balance Of Admin        --->  99890000000000000000000
  Balance Of Alice        --->  5000000000000000000
  Balance Of Bobby        --->  10000000000000000000000
  Balance Of Carla        --->  5000000000000000000
  Owner of the NFT        --->  0x61A1D7fD8C9bbd82932D99DFD47bD2581C23b08c

  ****** STATE 1 *******
  TX: ALICE ---> ACCEPT BOBBY'S OFFER

  ****** BALANCES ******
  Balance Of Admin        --->  99890000000000000000000
  Balance Of Alice        --->  6000000000000000000
  Balance Of Bobby        --->  99000000000000000000
  Balance Of Carla        --->  5000000000000000000
  Owner of the NFT        --->  0x5f67D76CDa93c064D8f8386fdd5Bd0caE49957A3

  ****** LOAN DATA ******
  loanPrincipalAmount        --->  1000000000000000000
  maximumRepaymentAmount     --->  1200000000000000000
  nftCollateralId            --->  0
  loanERC20Denomination      --->  0xe6CCdc781deb32A0463fb1d81D0fF86d2E184941
  loanDuration               --->  864000
  interestRateForDuration    --->  0
  loanAdminFeeInBasisPoints  --->  500
  nftCollateralWrapper       --->  0x24a683C0D64AC3B75d1e1aE6D9EC4d5bcA11E387
  loanStartTime              --->  1
  nftCollateralContract      --->  0x7D28001937fe8e131F76DaE9E9947adEDbD0abdE
  borrower                   --->  0x61A1D7fD8C9bbd82932D99DFD47bD2581C23b08c
  LOAN LIQUIDATED / REPAYED  --->  false

  ---------------
  5 DAYS LATER...
  ---------------
  ****** STATE 2 *******
  TX: CARLA ---> PAY BACK LOAN

  ****** BALANCES ******
  Balance Of Admin        --->  99890010000000000000000
  Balance Of Alice        --->  6000000000000000000
  Balance Of Bobby        --->  100190000000000000000
  Balance Of Carla        --->  3800000000000000000
  Owner of the NFT        --->  0x61A1D7fD8C9bbd82932D99DFD47bD2581C23b08c

  ****** 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 7

```
function test_2() public {

        // INITIAL STATE LOGS
        console.log("****** STATE 0 (ENV) *******");
        console.log(" ");
        getBalances();

        vm.prank(alice);
        punkContract.offerPunkForSaleToAddress(0, 0, address(
↳ directLoanFixedOfferRedeploy));

        // TOKEN APPROVAL
        vm.prank(bobby);
        token.approve(address(directLoanFixedOfferRedeploy), 10
↳ _000000000000000000);

        // PREPARING SINGNATURE
        LoanData.Offer memory offer = declareOffer();
        LoanData.Signature memory signaturefi =
↳ getBobbyOfferSignature();

        // ALICE ACCEPTS BOBBY'S OFFER
        LoanData.BorrowerSettings memory borrowerSettings;
        vm.prank(alice);
        directLoanFixedOfferRedeploy.acceptOffer(offer,
↳ signaturefi, borrowerSettings);

        // LOGS
        console.log("****** STATE 1 *******");
        console.log("TX: ALICE ---> ACCEPT BOBBY'S OFFER");
        console.log(" ");
        getBalances(1);

        // 15 DAYS LATER
        console.log("----------------");
        console.log("15 DAYS LATER...");
        console.log("----------------");
        vm.warp(15 days);

        // ALICE PAY THE MONEY
```

```
37          vm.prank(alice);
38          token.approve(address(directLoanFixedOfferRedeploy), 12
↳ _000000000000000000);
39
40          vm.prank(alice);
41          vm.expectRevert("Loan is expired");
42          directLoanFixedOfferRedeploy.payBackLoan(1);
43
44          // LOGS
45          console.log("****** STATE 2 *******");
46          console.log("TX: ALICE ---> TRIES TO REPAY");
47          console.log(" ");
48          getBalances(1);
49      }
```

MANUAL TESTING

Output

```
[PASS] test_2() (gas: 642317)
Logs:
  ********************************************************************************
  **********  TEST 2: General repayment cycle out of time (revert expected)  **********
  ********************************************************************************

  ****** STATE 0 (ENV) *******

  ****** BALANCES ******
  Balance Of Admin        --->  998900000000000000000000
  Balance Of Alice        --->  5000000000000000000
  Balance Of Bobby        --->  10000000000000000000
  Balance Of Carla        --->  5000000000000000000
  Owner of the NFT        --->  0x61A1D7fD8C9bbd82932D99DFD47bD2581C23b08c

  ****** STATE 1 *******
  TX: ALICE ---> ACCEPT BOBBY'S OFFER

  ****** BALANCES ******
  Balance Of Admin        --->  998900000000000000000000
  Balance Of Alice        --->  6000000000000000000
  Balance Of Bobby        --->  9900000000000000000
  Balance Of Carla        --->  5000000000000000000
  Owner of the NFT        --->  0x5f67D76CDa93c064D8f8386fdd5Bd0caE49957A3

  ****** LOAN DATA ******
  loanPrincipalAmount        --->  1000000000000000000
  maximumRepaymentAmount     --->  1200000000000000000
  nftCollateralId            --->  0
  loanERC20Denomination      --->  0xe6CCdc781deb32A0463fb1d81D0fF86d2E184941
  loanDuration               --->  864000
  interestRateForDuration    --->  0
  loanAdminFeeInBasisPoints  --->  500
  nftCollateralWrapper       --->  0x24a683C0D64AC3B75d1e1aE6D9EC4d5bcA11E387
  loanStartTime              --->  1
  nftCollateralContract      --->  0x7D28001937fe8e131F76DaE9E9947adEDbD0abdE
  borrower                   --->  0x61A1D7fD8C9bbd82932D99DFD47bD2581C23b08c
  LOAN LIQUIDATED / REPAYED  --->  false

  ----------------
  15 DAYS LATER...
  ----------------
  ****** STATE 2 *******
  TX: ALICE ---> TRIES TO REPAY

  ****** BALANCES ******
  Balance Of Admin        --->  998900000000000000000000
  Balance Of Alice        --->  6000000000000000000
  Balance Of Bobby        --->  9900000000000000000
  Balance Of Carla        --->  5000000000000000000
  Owner of the NFT        --->  0x5f67D76CDa93c064D8f8386fdd5Bd0caE49957A3

  ****** LOAN DATA ******
  loanPrincipalAmount        --->  1000000000000000000
  maximumRepaymentAmount     --->  1200000000000000000
  nftCollateralId            --->  0
  loanERC20Denomination      --->  0xe6CCdc781deb32A0463fb1d81D0fF86d2E184941
  loanDuration               --->  864000
  interestRateForDuration    --->  0
  loanAdminFeeInBasisPoints  --->  500
  nftCollateralWrapper       --->  0x24a683C0D64AC3B75d1e1aE6D9EC4d5bcA11E387
  loanStartTime              --->  1
  nftCollateralContract      --->  0x7D28001937fe8e131F76DaE9E9947adEDbD0abdE
  borrower                   --->  0x61A1D7fD8C9bbd82932D99DFD47bD2581C23b08c
  LOAN LIQUIDATED / REPAYED  --->  false
```

TEST 3:

Script

Listing 8

```
function test_3() public {

        // INITIAL STATE LOGS
        console.log("****** STATE 0 (ENV) *******");
        console.log(" ");
        getBalances();

        vm.prank(alice);
        punkContract.offerPunkForSaleToAddress(0, 0, address(
↳ directLoanFixedOfferRedeploy));

        // TOKEN APPROVAL
        vm.prank(bobby);
        token.approve(address(directLoanFixedOfferRedeploy), 10
↳ _000000000000000000);

        // PREPARING SINGNATURE
        LoanData.Offer memory offer = declareOffer();
        LoanData.Signature memory signaturefi =
↳ getBobbyOfferSignature();

        // ALICE ACCEPTS BOBBY'S OFFER
        LoanData.BorrowerSettings memory borrowerSettings;
        vm.prank(alice);
        directLoanFixedOfferRedeploy.acceptOffer(offer,
↳ signaturefi, borrowerSettings);

        // LOGS
        console.log("****** STATE 1 *******");
        console.log("TX: ALICE ---> ACCEPT OFFER");
        console.log(" ");
        getBalances(1);

        // 11 DAYS LATER
        vm.warp(11 days);
        console.log("----------------");
        console.log("11 DAYS LATER...");
        console.log("----------------");

        // BOBBY LIQUIDATES THE LOAN
```

```
        vm.prank(bobby);
        directLoanFixedOfferRedeploy.liquidateOverdueLoan(1);

        // LOGS
        console.log("****** STATE 2 *******");
        console.log("TX: BOBBY ---> LIQUIDATE LOAN");
        console.log(" ");
        getBalances(1);
    }
```

Output

```
[PASS] test_3() (gas: 619856)
Logs:
  ********************************************************************************
  **********  TEST 3: General liquidation cycle (Borrower - Escrow - Lender)  **********
  ********************************************************************************

  ****** STATE 0 (ENV) *******

  ****** BALANCES ******
  Balance Of Admin        --->  99890000000000000000000
  Balance Of Alice        --->  5000000000000000000
  Balance Of Bobby        --->  10000000000000000000000
  Balance Of Carla        --->  5000000000000000000
  Owner of the NFT        --->  0x61A1D7fD8C9bbd82932D99DFD47bD2581C23b08c

  ****** STATE 1 *******
  TX: ALICE ---> ACCEPT OFFER

  ****** BALANCES ******
  Balance Of Admin        --->  99890000000000000000000
  Balance Of Alice        --->  6000000000000000000
  Balance Of Bobby        --->  99000000000000000000000
  Balance Of Carla        --->  5000000000000000000
  Owner of the NFT        --->  0x5f67D76CDa93c064D8f8386fdd5Bd0caE49957A3

  ****** LOAN DATA ******
  loanPrincipalAmount        --->  1000000000000000000
  maximumRepaymentAmount     --->  1200000000000000000
  nftCollateralId            --->  0
  loanERC20Denomination      --->  0xe6CCdc781deb32A0463fb1d81D0fF86d2E184941
  loanDuration               --->  864000
  interestRateForDuration    --->  0
  loanAdminFeeInBasisPoints  --->  500
  nftCollateralWrapper       --->  0x24a683C0D64AC3B75d1e1aE6D9EC4d5bcA11E387
  loanStartTime              --->  1
  nftCollateralContract      --->  0x7D28001937fe8e131F76DaE9E9947adEDbD0abdE
  borrower                   --->  0x61A1D7fD8C9bbd82932D99DFD47bD2581C23b08c
  LOAN LIQUIDATED / REPAYED  --->  false

  ----------------
  11 DAYS LATER...
  ----------------
  ****** STATE 2 *******
  TX: BOBBY ---> LIQUIDATE LOAN

  ****** BALANCES ******
  Balance Of Admin        --->  99890000000000000000000
  Balance Of Alice        --->  6000000000000000000
  Balance Of Bobby        --->  99000000000000000000000
  Balance Of Carla        --->  5000000000000000000
  Owner of the NFT        --->  0x3CE907fF40299087175b849632c8e4979C3ebABF

  ****** 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 9

```
function test_4() public {

        // INITIAL STATE LOGS
        console.log("****** STATE 0 (ENV) *******");
        console.log(" ");
        getBalances();

        vm.prank(alice);
        punkContract.offerPunkForSaleToAddress(0, 0, address(
↳ directLoanFixedOfferRedeploy));

        // TOKEN APPROVAL
        vm.prank(bobby);
        token.approve(address(directLoanFixedOfferRedeploy), 10
↳ _000000000000000000);

        // PREPARING SINGNATURE
        LoanData.Offer memory offer = declareOffer();
        LoanData.Signature memory signaturefi =
↳ getBobbyOfferSignature();

        // ALICE ACCEPTS BOBBY'S OFFER
        LoanData.BorrowerSettings memory borrowerSettings;
        vm.prank(alice);
        directLoanFixedOfferRedeploy.acceptOffer(offer,
↳ signaturefi, borrowerSettings);

        // LOGS
        console.log("****** STATE 1 *******");
        console.log("TX: ALICE ---> ACCEPT OFFER");
        console.log(" ");
        getBalances(1);

        // 3 DAYS LATER
        vm.warp(3 days);
        console.log("---------------");
        console.log("3 DAYS LATER...");
        console.log("---------------");

        // BOBBY LIQUIDATES THE LOAN
```

```
37          vm.prank(bobby);
38          vm.expectRevert();
39          directLoanFixedOfferRedeploy.liquidateOverdueLoan(1);
40
41          // LOGS
42          console.log("****** STATE 1 *******");
43          console.log("TX: BOBBY ---> TRIES TO LIQUIDATE LOAN TOO
↳ EARLY");
44          console.log(" ");
45          getBalances(1);
46      }
```

Output

```
[PASS] test_4() (gas: 619160)
Logs:
  ********************************************************************************************
  **********  TEST 4: General liquidation cycle before agreement (revert expected)  **********
  ********************************************************************************************

  ****** STATE 0 (ENV) *******

  ****** BALANCES ******
  Balance Of Admin        --->  99890000000000000000000
  Balance Of Alice        --->  5000000000000000000
  Balance Of Bobby        --->  10000000000000000000000
  Balance Of Carla        --->  5000000000000000000
  Owner of the NFT        --->  0x61A1D7fD8C9bbd82932D99DFD47bD2581C23b08c

  ****** STATE 1 *******
  TX: ALICE ---> ACCEPT OFFER

  ****** BALANCES ******
  Balance Of Admin        --->  99890000000000000000000
  Balance Of Alice        --->  6000000000000000000
  Balance Of Bobby        --->  99000000000000000000000
  Balance Of Carla        --->  5000000000000000000
  Owner of the NFT        --->  0x5f67D76CDa93c064D8f8386fdd5Bd0caE49957A3

  ****** LOAN DATA ******
  loanPrincipalAmount        --->  1000000000000000000
  maximumRepaymentAmount     --->  1200000000000000000
  nftCollateralId            --->  0
  loanERC20Denomination      --->  0xe6CCdc781deb32A0463fb1d81D0fF86d2E184941
  loanDuration               --->  864000
  interestRateForDuration    --->  0
  loanAdminFeeInBasisPoints  --->  500
  nftCollateralWrapper       --->  0x24a683C0D64AC3B75d1e1aE6D9EC4d5bcA11E387
  loanStartTime              --->  1
  nftCollateralContract      --->  0x7D28001937fe8e131F76DaE9E9947adEDbD0abdE
  borrower                   --->  0x61A1D7fD8C9bbd82932D99DFD47bD2581C23b08c
  LOAN LIQUIDATED / REPAYED  --->  false

  ---------------
  3 DAYS LATER...
  ---------------
  ****** STATE 1 *******
  TX: BOBBY ---> TRIES TO LIQUIDATE LOAN TOO EARLY

  ****** BALANCES ******
  Balance Of Admin        --->  99890000000000000000000
  Balance Of Alice        --->  6000000000000000000
  Balance Of Bobby        --->  99000000000000000000000
  Balance Of Carla        --->  5000000000000000000
  Owner of the NFT        --->  0x5f67D76CDa93c064D8f8386fdd5Bd0caE49957A3

  ****** LOAN DATA ******
  loanPrincipalAmount        --->  1000000000000000000
  maximumRepaymentAmount     --->  1200000000000000000
  nftCollateralId            --->  0
  loanERC20Denomination      --->  0xe6CCdc781deb32A0463fb1d81D0fF86d2E184941
  loanDuration               --->  864000
  interestRateForDuration    --->  0
  loanAdminFeeInBasisPoints  --->  500
  nftCollateralWrapper       --->  0x24a683C0D64AC3B75d1e1aE6D9EC4d5bcA11E387
  loanStartTime              --->  1
  nftCollateralContract      --->  0x7D28001937fe8e131F76DaE9E9947adEDbD0abdE
  borrower                   --->  0x61A1D7fD8C9bbd82932D99DFD47bD2581C23b08c
  LOAN LIQUIDATED / REPAYED  --->  false
```

TEST 5:

Script

Listing 10

```
function test_5() public {

        // INITIAL STATE LOGS
        console.log("****** STATE 0 (ENV) *******");
        console.log(" ");
        getBalances();

        vm.prank(alice);
        punkContract.offerPunkForSaleToAddress(0, 0, address(
↳ directLoanFixedOfferRedeploy));

        // TOKEN APPROVAL
        vm.prank(bobby);
        token.approve(address(directLoanFixedOfferRedeploy), 10
↳ _000000000000000000);

        // PREPARING SINGNATURE
        LoanData.Offer memory offer = declareOffer();
        LoanData.Signature memory signaturefi =
↳ getBobbyOfferSignature();

        // ALICE ACCEPTS BOBBY'S OFFER
        LoanData.BorrowerSettings memory borrowerSettings;
        vm.prank(alice);
        directLoanFixedOfferRedeploy.acceptOffer(offer,
↳ signaturefi, borrowerSettings);

        // LOGS
        console.log("****** STATE 1 *******");
        console.log("TX: ALICE ---> ACCEPT OFFER");
        console.log(" ");
        getBalances(1);

        // 12 DAYS LATER
        console.log("----------------");
        console.log("12 DAYS LATER...");
        console.log("----------------");
        vm.warp(12 days);

        uint256 id2;
```

```
37          assembly {
38              id2 := chainid()
39          }
40
41          uint256 _expiry = block.timestamp + 20 days;
42
43          // GETTING THE MESSAGE HASH
44          bytes32 message = keccak256(
45              abi.encodePacked(
46                  uint256(1),
47                  uint32(30 days),
48                  uint256(20_000000000000000000),
49                  uint256(5_000000000000000000),
50                  abi.encodePacked(bobby, uint256(1), _expiry),
51                  address(directLoanFixedOfferRedeploy),
52                  id2
53              )
54          );
55
56          // EIP712 STANDARD
57          bytes32 signedMessage = ECDSA.toEthSignedMessageHash(
↳ message);
58
59          // GETTING THE V, R, S OF THE SIGNED MESSAGE
60          (uint8 v, bytes32 r, bytes32 s) = vm.sign(0xB0BB1,
↳ signedMessage);
61          bytes memory v_bytes;
62          if(v == 27){v_bytes = hex"1b";} else {v_bytes = hex"1c";}
63          bytes memory _lenderSignature = bytes.concat(r, s, v_bytes
↳ );
64
65          // ALICE WANTS TO RENEGOTIATE
66          vm.prank(alice);
67          token.approve(address(directLoanFixedOfferRedeploy), 5
↳ _000000000000000000);
68          vm.prank(alice);
69          directLoanFixedOfferRedeploy.renegotiateLoan(1, 30 days,
↳ 20_000000000000000000, 5_000000000000000000, 1, _expiry,
↳ _lenderSignature);
70
71          // LOGS
72          console.log("****** STATE 2 *******");
73          console.log("TX: ALICE ---> RENEGOTIATE LOAN");
74          console.log(" ");
```

```
75          getBalances(1);
76
77          // 10 DAYS LATER
78          console.log("----------------");
79          console.log("10 DAYS LATER...");
80          console.log("----------------");
81          vm.warp(10 days);
82
83          // ALICE PAY THE MONEY
84          vm.prank(alice);
85          token.approve(address(directLoanFixedOfferRedeploy), 20
↳ _000000000000000000);
86          vm.prank(alice);
87          token.approve(address(directLoanFixedOfferRedeploy), 20
↳ _000000000000000000);
88          vm.prank(alice);
89          directLoanFixedOfferRedeploy.payBackLoan(1);
90
91          // LOGS
92          console.log("****** STATE 3 *******");
93          console.log("TX: ALICE ---> PAY BACK LOAN");
94          console.log(" ");
95          getBalances(1);
96      }
```

```
[PASS] test_5() (gas: 751352)
Logs:
  ********************************************************************************
  **********  TEST 5: General repayment cycle out of time with renegotiation  **********
  ********************************************************************************

  ****** STATE 0 (ENV) *******

  ****** BALANCES ******
  Balance Of Admin        --->  998900000000000000000000
  Balance Of Alice        --->  5000000000000000000
  Balance Of Bobby        --->  10000000000000000000
  Balance Of Carla        --->  5000000000000000000
  Owner of the NFT        --->  0x61A1D7fD8C9bbd82932D99DFD47bD2581C23b08c

  ****** STATE 1 *******
  TX: ALICE ---> ACCEPT OFFER

  ****** BALANCES ******
  Balance Of Admin        --->  998900000000000000000000
  Balance Of Alice        --->  6000000000000000000
  Balance Of Bobby        --->  9900000000000000000000
  Balance Of Carla        --->  5000000000000000000
  Owner of the NFT        --->  0x5f67D76CDa93c064D8f8386fdd5Bd0caE49957A3

  ****** LOAN DATA ******
  loanPrincipalAmount        --->  1000000000000000000
  maximumRepaymentAmount     --->  1200000000000000000
  nftCollateralId            --->  0
  loanERC20Denomination      --->  0xe6CCdc781deb32A0463fb1d81D0fF86d2E184941
  loanDuration               --->  864000
  interestRateForDuration    --->  0
  loanAdminFeeInBasisPoints  --->  500
  nftCollateralWrapper       --->  0x24a683C0D64AC3B75d1e1aE6D9EC4d5bcA11E387
  loanStartTime              --->  1
  nftCollateralContract      --->  0x7D28001937fe8e131F76DaE9E9947adEDbD0abdE
  borrower                   --->  0x61A1D7fD8C9bbd82932D99DFD47bD2581C23b08c
  LOAN LIQUIDATED / REPAYED  --->  false

  ----------------
  12 DAYS LATER...
  ----------------
  ****** STATE 2 *******
  TX: ALICE ---> RENEGOTIATE LOAN

  ****** BALANCES ******
  Balance Of Admin        --->  998900250000000000000000
  Balance Of Alice        --->  5500000000000000000
  Balance Of Bobby        --->  9947500000000000000000
  Balance Of Carla        --->  5000000000000000000
  Owner of the NFT        --->  0x5f67D76CDa93c064D8f8386fdd5Bd0caE49957A3

  ****** LOAN DATA ******
  loanPrincipalAmount        --->  1000000000000000000
  maximumRepaymentAmount     --->  2000000000000000000
  nftCollateralId            --->  0
  loanERC20Denomination      --->  0xe6CCdc781deb32A0463fb1d81D0fF86d2E184941
  loanDuration               --->  2592000
  interestRateForDuration    --->  0
  loanAdminFeeInBasisPoints  --->  500
  nftCollateralWrapper       --->  0x24a683C0D64AC3B75d1e1aE6D9EC4d5bcA11E387
  loanStartTime              --->  1
  nftCollateralContract      --->  0x7D28001937fe8e131F76DaE9E9947adEDbD0abdE
  borrower                   --->  0x61A1D7fD8C9bbd82932D99DFD47bD2581C23b08c
  LOAN LIQUIDATED / REPAYED  --->  false
```

Output

```
  ----------------
  10 DAYS LATER...
  ----------------
  ****** STATE 3 *******
  TX: ALICE ---> PAY BACK LOAN

  ****** BALANCES ******
  Balance Of Admin        --->  998900750000000000000000
  Balance Of Alice        --->  3500000000000000000
  Balance Of Bobby        --->  10142500000000000000000
  Balance Of Carla        --->  5000000000000000000
  Owner of the NFT        --->  0x61A1D7fD8C9bbd82932D99DFD47bD2581C23b08c

  ****** 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 11

```
function test_6_CannotRenegotiateHisOwnLoan() public {

        // INITIAL STATE LOGS
        console.log("****** STATE 0 (ENV) *******");
        console.log(" ");
        getBalances();

        vm.prank(alice);
        punkContract.offerPunkForSaleToAddress(0, 0, address(
↳ directLoanFixedOfferRedeploy));

        // TOKEN APPROVAL
        vm.prank(bobby);
        token.approve(address(directLoanFixedOfferRedeploy), 10
↳ _000000000000000000);

        // PREPARING SINGNATURE
        LoanData.Offer memory offer = declareOffer();
        LoanData.Signature memory signaturefi =
↳ getBobbyOfferSignature();

        // ALICE ACCEPTS BOBBY'S OFFER
        LoanData.BorrowerSettings memory borrowerSettings;
        vm.prank(alice);
        directLoanFixedOfferRedeploy.acceptOffer(offer,
↳ signaturefi, borrowerSettings);

        // LOGS
        console.log("****** STATE 1 *******");
        console.log("TX: ALICE ---> ACCEPT OFFER");
        console.log(" ");
        getBalances(1);

        // 1 DAY LATER
        console.log("---------------");
        console.log("1 DAYS LATER...");
        console.log("---------------");
        vm.warp(1 days);

        // MSG HASH PREPARATION
```

```
37          uint256 id2;
38          assembly {
39              id2 := chainid()
40          }
41          uint256 _expiry = block.timestamp + 10; // + 10 seconds
42
43          // GETTING THE MESSAGE HASH
44          bytes32 message = keccak256(
45              abi.encodePacked(
46                  uint32(1),
47                  uint32(0),
48                  uint256(1_000000000000000000),
49                  uint256(0),
50                  abi.encodePacked(bobby, uint256(1), _expiry),
51                  address(directLoanFixedOfferRedeploy),
52                  id2
53              )
54          );
55
56          // EIP712 STANDARD
57          bytes32 signedMessage = ECDSA.toEthSignedMessageHash(
↳ message);
58
59          // GETTING THE V, R, S OF THE SIGNED MESSAGE
60          (uint8 v, bytes32 r, bytes32 s) = vm.sign(0xB0BB1,
↳ signedMessage);
61          bytes memory v_bytes;
62          if(v == 27){v_bytes = hex"1b";} else {v_bytes = hex"1c";}
63          bytes memory _lenderSignature = bytes.concat(r, s, v_bytes
↳ );
64
65          // ALICE WANTS TO RENEGOTIATE
66          vm.prank(bobby);
67          vm.expectRevert("Only borrower can initiate");
68          directLoanFixedOfferRedeploy.renegotiateLoan(1, 0, 1
↳ _000000000000000000, 0, 1, _expiry, _lenderSignature);
69
70          // BOBBY TRIES TO LIQUIDATE THE LOAN
71          vm.prank(bobby);
72          vm.expectRevert();
73          directLoanFixedOfferRedeploy.liquidateOverdueLoan(1);
74
75          // LOGS
76          console.log("****** STATE 2 *******");
```

```
77          console.log("TX: BOBBY ---> TRIES TO RENEGOTIATE HIS OWN
↳ LOAN AND REDUCE TIME TO 0 (revert)");
78          console.log("TX: BOBBY ---> LIQUIDATE LOAN (revert)");
79          console.log(" ");
80          getBalances(1);
81      }
```

Output

```
[PASS] test_6_CannotRenegotiateHisOwnLoan() (gas: 640155)
Logs:
  ********************************************************************************************************************************
  **********  TEST 6: Attack vector: Borrower renegotiate his own loan to steal the Punk (revert expected)  **********
  ********************************************************************************************************************************

  ****** STATE 0 (ENV) *******

  ****** BALANCES ******
  Balance Of Admin        --->  99890000000000000000000
  Balance Of Alice        --->  5000000000000000000
  Balance Of Bobby        --->  10000000000000000000000
  Balance Of Carla        --->  5000000000000000000
  Owner of the NFT        --->  0x61A1D7fD8C9bbd82932D99DFD47bD2581C23b08c

  ****** STATE 1 *******
  TX: ALICE ---> ACCEPT OFFER

  ****** BALANCES ******
  Balance Of Admin        --->  99890000000000000000000
  Balance Of Alice        --->  6000000000000000000
  Balance Of Bobby        --->  9900000000000000000000
  Balance Of Carla        --->  5000000000000000000
  Owner of the NFT        --->  0x5f67D76CDa93c064D8f8386fdd5Bd0caE49957A3

  ****** LOAN DATA ******
  loanPrincipalAmount        --->  1000000000000000000
  maximumRepaymentAmount     --->  1200000000000000000
  nftCollateralId            --->  0
  loanERC20Denomination      --->  0xe6CCdc781deb32A0463fb1d81D0fF86d2E184941
  loanDuration               --->  864000
  interestRateForDuration    --->  0
  loanAdminFeeInBasisPoints  --->  500
  nftCollateralWrapper       --->  0x24a683C0D64AC3B75d1e1aE6D9EC4d5bcA11E387
  loanStartTime              --->  1
  nftCollateralContract      --->  0x7D28001937fe8e131F76DaE9E9947adEDbD0abdE
  borrower                   --->  0x61A1D7fD8C9bbd82932D99DFD47bD2581C23b08c
  LOAN LIQUIDATED / REPAYED  --->  false

  ---------------
  1 DAYS LATER...
  ---------------
  ****** STATE 2 *******
  TX: BOBBY ---> TRIES TO RENEGOTIATE HIS OWN LOAN AND REDUCE TIME TO 0 (revert)
  TX: BOBBY ---> LIQUIDATE LOAN (revert)

  ****** BALANCES ******
  Balance Of Admin        --->  99890000000000000000000
  Balance Of Alice        --->  6000000000000000000
  Balance Of Bobby        --->  9900000000000000000000
  Balance Of Carla        --->  5000000000000000000
  Owner of the NFT        --->  0x5f67D76CDa93c064D8f8386fdd5Bd0caE49957A3

  ****** LOAN DATA ******
  loanPrincipalAmount        --->  1000000000000000000
  maximumRepaymentAmount     --->  1200000000000000000
  nftCollateralId            --->  0
  loanERC20Denomination      --->  0xe6CCdc781deb32A0463fb1d81D0fF86d2E184941
  loanDuration               --->  864000
  interestRateForDuration    --->  0
  loanAdminFeeInBasisPoints  --->  500
  nftCollateralWrapper       --->  0x24a683C0D64AC3B75d1e1aE6D9EC4d5bcA11E387
  loanStartTime              --->  1
  nftCollateralContract      --->  0x7D28001937fe8e131F76DaE9E9947adEDbD0abdE
  borrower                   --->  0x61A1D7fD8C9bbd82932D99DFD47bD2581C23b08c
  LOAN LIQUIDATED / REPAYED  --->  false
```

TEST 7:

Script

Listing 12

```
function test_7_PromisoryNoteExchange() public {

        // INITIAL STATE LOGS
        console.log("****** STATE 0 (ENV) *******");
        console.log(" ");
        getBalances();

        vm.prank(alice);
        punkContract.offerPunkForSaleToAddress(0, 0, address(
↳ directLoanFixedOfferRedeploy));

        // TOKEN APPROVAL
        vm.prank(bobby);
        token.approve(address(directLoanFixedOfferRedeploy), 10
↳ _000000000000000000);

        // PREPARING SINGNATURE
        LoanData.Offer memory offer = declareOffer();
        LoanData.Signature memory signaturefi =
↳ getBobbyOfferSignature();

        // ALICE ACCEPTS BOBBY'S OFFER
        LoanData.BorrowerSettings memory borrowerSettings;
        vm.prank(alice);
        directLoanFixedOfferRedeploy.acceptOffer(offer,
↳ signaturefi, borrowerSettings);

        // LOGS
        console.log("****** STATE 1 *******");
        console.log("TX: ALICE ---> ACCEPT OFFER");
        console.log(" ");
        getBalances(1);

        // TRNSFER PROMISSORY_NOTE
        nftObligationReceipt.balanceOf(address(alice));
        nftPromissoryNote.ownerOf(5929418158485165765);
        vm.startPrank(bobby);
        nftPromissoryNote.approve(address(carla),
↳ 5929418158485165765);
```

```
35          nftPromissoryNote.transferFrom(address(bobby), address(
↳ carla), 5929418158485165765);
36          vm.stopPrank();
37
38          // 5 DAYS LATER
39          console.log("---------------");
40          console.log("5 DAYS LATER...");
41          console.log("---------------");
42          vm.warp(5 days);
43
44          // ALICE PAY THE MONEY
45          vm.prank(alice);
46          token.approve(address(directLoanFixedOfferRedeploy), 12
↳ _00000000000000000);
47          vm.prank(alice);
48          directLoanFixedOfferRedeploy.payBackLoan(1);
49
50          // LOGS
51          console.log("****** STATE 2 *******");
52          console.log("TX: BOBBY ---> TRANSFER PROMISSORY NOTE TO
↳ CARLA");
53          console.log("TX: ALICE ---> PAY BACK LOAN");
54          console.log(" ");
55          getBalances(1);
56      }
```

Output

```
[PASS] test_7_PromisoryNoteExchange() (gas: 694762)
Logs:
  ********************************************************************************
  **********  TEST 7: General repayment cycle with promissory note transfer  **********
  ********************************************************************************

  ****** STATE 0 (ENV) *******

  ****** BALANCES ******
  Balance Of Admin        --->  99890000000000000000000
  Balance Of Alice        --->  5000000000000000000
  Balance Of Bobby        --->  10000000000000000000000
  Balance Of Carla        --->  5000000000000000000
  Owner of the NFT        --->  0x61A1D7fD8C9bbd82932D99DFD47bD2581C23b08c

  ****** STATE 1 *******
  TX: ALICE ---> ACCEPT OFFER

  ****** BALANCES ******
  Balance Of Admin        --->  99890000000000000000000
  Balance Of Alice        --->  6000000000000000000
  Balance Of Bobby        --->  9900000000000000000000
  Balance Of Carla        --->  5000000000000000000
  Owner of the NFT        --->  0x5f67D76CDa93c064D8f8386fdd5Bd0caE49957A3

  ****** LOAN DATA ******
  loanPrincipalAmount        --->  1000000000000000000
  maximumRepaymentAmount     --->  1200000000000000000
  nftCollateralId            --->  0
  loanERC20Denomination      --->  0xe6CCdc781deb32A0463fb1d81D0fF86d2E184941
  loanDuration               --->  864000
  interestRateForDuration    --->  0
  loanAdminFeeInBasisPoints  --->  500
  nftCollateralWrapper       --->  0x24a683C0D64AC3B75d1e1aE6D9EC4d5bcA11E387
  loanStartTime              --->  1
  nftCollateralContract      --->  0x7D28001937fe8e131F76DaE9E9947adEDbD0abdE
  borrower                   --->  0x61A1D7fD8C9bbd82932D99DFD47bD2581C23b08c
  LOAN LIQUIDATED / REPAYED  --->  false

  ---------------
  5 DAYS LATER...
  ---------------
  ****** STATE 2 *******
  TX: BOBBY ---> TRANSFER PROMISSORY NOTE TO CARLA
  TX: ALICE ---> PAY BACK LOAN

  ****** BALANCES ******
  Balance Of Admin        --->  99890010000000000000000
  Balance Of Alice        --->  4800000000000000000
  Balance Of Bobby        --->  9900000000000000000000
  Balance Of Carla        --->  6190000000000000000
  Owner of the NFT        --->  0x61A1D7fD8C9bbd82932D99DFD47bD2581C23b08c

  ****** 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 8:

Script

Listing 13

```
function test_8_ObligationReceiptExchange() public {

        // INITIAL STATE LOGS
        console.log("****** STATE 0 (ENV) *******");
        console.log(" ");
        getBalances();

        vm.prank(alice);
        punkContract.offerPunkForSaleToAddress(0, 0, address(
↳ directLoanFixedOfferRedeploy));


        // TOKEN APPROVAL
        vm.prank(bobby);
        token.approve(address(directLoanFixedOfferRedeploy), 10
↳ _000000000000000000);

        // PREPARING SINGNATURE
        LoanData.Offer memory offer = declareOffer();
        LoanData.Signature memory signaturefi =
↳ getBobbyOfferSignature();

        // ALICE ACCEPTS BOBBY'S OFFER
        LoanData.BorrowerSettings memory borrowerSettings;
        vm.prank(alice);
        directLoanFixedOfferRedeploy.acceptOffer(offer,
↳ signaturefi, borrowerSettings);

        // LOGS
        console.log("****** STATE 1 *******");
        console.log("TX: ALICE ---> ACCEPT OFFER");
        console.log(" ");
        getBalances(1);

        // MINT AND TRNSFER PROMISSORY_NOTE
        nftObligationReceipt.balanceOf(address(alice));
        vm.startPrank(alice);
        directLoanFixedOfferRedeploy.mintObligationReceipt(1);
        nftObligationReceipt.balanceOf(address(alice));
```

```
36          nftObligationReceipt.approve(address(carla),
↳ 5929418158485165765);
37          nftObligationReceipt.transferFrom(address(alice), address(
↳ carla), 5929418158485165765);
38          vm.stopPrank();
39
40          // ALICE TRY TO STEAL NFT
41          vm.startPrank(carla);
42          token.approve(address(directLoanFixedOfferRedeploy), 12
↳ _000000000000000000);
43          directLoanFixedOfferRedeploy.payBackLoan(1);
44          vm.stopPrank();
45
46          // LOGS
47          console.log("****** STATE 2 *******");
48          console.log("TX: ALICE ---> TRANSFER OBLIGATION RECEIPT TO
↳  CARLA");
49          console.log("TX: CARLA ---> PAY BACK LOAN");
50          console.log(" ");
51          getBalances(1);
52      }
```

Output

```
[PASS] test_8_ObligationReceiptExchange() (gas: 798521)
Logs:
  ********************************************************************************************
  **********  TEST 8: General repayment cycle with obligation receipt transfer  **********
  ********************************************************************************************

  ****** STATE 0 (ENV) *******

  ****** BALANCES ******
  Balance Of Admin        --->  99890000000000000000000
  Balance Of Alice        --->  5000000000000000000
  Balance Of Bobby        --->  10000000000000000000000
  Balance Of Carla        --->  5000000000000000000
  Owner of the NFT        --->  0x61A1D7fD8C9bbd82932D99DFD47bD2581C23b08c

  ****** STATE 1 *******
  TX: ALICE ---> ACCEPT OFFER

  ****** BALANCES ******
  Balance Of Admin        --->  99890000000000000000000
  Balance Of Alice        --->  6000000000000000000
  Balance Of Bobby        --->  99000000000000000000000
  Balance Of Carla        --->  5000000000000000000
  Owner of the NFT        --->  0x5f67D76CDa93c064D8f8386fdd5Bd0caE49957A3

  ****** LOAN DATA ******
  loanPrincipalAmount       --->  1000000000000000000
  maximumRepaymentAmount    --->  1200000000000000000
  nftCollateralId           --->  0
  loanERC20Denomination     --->  0xe6CCdc781deb32A0463fb1d81D0fF86d2E184941
  loanDuration              --->  864000
  interestRateForDuration   --->  0
  loanAdminFeeInBasisPoints  --->  500
  nftCollateralWrapper      --->  0x24a683C0D64AC3B75d1e1aE6D9EC4d5bcA11E387
  loanStartTime             --->  1
  nftCollateralContract     --->  0x7D28001937fe8e131F76DaE9E9947adEDbD0abdE
  borrower                  --->  0x61A1D7fD8C9bbd82932D99DFD47bD2581C23b08c
  LOAN LIQUIDATED / REPAYED  --->  false

  ****** STATE 2 *******
  TX: ALICE ---> TRANSFER OBLIGATION RECEIPT TO CARLA
  TX: CARLA ---> PAY BACK LOAN

  ****** BALANCES ******
  Balance Of Admin        --->  99890010000000000000000
  Balance Of Alice        --->  6000000000000000000
  Balance Of Bobby        --->  100190000000000000000
  Balance Of Carla        --->  3800000000000000000
  Owner of the NFT        --->  0xAB846040C285175d375b26a8074cCaB4E0A0075B

  ****** 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 8.1:

Script

Listing 14

```
function test_82_ObligationReceiptExchangeAttack() public {

        // INITIAL STATE LOGS
        console.log("****** STATE 0 (ENV) *******");
        console.log(" ");
        getBalances();

        vm.prank(alice);
        punkContract.offerPunkForSaleToAddress(0, 0, address(
↳ directLoanFixedOfferRedeploy));


        // TOKEN APPROVAL
        vm.prank(bobby);
        token.approve(address(directLoanFixedOfferRedeploy), 10
↳ _000000000000000000);

        // PREPARING SINGNATURE
        LoanData.Offer memory offer = declareOffer();
        LoanData.Signature memory signaturefi =
↳ getBobbyOfferSignature();

        // ALICE ACCEPTS BOBBY'S OFFER
        LoanData.BorrowerSettings memory borrowerSettings;
        vm.prank(alice);
        directLoanFixedOfferRedeploy.acceptOffer(offer,
↳ signaturefi, borrowerSettings);

        // LOGS
        console.log("****** STATE 1 *******");
        console.log("TX: ALICE ---> ACCEPT OFFER");
        console.log(" ");
        getBalances(1);

        // MINT AND TRNSFER PROMISSORY_NOTE
        nftObligationReceipt.balanceOf(address(alice));
        vm.startPrank(alice);
        directLoanFixedOfferRedeploy.mintObligationReceipt(1);
        nftObligationReceipt.balanceOf(address(alice));
```

```
36          nftObligationReceipt.approve(address(carla),
↳ 5929418158485165765);
37          nftObligationReceipt.transferFrom(address(alice), address(
↳ carla), 5929418158485165765);
38
39          // ALICE TRY TO STEAL NFT
40          token.approve(address(directLoanFixedOfferRedeploy), 12
↳ _000000000000000000);
41          directLoanFixedOfferRedeploy.payBackLoan(1);
42          vm.stopPrank();
43
44          // LOGS
45          console.log("****** STATE 2 *******");
46          console.log("TX: ALICE ---> TRANSFER OBLIGATION RECEIPT TO
↳  CARLA");
47          console.log("TX: ALICE ---> PAY BACK LOAN");
48          console.log(" ");
49          getBalances(1);
50      }
```

Output

```
[PASS] test_82_ObligationReceiptExchangeAttack() (gas: 795549)
Logs:
  ********************************************************************************
  **********  TEST 8.2: Obligation receipt transfer + repayment attack  **********
  ********************************************************************************

  ****** STATE 0 (ENV) *******

  ****** BALANCES ******
  Balance Of Admin        --->  99890000000000000000000
  Balance Of Alice        --->  5000000000000000000
  Balance Of Bobby        --->  10000000000000000000000
  Balance Of Carla        --->  5000000000000000000
  Owner of the NFT        --->  0x61A1D7fD8C9bbd82932D99DFD47bD2581C23b08c

  ****** STATE 1 *******
  TX: ALICE ---> ACCEPT OFFER

  ****** BALANCES ******
  Balance Of Admin        --->  99890000000000000000000
  Balance Of Alice        --->  6000000000000000000
  Balance Of Bobby        --->  99000000000000000000000
  Balance Of Carla        --->  5000000000000000000
  Owner of the NFT        --->  0x5f67D76CDa93c064D8f8386fdd5Bd0caE49957A3

  ****** LOAN DATA ******
  loanPrincipalAmount        --->  1000000000000000000
  maximumRepaymentAmount     --->  1200000000000000000
  nftCollateralId            --->  0
  loanERC20Denomination      --->  0xe6CCdc781deb32A0463fb1d81D0fF86d2E184941
  loanDuration               --->  864000
  interestRateForDuration    --->  0
  loanAdminFeeInBasisPoints  --->  500
  nftCollateralWrapper       --->  0x24a683C0D64AC3B75d1e1aE6D9EC4d5bcA11E387
  loanStartTime              --->  1
  nftCollateralContract      --->  0x7D28001937fe8e131F76DaE9E9947adEDbD0abdE
  borrower                   --->  0x61A1D7fD8C9bbd82932D99DFD47bD2581C23b08c
  LOAN LIQUIDATED / REPAYED  --->  false

  ****** STATE 2 *******
  TX: ALICE ---> TRANSFER OBLIGATION RECEIPT TO CARLA
  TX: ALICE ---> PAY BACK LOAN

  ****** BALANCES ******
  Balance Of Admin        --->  99890010000000000000000
  Balance Of Alice        --->  4800000000000000000
  Balance Of Bobby        --->  10019000000000000000000
  Balance Of Carla        --->  5000000000000000000
  Owner of the NFT        --->  0xAB846040C285175d375b26a8074cCaB4E0A0075B

  ****** 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 9:

Script

Listing 15

```
function test_9_airdrop() public {
        function test_9_airdrop() public {

        // INITIAL STATE LOGS
        console.log("****** STATE 0 (ENV) *******");
        console.log(" ");
        getBalances();

        // AIRDROP RECEIVER (alt 2)
        vm.startPrank(admin);
        airdropReceiverFactory = new AirdropReceiverFactory(address(admin), address(nftFiHub));
        airdropReceiverToClone = new AirdropReceiver(address(nftFiHub));
        permittedNFTsAndTypeRegistry.setNftType('AirdropWrapper', address(airdropReceiverToClone));

        // SETTING CONTRACTS AND PERMISIONS
        permittedNftContracts.push(address(airdropReceiverToClone));
        permittedNftTypes.push('AIRDROP_RECEIVER');
        nftFiHub.setContract('AIRDROP_RECEIVER', address(airdropReceiverToClone));
        nftFiHub.setContract('AIRDROP_FACTORY', address(airdropReceiverFactory));
        nftFiHub.setContract('PERMITTED_NFTS', address(permittedNFTsAndTypeRegistry));
        nftFiHub.setContract('PERMITTED_AIRDROPS', address(permittedAirdrop));

        vm.stopPrank();
        vm.startPrank(alice);
        uint256 aliceWrapperId;
        address aliceAirdropReceiverAddr;
        console.logBytes(bytes(ContractKeys.AIRDROP_WRAPPER_STRING));
        vm.stopPrank();

        // ------------------------------------------
        // ALICE BORROWS GIVING PUNK AS COLLATERAL
```

```
32          // ----------------------------------------
33
34          vm.prank(alice);
35          punkContract.offerPunkForSaleToAddress(0, 0, address(
↳ directLoanFixedOfferRedeploy));
36
37          // TOKEN APPROVAL
38          vm.prank(bobby);
39          token.approve(address(directLoanFixedOfferRedeploy), 10
↳ _000000000000000000);
40
41          // PREPARING SINGNATURE
42          LoanData.Offer memory offer = declareOffer();
43          LoanData.Signature memory signaturefi =
↳ getBobbyOfferSignature();
44
45          console.log("0xx", punkContract.punkIndexToAddress(0));
46          // ALICE ACCEPTS BOBBY'S OFFER
47          LoanData.BorrowerSettings memory borrowerSettings;
48          vm.prank(alice);
49          directLoanFixedOfferRedeploy.acceptOffer(offer,
↳ signaturefi, borrowerSettings);
50          console.log("0yy", punkContract.punkIndexToAddress(0));
51
52          // LOGS
53          console.log("****** STATE 1 *******");
54          console.log("TX: ALICE ---> ACCEPT BOBBY'S OFFER");
55          console.log(" ");
56          getBalances(1);
57
58          // 5 DAYS LATER
59          console.log("---------------");
60          console.log("5 DAYS LATER...");
61          console.log("---------------");
62          console.log(" ");
63          vm.warp(5 days);
64          vm.prank(alice);
65          directLoanFixedOfferRedeploy.wrapCollateral(1);
66
67          // ALICE PULLS THE AIRDROP
68          bytes memory encodedFunctionData = abi.encodeWithSignature
↳ ("mintNFT(uint256)", 1);
69          vm.prank(alice);
```

```
70          AirdropReceiver(address(0
↳ x6d69556Cf844F68065f814F1E9E00854dDf91A28)).pullAirdrop(address(
↳ nftContract), encodedFunctionData);
71
72          // THE LOAN IS PAID BY ALICE
73          vm.startPrank(alice);
74          token.approve(address(directLoanFixedOfferRedeploy), 12
↳ _000000000000000000);
75          directLoanFixedOfferRedeploy.payBackLoan(1);
76          AirdropReceiver(address(0
↳ x6d69556Cf844F68065f814F1E9E00854dDf91A28)).unwrap(alice);
77          AirdropReceiver(address(0
↳ x6d69556Cf844F68065f814F1E9E00854dDf91A28)).drainERC721Airdrop(
↳ address(nftContract), 10, alice);
78
79          // LOGS
80          vm.stopPrank();
81          console.log("ownerOfNFT", nftContract.ownerOf(10));
82          console.log("0zz", punkContract.punkIndexToAddress(0));
83          console.log("alice", alice);
84      }
```

Output

```
Running 1 test for test/punk.t.sol:punkTest
[FAIL. Reason: NFT not successfully transferred] test_9_airdrop() (gas: 4762138)
Logs:
  ********************************************************************************
  **********  TEST 9: Airdrop functionality (Airdrop receiver - Wrap - Unwrap)  **********
  ********************************************************************************

  ****** STATE 0 (ENV) *******

  ****** BALANCES ******
  Balance Of Admin        --->  998900000000000000000000
  Balance Of Alice        --->  5000000000000000000
  Balance Of Bobby        --->  10000000000000000000
  Balance Of Carla        --->  5000000000000000000
  Owner of the NFT        --->  0x61A1D7fD8C9bbd82932D99DFD47bD2581C23b08c

  0x41697264726f7057726170706572
  0xx 0x61A1D7fD8C9bbd82932D99DFD47bD2581C23b08c
  0yy 0xC03dCc12ABceB85ca9eB6b23331810F9cA7d40Fc
  ****** STATE 1 *******
  TX: ALICE ---> ACCEPT BOBBY'S OFFER

  ****** BALANCES ******
  Balance Of Admin        --->  998900000000000000000000
  Balance Of Alice        --->  6000000000000000000
  Balance Of Bobby        --->  9900000000000000000000
  Balance Of Carla        --->  5000000000000000000
  Owner of the NFT        --->  0xC03dCc12ABceB85ca9eB6b23331810F9cA7d40Fc

  ****** LOAN DATA ******
  loanPrincipalAmount        --->  1000000000000000000
  maximumRepaymentAmount     --->  1200000000000000000
  nftCollateralId            --->  0
  loanERC20Denomination      --->  0x2efD0Bc8Fc1050Ed24f52DBCeC99644c0072B937
  loanDuration               --->  864000
  interestRateForDuration    --->  0
  loanAdminFeeInBasisPoints  --->  500
  nftCollateralWrapper       --->  0x24a683C0D64AC3B75d1e1aE6D9EC4d5bcA11E387
  loanStartTime              --->  1
  nftCollateralContract      --->  0xcB5Ea74a0483B47A6e6560D4182C2d54c3C4805a
  borrower                   --->  0x61A1D7fD8C9bbd82932D99DFD47bD2581C23b08c
  LOAN LIQUIDATED / REPAYED  --->  false

  --------------
  5 DAYS LATER...
  --------------


Test result: FAILED. 0 passed; 1 failed; finished in 5.82ms

Failing tests:
Encountered 1 failing test in test/punk.t.sol:punkTest
[FAIL. Reason: NFT not successfully transferred] test_9_airdrop() (gas: 4762138)

Encountered a total of 1 failing tests, 0 tests succeeded
```

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

```
AirdropReceiver (contracts/airdrop/AirdropReceiver.sol#28-314) is an upgradeable contract that does not protect its initialize functions: Aird
r.wrap(address,address,address,uint256) (contracts/airdrop/AirdropReceiver.sol#91-117)AirdropReceiver.unwrap(address) (contracts/airdrop/Airdr

Reentrancy in AirdropReceiver.unwrap(address) (contracts/airdrop/AirdropReceiver.sol#119-130):
        External calls:
        - _transferNFT(nftTransferWrapper,address(this),_receiver,wrappedNft,wrappedNftId) (contracts/airdrop/AirdropReceiver.sol#122)
                - _nftTransferWrapper.functionDelegateCall(abi.encodeWithSelector(INftWrapper(_nftTransferWrapper).transferNFT.selector,_sende
#212-221)
                - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#191)
        State variables written after the call(s):
        - nftTransferWrapper = address(0) (contracts/airdrop/AirdropReceiver.sol#129)
        AirdropReceiver.nftTransferWrapper (contracts/airdrop/AirdropReceiver.sol#36) can be used in cross function reentrancies:
        - AirdropReceiver._receiveAndWrap(address,address,address,uint256) (contracts/airdrop/AirdropReceiver.sol#296-313)
        - AirdropReceiver.nftTransferWrapper (contracts/airdrop/AirdropReceiver.sol#36)
        - AirdropReceiver.unwrap(address) (contracts/airdrop/AirdropReceiver.sol#119-130)
        - AirdropReceiver.wrap(address,address,address,uint256) (contracts/airdrop/AirdropReceiver.sol#91-117)
        - wrappedNft = address(0) (contracts/airdrop/AirdropReceiver.sol#127)
        AirdropReceiver.wrappedNft (contracts/airdrop/AirdropReceiver.sol#38) can be used in cross function reentrancies:
        - AirdropReceiver._receiveAndWrap(address,address,address,uint256) (contracts/airdrop/AirdropReceiver.sol#296-313)
        - AirdropReceiver.drainERC1155Airdrop(address,uint256,address) (contracts/airdrop/AirdropReceiver.sol#193-203)
        - AirdropReceiver.drainERC721Airdrop(address,uint256,address) (contracts/airdrop/AirdropReceiver.sol#176-184)
        - AirdropReceiver.onERC1155BatchReceived(address,address,uint256[],uint256[],bytes) (contracts/airdrop/AirdropReceiver.sol#273-294)
        - AirdropReceiver.onERC1155Received(address,address,uint256,uint256,bytes) (contracts/airdrop/AirdropReceiver.sol#251-271)
        - AirdropReceiver.onERC721Received(address,address,uint256,bytes) (contracts/airdrop/AirdropReceiver.sol#231-249)
        - AirdropReceiver.unwrap(address) (contracts/airdrop/AirdropReceiver.sol#119-130)
        - AirdropReceiver.wrap(address,address,address,uint256) (contracts/airdrop/AirdropReceiver.sol#91-117)
        - AirdropReceiver.wrappedNft (contracts/airdrop/AirdropReceiver.sol#38)
        - wrappedNftId = 0 (contracts/airdrop/AirdropReceiver.sol#128)
        AirdropReceiver.wrappedNftId (contracts/airdrop/AirdropReceiver.sol#39) can be used in cross function reentrancies:
        - AirdropReceiver._receiveAndWrap(address,address,address,uint256) (contracts/airdrop/AirdropReceiver.sol#296-313)
        - AirdropReceiver.drainERC1155Airdrop(address,uint256,address) (contracts/airdrop/AirdropReceiver.sol#193-203)
        - AirdropReceiver.drainERC721Airdrop(address,uint256,address) (contracts/airdrop/AirdropReceiver.sol#176-184)
        - AirdropReceiver.unwrap(address) (contracts/airdrop/AirdropReceiver.sol#119-130)
        - AirdropReceiver.wrap(address,address,address,uint256) (contracts/airdrop/AirdropReceiver.sol#91-117)
        - AirdropReceiver.wrappedNftId (contracts/airdrop/AirdropReceiver.sol#39)
Reentrancy in AirdropReceiver.wrap(address,address,address,uint256) (contracts/airdrop/AirdropReceiver.sol#91-117):
        External calls:
        - _transferNFT(nftTransferWrapper,_from,address(this),_nftCollateralContract,_nftCollateralId) (contracts/airdrop/AirdropReceiver.sol#
                - _nftTransferWrapper.functionDelegateCall(abi.encodeWithSelector(INftWrapper(_nftTransferWrapper).transferNFT.selector,_sende
#212-221)
                - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#191)
        State variables written after the call(s):
        - wrappedNft = _nftCollateralContract (contracts/airdrop/AirdropReceiver.sol#111)
        AirdropReceiver.wrappedNft (contracts/airdrop/AirdropReceiver.sol#38) can be used in cross function reentrancies:
        - AirdropReceiver._receiveAndWrap(address,address,address,uint256) (contracts/airdrop/AirdropReceiver.sol#296-313)
        - AirdropReceiver.drainERC1155Airdrop(address,uint256,address) (contracts/airdrop/AirdropReceiver.sol#193-203)
        - AirdropReceiver.drainERC721Airdrop(address,uint256,address) (contracts/airdrop/AirdropReceiver.sol#176-184)
        - AirdropReceiver.onERC1155BatchReceived(address,address,uint256[],uint256[],bytes) (contracts/airdrop/AirdropReceiver.sol#273-294)
        - AirdropReceiver.onERC1155Received(address,address,uint256,uint256,bytes) (contracts/airdrop/AirdropReceiver.sol#251-271)
        - AirdropReceiver.onERC721Received(address,address,uint256,bytes) (contracts/airdrop/AirdropReceiver.sol#231-249)
        - AirdropReceiver.unwrap(address) (contracts/airdrop/AirdropReceiver.sol#119-130)
        - AirdropReceiver.wrap(address,address,address,uint256) (contracts/airdrop/AirdropReceiver.sol#91-117)
        - AirdropReceiver.wrappedNft (contracts/airdrop/AirdropReceiver.sol#38)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

AirdropReceiver.pullAirdrop(address,bytes) (contracts/airdrop/AirdropReceiver.sol#132-141) ignores return value by _target.functionCall(_data)
AirdropReceiver._transferNFT(address,address,address,address,uint256) (contracts/airdrop/AirdropReceiver.sol#205-222) ignores return value by
ipient,_nftCollateralContract,_nftCollateralId),NFT was not successfully transferred) (contracts/airdrop/AirdropReceiver.sol#212-221)
ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#394-416) ignores re
721/ERC721.sol#401-412)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).retval (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#40
394-416) potentially used before declaration: retval == IERC721Receiver.onERC721Received.selector (node_modules/@openzeppelin/contracts/token/
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#40
394-416) potentially used before declaration: reason.length == 0 (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#404)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#40
394-416) potentially used before declaration: revert(uint256,uint256)(32 + reason,mload(uint256)(reason)) (node_modules/@openzeppelin/contract
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

Reentrancy in AirdropReceiver.unwrap(address) (contracts/airdrop/AirdropReceiver.sol#119-130):
        External calls:
```

```
        - _transferNFT(nftTransferWrapper,address(this),_receiver,wrappedNft,wrappedNftId) (contracts/airdrop/AirdropReceiver.sol#122)
                - _nftTransferWrapper.functionDelegateCall(abi.encodeWithSelector(INftWrapper(_nftTransferWrapper).transferNFT.selector,_sende
#212-221)
                - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#191)
        State variables written after the call(s):
        - beneficiary = address(0) (contracts/airdrop/AirdropReceiver.sol#126)
Reentrancy in AirdropReceiver.wrap(address,address,address,uint256) (contracts/airdrop/AirdropReceiver.sol#91-117):
        External calls:
        - _transferNFT(nftTransferWrapper,_from,address(this),_nftCollateralContract,_nftCollateralId) (contracts/airdrop/AirdropReceiver.sol#
                - _nftTransferWrapper.functionDelegateCall(abi.encodeWithSelector(INftWrapper(_nftTransferWrapper).transferNFT.selector,_sende
#212-221)
                - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#191)
        State variables written after the call(s):
        - beneficiary = _beneficiary (contracts/airdrop/AirdropReceiver.sol#110)
        - wrappedNftId = _nftCollateralId (contracts/airdrop/AirdropReceiver.sol#112)
        - wrapping_ = false (contracts/airdrop/AirdropReceiver.sol#116)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in AirdropReceiver.unwrap(address) (contracts/airdrop/AirdropReceiver.sol#119-130):
        External calls:
        - _transferNFT(nftTransferWrapper,address(this),_receiver,wrappedNft,wrappedNftId) (contracts/airdrop/AirdropReceiver.sol#122)
                - _nftTransferWrapper.functionDelegateCall(abi.encodeWithSelector(INftWrapper(_nftTransferWrapper).transferNFT.selector,_sende
#212-221)
                - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#191)
        Event emitted after the call(s):
        - NftUnwrapped(wrappedNft,wrappedNftId,_receiver,msg.sender) (contracts/airdrop/AirdropReceiver.sol#124)
Reentrancy in AirdropReceiver.wrap(address,address,address,uint256) (contracts/airdrop/AirdropReceiver.sol#91-117):
        External calls:
        - _transferNFT(nftTransferWrapper,_from,address(this),_nftCollateralContract,_nftCollateralId) (contracts/airdrop/AirdropReceiver.sol#
                - _nftTransferWrapper.functionDelegateCall(abi.encodeWithSelector(INftWrapper(_nftTransferWrapper).transferNFT.selector,_sende
#212-221)
                - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#191)
        Event emitted after the call(s):
        - NftWrapped(_nftCollateralContract,_nftCollateralId,_from,_beneficiary,msg.sender) (contracts/airdrop/AirdropReceiver.sol#114)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

AirdropReceiver._getSelector(bytes) (contracts/airdrop/AirdropReceiver.sol#224-229) uses assembly
        - INLINE ASM (contracts/airdrop/AirdropReceiver.sol#226-228)
ContractKeys.getIdFromStringKey(string) (contracts/utils/ContractKeys.sol#31-38) uses assembly
        - INLINE ASM (contracts/utils/ContractKeys.sol#35-37)
ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#394-416) uses assem
        - INLINE ASM (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#408-410)
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)
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', '^0.8.2']
        - 0.8.4 (contracts/airdrop/AirdropReceiver.sol#3)
        - 0.8.4 (contracts/interfaces/INftWrapper.sol#3)
        - 0.8.4 (contracts/interfaces/INftfiHub.sol#3)
        - 0.8.4 (contracts/interfaces/IPermittedAirdrops.sol#3)
        - 0.8.4 (contracts/interfaces/IPermittedNFTs.sol#3)
        - 0.8.4 (contracts/interfaces/IPunks.sol#3)
        - 0.8.4 (contracts/nftTypeRegistry/nftTypes/PunkWrapper.sol#2)
        - 0.8.4 (contracts/utils/ContractKeys.sol#3)
        - ^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/ERC1155/utils/ERC1155Holder.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155/utils/ERC1155Receiver.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/draft-IERC20Permit.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.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/extensions/ERC721Enumerable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/extensions/IERC721Enumerable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/extensions/IERC721Metadata.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/utils/ERC721Holder.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/introspection/ERC165.sol#4)
```

```
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/IERC165.sol#4)
        - ^0.8.1 (node_modules/@openzeppelin/contracts/utils/Address.sol#4)
        - ^0.8.2 (node_modules/@openzeppelin/contracts/proxy/utils/Initializable.sol#4)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Address.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#114-120) is never used and should
Address.functionDelegateCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#174-176) is never used and should be remov
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
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
ERC721._burn(uint256) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#303-317) is never used and should be removed
Initializable._disableInitializers() (node_modules/@openzeppelin/contracts/proxy/utils/Initializable.sol#131-137) is never used and should be
SafeERC20.safeApprove(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#46-59) is never used and sh
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#70-81) is never
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#61-68) is never
SafeERC20.safePermit(IERC20Permit,address,address,uint256,uint256,uint8,bytes32,bytes32) (node_modules/@openzeppelin/contracts/token/ERC20/uti
SafeERC20.safeTransferFrom(IERC20,address,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#30-37) is nev
Strings.toHexString(address) (node_modules/@openzeppelin/contracts/utils/Strings.sol#72-74) is never used and should be removed
Strings.toHexString(uint256) (node_modules/@openzeppelin/contracts/utils/Strings.sol#41-52) is never used and should be removed
Strings.toHexString(uint256,uint256) (node_modules/@openzeppelin/contracts/utils/Strings.sol#57-67) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version0.8.4 (contracts/airdrop/AirdropReceiver.sol#3) allows old versions
Pragma version0.8.4 (contracts/interfaces/INftWrapper.sol#3) allows old versions
Pragma version0.8.4 (contracts/interfaces/INftfiHub.sol#3) allows old versions
Pragma version0.8.4 (contracts/interfaces/IPermittedAirdrops.sol#3) allows old versions
Pragma version0.8.4 (contracts/interfaces/IPermittedNFTs.sol#3) allows old versions
Pragma version0.8.4 (contracts/interfaces/IPunks.sol#3) allows old versions
Pragma version0.8.4 (contracts/nftTypeRegistry/nftTypes/PunkWrapper.sol#2) allows old versions
Pragma version0.8.4 (contracts/utils/ContractKeys.sol#3) allows old versions
Pragma version^0.8.2 (node_modules/@openzeppelin/contracts/proxy/utils/Initializable.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/ERC1155/utils/ERC1155Holder.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155/utils/ERC1155Receiver.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/extensions/draft-IERC20Permit.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/ERC721.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/extensions/ERC721Enumerable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/extensions/IERC721Enumerable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/extensions/IERC721Metadata.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/introspection/ERC165.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/IERC165.sol#4) allows old versions
solc-0.8.4 is not recommended for deployment
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)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter AirdropReceiver.initialize(address)._to (contracts/airdrop/AirdropReceiver.sol#82) is not in mixedCase
Parameter AirdropReceiver.wrap(address,address,address,uint256)._from (contracts/airdrop/AirdropReceiver.sol#92) is not in mixedCase
Parameter AirdropReceiver.wrap(address,address,address,uint256)._beneficiary (contracts/airdrop/AirdropReceiver.sol#93) is not in mixedCase
Parameter AirdropReceiver.wrap(address,address,address,uint256)._nftCollateralContract (contracts/airdrop/AirdropReceiver.sol#94) is not in mi
Parameter AirdropReceiver.wrap(address,address,address,uint256)._nftCollateralId (contracts/airdrop/AirdropReceiver.sol#95) is not in mixedCas
Parameter AirdropReceiver.unwrap(address)._receiver (contracts/airdrop/AirdropReceiver.sol#119) is not in mixedCase
Parameter AirdropReceiver.pullAirdrop(address,bytes)._target (contracts/airdrop/AirdropReceiver.sol#132) is not in mixedCase
Parameter AirdropReceiver.pullAirdrop(address,bytes)._data (contracts/airdrop/AirdropReceiver.sol#132) is not in mixedCase
Parameter AirdropReceiver.supportsInterface(bytes4)._interfaceId (contracts/airdrop/AirdropReceiver.sol#146) is not in mixedCase
Parameter AirdropReceiver.onERC721Received(address,address,uint256,bytes)._from (contracts/airdrop/AirdropReceiver.sol#233) is not in mixedCas
Parameter AirdropReceiver.onERC721Received(address,address,uint256,bytes)._tokenId (contracts/airdrop/AirdropReceiver.sol#234) is not in mixed
Parameter AirdropReceiver.onERC721Received(address,address,uint256,bytes)._data (contracts/airdrop/AirdropReceiver.sol#235) is not in mixedCas
Parameter AirdropReceiver.onERC1155Received(address,address,uint256,uint256,bytes)._from (contracts/airdrop/AirdropReceiver.sol#253) is not in
Parameter AirdropReceiver.onERC1155Received(address,address,uint256,uint256,bytes)._id (contracts/airdrop/AirdropReceiver.sol#254) is not in m
Parameter AirdropReceiver.onERC1155Received(address,address,uint256,uint256,bytes)._data (contracts/airdrop/AirdropReceiver.sol#256) is not in
Parameter AirdropReceiver.onERC1155BatchReceived(address,address,uint256[],uint256[],bytes)._from (contracts/airdrop/AirdropReceiver.sol#275)
Parameter AirdropReceiver.onERC1155BatchReceived(address,address,uint256[],uint256[],bytes)._ids (contracts/airdrop/AirdropReceiver.sol#276) i
Parameter AirdropReceiver.onERC1155BatchReceived(address,address,uint256[],uint256[],bytes)._data (contracts/airdrop/AirdropReceiver.sol#278)
Parameter PunkWrapper.transferNFT(address,address,address,uint256)._sender (contracts/nftTypeRegistry/nftTypes/PunkWrapper.sol#25) is not in m
Parameter PunkWrapper.transferNFT(address,address,address,uint256)._recipient (contracts/nftTypeRegistry/nftTypes/PunkWrapper.sol#26) is not i
Parameter PunkWrapper.transferNFT(address,address,address,uint256)._nftContract (contracts/nftTypeRegistry/nftTypes/PunkWrapper.sol#27) is not
Parameter PunkWrapper.transferNFT(address,address,address,uint256)._nftId (contracts/nftTypeRegistry/nftTypes/PunkWrapper.sol#28) is not in mi
Parameter PunkWrapper.isOwner(address,address,uint256)._owner (contracts/nftTypeRegistry/nftTypes/PunkWrapper.sol#40) is not in mixedCase
Parameter PunkWrapper.isOwner(address,address,uint256)._nftContract (contracts/nftTypeRegistry/nftTypes/PunkWrapper.sol#41) is not in mixedCas
Parameter PunkWrapper.isOwner(address,address,uint256)._tokenId (contracts/nftTypeRegistry/nftTypes/PunkWrapper.sol#42) is not in mixedCase
Parameter PunkWrapper.wrapAirdropReceiver(address,address,uint256,address)._recipient (contracts/nftTypeRegistry/nftTypes/PunkWrapper.sol#48)
Parameter PunkWrapper.wrapAirdropReceiver(address,address,uint256,address)._nftContract (contracts/nftTypeRegistry/nftTypes/PunkWrapper.sol#49
Parameter PunkWrapper.wrapAirdropReceiver(address,address,uint256,address)._nftId (contracts/nftTypeRegistry/nftTypes/PunkWrapper.sol#50) is n
Parameter PunkWrapper.wrapAirdropReceiver(address,address,uint256,address)._beneficiary (contracts/nftTypeRegistry/nftTypes/PunkWrapper.sol#51
Parameter ContractKeys.getIdFromStringKey(string)._key (contracts/utils/ContractKeys.sol#31) is not in mixedCase
Function IERC20Permit.DOMAIN_SEPARATOR() (node_modules/@openzeppelin/contracts/token/ERC20/extensions/draft-IERC20Permit.sol#59) is not in mix
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

- 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

```
Report for contracts/airdrop/AirdropReceiver.sol
https://dashboard.mythx.io/#/console/analyses/28dd4b11-b951-424d-9869-fba4a642a821
```

Line	SWC Title	Severity	Short Description
28	(SWC-123) Requirement Violation	Low	Requirement violation.
60	(SWC-123) Requirement Violation	Low	Requirement violation.
183	(SWC-123) Requirement Violation	Low	Requirement violation.
200	(SWC-123) Requirement Violation	Low	Requirement violation.

- 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 modifier.

THANK YOU FOR CHOOSING

// HALBORN