SHELLBOXES

NFHeroes

Smart Contract Security Audit

Prepared by ShellBoxes

March 16th, 2022 - March 22nd, 2022

Shellboxes.com

contact@shellboxes.com

Document Properties

Client	NFHeroes
Version	1.0
Classification	Public

Scope

The NFHeroes Contract in the NFHeroes Repository

Repo	Commit Hash
`https://github.com/boring-bananas-co/non-fungible-heroes-token`	d10170be296d65b710726fb53b1281098ef01974

Files	MD5 Hash
CreditVault.sol	2dafd684c898664488b6d833e9d6fa9f
LoreToken.sol	54e8175d8934d77ef1edff47bdd2f5da
NftLocker.sol	86d69c79dce4129f06690027e1941659

Re-Audit

Repo	Commit Hash
`https://github.com/boring-bananas-co/non-fungible-heroes-token`	ac4e145956e45537631c6421fcc4b6eeb41b7049

Contacts

COMPANY	EMAIL
ShellBoxes	contact@shellboxes.com

Contents

1 Introduction

NFHeroes engaged ShellBoxes to conduct a security assessment on the NFHeroes beginning on March 16th, 2022 and ending March 22nd, 2022. In this report, we detail our methodical approach to evaluate potential security issues associated with the implementation of smart contracts, by exposing possible semantic discrepancies between the smart contract code and design document, and by recommending additional ideas to optimize the existing code. Our findings indicate that the current version of smart contracts can still be enhanced further due to the presence of many security and performance concerns.

This document summarizes the findings of our audit.

1.1 About NFHeroes

$LORE is an in-game token used for the NFH Interactive Quest experience. It will be accrued by users and utilized to access digital goods in the NFH ecosystem. The team will not be providing any liquidity for the token and does not endorse exchange of $LORE for any other currency. It is intended for use as an in-game reward system only.

Issuer	NFHeroes
Website	`https://www.nfheroes.io`
Type	Solidity Smart Contract
Audit Method	Whitebox

1.2 Approach & Methodology

ShellBoxes used a combination of manual and automated security testing to achieve a balance between efficiency, timeliness, practicability, and correctness within the audit's scope. While manual testing is advised for identifying problems in logic, procedure, and implementation, automated testing techniques help to expand the coverage of smart contracts and can quickly detect code that does not comply with security best practices.

1.2.1 Risk Methodology

Vulnerabilities or bugs identified by ShellBoxes are ranked using a risk assessment technique that considers both the LIKELIHOOD and IMPACT of a security incident. This framework is effective at conveying the features and consequences of technological vulnerabilities.

Its quantitative paradigm enables repeatable and precise measurement, while also revealing the underlying susceptibility characteristics that were used to calculate the Risk scores. A risk level will be assigned to each vulnerability on a scale of 5 to 1, with 5 indicating the greatest possibility or impact.

- Likelihood quantifies the probability of a certain vulnerability being discovered and exploited in the untamed.
- Impact quantifies the technical and economic costs of a successful attack.
- Severity indicates the risk's overall criticality.

Probability and impact are classified into three categories: H, M, and L, which correspond to high, medium, and low, respectively. Severity is determined by probability and impact and is categorized into four levels, namely Critical, High, Medium, and Low.

Impact	High	Critical	High	Medium
	Medium	High	Medium	Low
	Low	Medium	Low	Low
		High	Medium	Low
			Likelihood	

2 Findings Overview

2.1 Summary

The following is a synopsis of our conclusions from our analysis of the NFHeroes implementation. During the first part of our audit, we examine the smart contract source code and run the codebase via a static code analyzer. The objective here is to find known coding problems statically and then manually check (reject or confirm) issues highlighted by the tool. Additionally, we check business logics, system processes, and DeFi-related components manually to identify potential hazards and/or defects.

2.2 Key Findings

In general, these smart contracts are well-designed and constructed, but their implementation might be improved by addressing the discovered flaws, which include , 1 medium-severity, 7 low-severity vulnerabilities.

Vulnerabilities	Severity	Status
Missing Length Verification	MEDIUM	Fixed
Missing Address Verification	LOW	Fixed
Renounce Ownership	LOW	Fixed
Public Function Can Be Called External	LOW	Fixed
Missing Address Verification	LOW	Fixed
Renounce Ownership	LOW	Fixed
For Loop Over Dynamic Array	LOW	Acknowledged
Renounce Ownership	LOW	Fixed

3 Finding Details

A CreditVault.sol

A.1 Missing Address Verification [LOW]

Description:

Certain functions lack a safety check in the address, the address-type argument should include a zero-address test, otherwise, some of the contract's functionality may become inaccessible.

Code:

Listing 1: CreditVault.sol

```
function initialize(
IERC20Upgradeable _loreToken,
address _authProvider,
address _treasuryAddress
) public initializer {
    __Ownable_init();
    __ReentrancyGuard_init();
    __Pausable_init();

    loreToken = _loreToken;
    treasuryAddress = _treasuryAddress;
    updateAuthProvider(_authProvider);
}
```

Risk Level:

Likelihood – 1
Impact - 3

Recommendation:

It is recommended to undertake further validation to prevent injecting zero address. The concerns can be resolved by utilizing a whitelist technique or a modifier.

Status - Fixed

The NFHeroes team has fixed the issue by adding require statements to make sure that the addresses provided in the arguments are different from the address(0).

A.2 Renounce Ownership [LOW]

Description:

Typically, the contract's owner is the account that deploys the contract. As a result, the owner can perform certain privileged activities. The renounceOwnership function is used in smart contracts to renounce ownership. However, if the contract's ownership has never been transferred before renouncing it, it will never have an Owner, which may result in a denial of service.

Code:

Listing 2: CreditVault.sol

```
contract CreditVault is Initializable, UUPSUpgradeable,
    ↪ OwnableUpgradeable, ReentrancyGuardUpgradeable,
    ↪ PausableUpgradeable {
```

Risk Level:

Likelihood – 1
Impact - 2

Recommendation:

It is advised that the Owner cannot call renounceOwnership without first transferring ownership to a different address. Additionally, if a multi-signature wallet is utilized, executing the renounceOwnership method will require two or more users to sign the transaction. Alternatively, the Renounce Ownership functionality can be disabled by overriding it.

Status - Fixed

The NFHeroes team has fixed the issue by overriding the renounceOwnership() function in order to disable the renounce ownership functionality.

A.3 Public Function Can Be Called External [LOW]

Description:

Functions with a public scope that are not called inside the contract should be declared external to reduce the gas fees.

Code:

Listing 3: CreditVault.sol

```
function deposit(uint256 amount) public {
    loreToken.safeTransferFrom(_msgSender(), treasuryAddress, amount);
    emit Deposited(_msgSender(), amount);
}
```

Risk Level:

Likelihood – 1
Impact - 1

Recommendation:

Declare the deposit() function as external.

Status - Fixed

The NFHeroes team has fixed the issue by declaring the deposit function as external.

B LoreToken.sol

B.1 Missing Address Verification [LOW]

Description:

Certain functions lack a safety check in the address, the address-type argument should include a zero-address test, otherwise, some of the contract's functionality may become inaccessible.

Code:

Listing 4: LoreToken.sol

```
constructor(address _treasuryAddress) ERC20('Lore Token', 'LORE') {
    treasuryAddress = _treasuryAddress;
    _mint(treasuryAddress, 100 * MILLION);
}
```

Listing 5: LoreToken.sol

```
function setTreasuryAddress(address _treasuryAddress) external onlyOwner
    ↪ {
    treasuryAddress = _treasuryAddress;
}
```

Risk Level:

Likelihood – 1
Impact - 3

Recommendation:

It is recommended to undertake further validation to prevent injecting zero address. The concerns can be resolved by utilizing a whitelist technique or a modifier.

Status - Fixed

The NFHeroes team has fixed the issue by adding require statements to make sure that the addresses provided in the arguments are different from the address(0).

B.2 Renounce Ownership [LOW]

Description:

Typically, the contract's owner is the account that deploys the contract. As a result, the owner can perform certain privileged activities. The renounceOwnership function is used in smart contracts to renounce ownership. However, if the contract's ownership has never been transferred before renouncing it, it will never have an Owner, which may result in a denial of service.

Code:

Listing 6: LoreToken.sol

```
contract LoreToken is Ownable, ERC20, ERC20Burnable {
```

Risk Level:

Likelihood – 1
Impact - 2

Recommendation:

It is advised that the Owner cannot call renounceOwnership without first transferring ownership to a different address. Additionally, if a multi-signature wallet is utilized, executing the renounceOwnership method will require two or more users to sign the transaction. Alternatively, the Renounce Ownership functionality can be disabled by overriding it.

Status - Fixed

The NFHeroes team has fixed the issue by overriding the renounceOwnership() function in order to disable the renounce ownership functionality.

C NftLocker.sol

C.1 Missing Length Verification [MEDIUM]

Description:

The lock() function takes two arrays as arguments. Every tokenId is associated to a nftCollection, therefore if the function is given two arrays with different lengths, it can generate unexpected behaviors.

Code:

Listing 7: NftLocker.sol

```
function lock(address[] memory nftCollections, uint256[] memory tokenIds ) external {
  for (uint256 i = 0; i < nftCollections.length; i++) {
   StakedNFT[] storage _stakedNfts = stakedNfts[_msgSender()][nftCollections[i]];

   _stakedNfts.push(StakedNFT({
    lockedTs: block.timestamp,
    tokenId: tokenIds[i]
   }));

   userNfts[nftCollections[i]][tokenIds[i]] = _stakedNfts.length - 1;

   IERC721(nftCollections[i]).safeTransferFrom(
    _msgSender(),
    address(this),
    tokenIds[i]
   );
  }
}
```

Risk Level:

Likelihood – 3
Impact - 3

Recommendation:

Use a require statement to make sure the length of the two arrays are equals.

Status - Fixed

The NFHeroes team has fixed the issue by adding a require statement to make sure the array provided in the arguments have the same length.

C.2 For Loop Over Dynamic Array [LOW]

Description:

When smart contracts are deployed or their associated functions are invoked, the execution of these operations always consumes a certain quantity of gas, according to the amount of computation required to accomplish them. Modifying an unknown-size array that grows in size over time can result in a Denial-of-Service. Simply by having an excessively huge array, users can exceed the gas limit, therefore preventing the transaction from ever succeeding.

Code:

Listing 8: NftLocker.sol

```
function lock(address[] memory nftCollections, uint256[] memory tokenIds
    ↪ ) external {
    for (uint256 i = 0; i < nftCollections.length; i++) {
      StakedNFT[] storage _stakedNfts = stakedNfts[_msgSender()][
          ↪ nftCollections[i]];

      _stakedNfts.push(StakedNFT({
        lockedTs: block.timestamp,
        tokenId: tokenIds[i]
      }));

      userNfts[nftCollections[i]][tokenIds[i]] = _stakedNfts.length - 1;

      IERC721(nftCollections[i]).safeTransferFrom(
        _msgSender(),
        address(this),
        tokenIds[i]
      );
    }
}
```

Risk Level:

Likelihood – 2
Impact - 2

Recommendation:

Avoid actions that involve looping across the entire data structure. If you really must loop over an array of unknown size, arrange for it to consume many blocs and thus multiple transactions.

Status - Acknowledged

The NFHeroes team has acknowledged the risk.

C.3 Renounce Ownership [LOW]

Description:

Typically, the contract's owner is the account that deploys the contract. As a result, the owner can perform certain privileged activities. The renounceOwnership function is used in smart contracts to renounce ownership. However, if the contract's ownership has never been transferred before renouncing it, it will never have an Owner, which may result in a denial of service.

Code:

Listing 9: NftLocker.sol

```
contract NftLocker is Initializable, UUPSUpgradeable, OwnableUpgradeable
    ↪ , IERC721Receiver {
```

Risk Level:

Likelihood – 1
Impact - 2

Recommendation:

It is advised that the Owner cannot call renounceOwnership without first transferring ownership to a different address. Additionally, if a multi-signature wallet is utilized, executing the renounceOwnership method will require two or more users to sign the transaction. Alternatively, the Renounce Ownership functionality can be disabled by overriding it.

Status - Fixed

The NFHeroes team has fixed the issue by overriding the renounceOwnership() function in order to disable the renounce ownership functionality.

4 Best Practices

BP.1 Remove The Hardhat Console In Production

Description:

Remove the hardhat console import before deploying the contract in production.

Code:

Listing 10: CreditVault.sol

```
import "hardhat/console.sol";
```

5 Static Analysis (Slither)

Description:

ShellBoxes expanded the coverage of the specific contract areas using automated testing methodologies. Slither, a Solidity static analysis framework, was one of the tools used. Slither was run on all-scoped contracts in both text and binary formats. This tool can be used to test mathematical relationships between Solidity instances statically and variables that allow for the detection of errors or inconsistent usage of the contracts' APIs throughout the entire codebase.

Results:

```
//+CreditVault.sol
ERC1967UpgradeUpgradeable._functionDelegateCall(address,bytes) (
    ↪ node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/
    ↪ ERC1967UpgradeUpgradeable.sol#198-204) uses delegatecall to a
    ↪ input-controlled function id
        - (success,returndata) = target.delegatecall(data) (node_modules/
          ↪ @openzeppelin/contracts-upgradeable/proxy/ERC1967/
          ↪ ERC1967UpgradeUpgradeable.sol#202)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #controlled-delegatecall

CreditVault.tokenRescue(IERC20,address,uint256) (contracts/CreditVault.
    ↪ sol#176-182) ignores return value by token.transfer(recipient,
    ↪ amount) (contracts/CreditVault.sol#181)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #unchecked-transfer

CreditVault (contracts/CreditVault.sol#16-196) is an upgradeable
    ↪ contract that does not protect its initiliaze functions:
    ↪ CreditVault.initialize(IERC20Upgradeable,address,address) (
    ↪ contracts/CreditVault.sol#34-46). Anyone can delete the contract
    ↪ with: UUPSUpgradeable.upgradeTo(address) (node_modules/
    ↪ @openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.
    ↪ sol#72-75)UUPSUpgradeable.upgradeToAndCall(address,bytes) (
    ↪ node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/
    ↪ UUPSUpgradeable.sol#85-88)Reference: https://github.com/crytic/
    ↪ slither/wiki/Detector-Documentation#unprotected-upgradeable-
    ↪ contract

ERC1967UpgradeUpgradeable._upgradeToAndCallUUPS(address,bytes,bool).slot
    ↪  (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/
    ↪ ERC1967UpgradeUpgradeable.sol#98) is a local variable never
    ↪ initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #uninitialized-local-variables

ERC1967UpgradeUpgradeable._upgradeToAndCallUUPS(address,bytes,bool) (
    ↪ node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/
    ↪ ERC1967UpgradeUpgradeable.sol#87-105) ignores return value by
    ↪ IERC1822ProxiableUpgradeable(newImplementation).proxiableUUID() (
    ↪ node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/
    ↪ ERC1967UpgradeUpgradeable.sol#98-102)
ERC721._checkOnERC721Received(address,address,uint256,bytes) (
    ↪ node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol
    ↪ #388-409) ignores return value by IERC721Receiver(to).
    ↪ onERC721Received(_msgSender(),from,tokenId,_data) (node_modules/
    ↪ @openzeppelin/contracts/token/ERC721/ERC721.sol#395-405)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #unused-return

CreditVault.initialize(IERC20Upgradeable,address,address).
    ↪ _treasuryAddress (contracts/CreditVault.sol#37) lacks a zero-
    ↪ check on :
                - treasuryAddress = _treasuryAddress (contracts/
                    ↪ CreditVault.sol#44)
```

```
CreditVault.updateAuthProvider(address)._authProvider (contracts/
   ↪ CreditVault.sol#56) lacks a zero-check on :
              - authProvider = _authProvider (contracts/CreditVault.sol
                 ↪ #57)
CreditVault.setTreasuryAddress(address)._treasuryAddress (contracts/
   ↪ CreditVault.sol#99) lacks a zero-check on :
              - treasuryAddress = _treasuryAddress (contracts/
                 ↪ CreditVault.sol#100)
CreditVault.etherRescue(address,uint256).recipient (contracts/
   ↪ CreditVault.sol#190) lacks a zero-check on :
              - (success) = address(recipient).call{value: amount}() (
                 ↪ contracts/CreditVault.sol#193)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #missing-zero-address-validation

CreditVault.claim(uint256[],uint256,uint256,address[],bytes32,bytes32,
   ↪ uint8) (contracts/CreditVault.sol#107-133) has external calls
   ↪ inside a loop: require(bool,string)(IERC721(nftCollections[i]).
   ↪ ownerOf(tokenIds[i]) == _msgSender(),nft not owned by user) (
   ↪ contracts/CreditVault.sol#125)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ /#calls-inside-a-loop

Variable 'ERC1967UpgradeUpgradeable._upgradeToAndCallUUPS(address,bytes,
   ↪ bool).slot (node_modules/@openzeppelin/contracts-upgradeable/
   ↪ proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#98)' in
   ↪ ERC1967UpgradeUpgradeable._upgradeToAndCallUUPS(address,bytes,
   ↪ bool) (node_modules/@openzeppelin/contracts-upgradeable/proxy/
   ↪ ERC1967/ERC1967UpgradeUpgradeable.sol#87-105) potentially used
   ↪ before declaration: require(bool,string)(slot ==
   ↪ _IMPLEMENTATION_SLOT,ERC1967Upgrade: unsupported proxiableUUID) (
   ↪ node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/
   ↪ ERC1967UpgradeUpgradeable.sol#99)
```

```
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).
    ↪ retval (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.
    ↪ sol#395)' in ERC721._checkOnERC721Received(address,address,
    ↪ uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721
    ↪ /ERC721.sol#388-409) potentially used before declaration: retval
    ↪ == IERC721Receiver.onERC721Received.selector (node_modules/
    ↪ @openzeppelin/contracts/token/ERC721/ERC721.sol#396)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).
    ↪ reason (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.
    ↪ sol#397)' in ERC721._checkOnERC721Received(address,address,
    ↪ uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721
    ↪ /ERC721.sol#388-409) potentially used before declaration: reason.
    ↪ length == 0 (node_modules/@openzeppelin/contracts/token/ERC721/
    ↪ ERC721.sol#398)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).
    ↪ reason (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.
    ↪ sol#397)' in ERC721._checkOnERC721Received(address,address,
    ↪ uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721
    ↪ /ERC721.sol#388-409) potentially used before declaration: revert(
    ↪ uint256,uint256)(32 + reason,mload(uint256)(reason)) (
    ↪ node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#402)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #pre-declaration-usage-of-local-variables

Reentrancy in CreditVault.claim(uint256[],uint256,uint256,address[],
    ↪ bytes32,bytes32,uint8) (contracts/CreditVault.sol#107-133):
       External calls:
       - loreToken.safeTransferFrom(treasuryAddress,_msgSender(),amount)
           ↪  (contracts/CreditVault.sol#131)
       Event emitted after the call(s):
       - Claimed(_msgSender(),nftCollections,tokenIds,amount) (contracts
           ↪ /CreditVault.sol#132)
Reentrancy in CreditVault.deposit(uint256) (contracts/CreditVault.sol
    ↪ #164-167):
       External calls:
       - loreToken.safeTransferFrom(_msgSender(),treasuryAddress,amount)
          ↪  (contracts/CreditVault.sol#165)
       Event emitted after the call(s):
       - Deposited(_msgSender(),amount) (contracts/CreditVault.sol#166)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #reentrancy-vulnerabilities-3

AddressUpgradeable.verifyCallResult(bool,bytes,string) (node_modules/
   ↪ @openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol
   ↪ #174-194) uses assembly
       - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/
          ↪ utils/AddressUpgradeable.sol#186-189)
StorageSlotUpgradeable.getAddressSlot(bytes32) (node_modules/
   ↪ @openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.
   ↪ sol#52-56) uses assembly
       - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/
          ↪ utils/StorageSlotUpgradeable.sol#53-55)
StorageSlotUpgradeable.getBooleanSlot(bytes32) (node_modules/
   ↪ @openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.
   ↪ sol#61-65) uses assembly
       - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/
          ↪ utils/StorageSlotUpgradeable.sol#62-64)
StorageSlotUpgradeable.getBytes32Slot(bytes32) (node_modules/
   ↪ @openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.
   ↪ sol#70-74) uses assembly
       - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/
          ↪ utils/StorageSlotUpgradeable.sol#71-73)
StorageSlotUpgradeable.getUint256Slot(bytes32) (node_modules/
   ↪ @openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.
   ↪ sol#79-83) uses assembly
       - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/
          ↪ utils/StorageSlotUpgradeable.sol#80-82)
```

```
ERC721._checkOnERC721Received(address,address,uint256,bytes) (
   ↪ node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol
   ↪ #388-409) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/token/ERC721/
           ↪ ERC721.sol#401-403)
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: ['0.8.11', '^0.8.0', '^0.8.1', '^0.8.2']
        - 0.8.11 (contracts/CreditVault.sol#2)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/access
           ↪ /OwnableUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/
           ↪ interfaces/draft-IERC1822Upgradeable.sol#4)
        - ^0.8.2 (node_modules/@openzeppelin/contracts-upgradeable/proxy/
           ↪ ERC1967/ERC1967UpgradeUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/
           ↪ beacon/IBeaconUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/
           ↪ utils/Initializable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/
           ↪ utils/UUPSUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/
           ↪ security/PausableUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/
           ↪ security/ReentrancyGuardUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/
           ↪ ERC20/IERC20Upgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/
          ↪ ERC20/utils/SafeERC20Upgradeable.sol#4)
        - ^0.8.1 (node_modules/@openzeppelin/contracts-upgradeable/utils/
          ↪ AddressUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/
          ↪ ContextUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/
          ↪ StorageSlotUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.
          ↪ sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20
          ↪ .sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/
          ↪ extensions/IERC20Metadata.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/IERC721Metadata.sol#4)
        - ^0.8.1 (node_modules/@openzeppelin/contracts/utils/Address.sol
          ↪ #4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol
          ↪ #4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Strings.sol
          ↪ #4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/
          ↪ introspection/ERC165.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/
          ↪ introspection/IERC165.sol#4)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #different-pragma-directives-are-used
```

```
Address.functionCall(address,bytes) (node_modules/@openzeppelin/
    ↪ contracts/utils/Address.sol#85-87) is never used and should be
    ↪ removed
Address.functionCall(address,bytes,string) (node_modules/@openzeppelin/
    ↪ contracts/utils/Address.sol#95-101) is never used and should be
    ↪ removed
Address.functionCallWithValue(address,bytes,uint256) (node_modules/
    ↪ @openzeppelin/contracts/utils/Address.sol#114-120) is never used
    ↪ and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (
    ↪ node_modules/@openzeppelin/contracts/utils/Address.sol#128-139)
    ↪ is never used and should be removed
Address.functionDelegateCall(address,bytes) (node_modules/@openzeppelin/
    ↪ contracts/utils/Address.sol#174-176) is never used and should be
    ↪ removed
Address.functionDelegateCall(address,bytes,string) (node_modules/
    ↪ @openzeppelin/contracts/utils/Address.sol#184-193) is never used
    ↪ and should be removed
Address.functionStaticCall(address,bytes) (node_modules/@openzeppelin/
    ↪ contracts/utils/Address.sol#147-149) is never used and should be
    ↪ removed
Address.functionStaticCall(address,bytes,string) (node_modules/
    ↪ @openzeppelin/contracts/utils/Address.sol#157-166) is never used
    ↪ and should be removed
Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts
    ↪ /utils/Address.sol#60-65) is never used and should be removed
Address.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/
    ↪ contracts/utils/Address.sol#201-221) is never used and should be
    ↪ removed
AddressUpgradeable.functionCall(address,bytes) (node_modules/
    ↪ @openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol
    ↪ #85-87) is never used and should be removed
```

```
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (
    ↪ node_modules/@openzeppelin/contracts-upgradeable/utils/
    ↪ AddressUpgradeable.sol#114-120) is never used and should be
    ↪ removed
AddressUpgradeable.functionStaticCall(address,bytes) (node_modules/
    ↪ @openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol
    ↪ #147-149) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes,string) (
    ↪ node_modules/@openzeppelin/contracts-upgradeable/utils/
    ↪ AddressUpgradeable.sol#157-166) is never used and should be
    ↪ removed
AddressUpgradeable.sendValue(address,uint256) (node_modules/
    ↪ @openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.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
ContextUpgradeable.__Context_init() (node_modules/@openzeppelin/
    ↪ contracts-upgradeable/utils/ContextUpgradeable.sol#18-19) is
    ↪ never used and should be removed
ContextUpgradeable.__Context_init_unchained() (node_modules/
    ↪ @openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol
    ↪ #21-22) is never used and should be removed
ContextUpgradeable._msgData() (node_modules/@openzeppelin/contracts-
    ↪ upgradeable/utils/ContextUpgradeable.sol#27-29) is never used and
    ↪  should be removed
ERC1967UpgradeUpgradeable.__ERC1967Upgrade_init() (node_modules/
    ↪ @openzeppelin/contracts-upgradeable/proxy/ERC1967/
    ↪ ERC1967UpgradeUpgradeable.sol#21-22) is never used and should be
    ↪ removed
ERC1967UpgradeUpgradeable.__ERC1967Upgrade_init_unchained() (
    ↪ node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/
    ↪ ERC1967UpgradeUpgradeable.sol#24-25) is never used and should be
    ↪ removed
```

```
ERC1967UpgradeUpgradeable._changeAdmin(address) (node_modules/
    ↪ @openzeppelin/contracts-upgradeable/proxy/ERC1967/
    ↪ ERC1967UpgradeUpgradeable.sol#139-142) is never used and should
    ↪ be removed
ERC1967UpgradeUpgradeable._getAdmin() (node_modules/@openzeppelin/
    ↪ contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol
    ↪ #122-124) is never used and should be removed
ERC1967UpgradeUpgradeable._getBeacon() (node_modules/@openzeppelin/
    ↪ contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol
    ↪ #158-160) is never used and should be removed
ERC1967UpgradeUpgradeable._setAdmin(address) (node_modules/@openzeppelin
    ↪ /contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.
    ↪ sol#129-132) is never used and should be removed
ERC1967UpgradeUpgradeable._setBeacon(address) (node_modules/
    ↪ @openzeppelin/contracts-upgradeable/proxy/ERC1967/
    ↪ ERC1967UpgradeUpgradeable.sol#165-172) is never used and should
    ↪ be removed
ERC1967UpgradeUpgradeable._upgradeBeaconToAndCall(address,bytes,bool) (
    ↪ node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/
    ↪ ERC1967UpgradeUpgradeable.sol#180-190) is never used and should
    ↪ be removed
ERC20._burn(address,uint256) (node_modules/@openzeppelin/contracts/token
    ↪ /ERC20/ERC20.sol#280-295) is never used and should be removed
ERC20._mint(address,uint256) (node_modules/@openzeppelin/contracts/token
    ↪ /ERC20/ERC20.sol#257-267) is never used and should be removed
ERC721._burn(uint256) (node_modules/@openzeppelin/contracts/token/ERC721
    ↪ /ERC721.sol#304-318) is never used and should be removed
ERC721._mint(address,uint256) (node_modules/@openzeppelin/contracts/
    ↪ token/ERC721/ERC721.sol#280-292) is never used and should be
    ↪ removed
ERC721._safeMint(address,uint256) (node_modules/@openzeppelin/contracts/
    ↪ token/ERC721/ERC721.sol#248-250) is never used and should be
    ↪ removed
```

```
ERC721._safeMint(address,uint256,bytes) (node_modules/@openzeppelin/
    ↪ contracts/token/ERC721/ERC721.sol#256-266) is never used and
    ↪ should be removed
SafeERC20Upgradeable.safeApprove(IERC20Upgradeable,address,uint256) (
    ↪ node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/
    ↪ utils/SafeERC20Upgradeable.sol#45-58) is never used and should be
    ↪  removed
SafeERC20Upgradeable.safeDecreaseAllowance(IERC20Upgradeable,address,
    ↪ uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/
    ↪ ERC20/utils/SafeERC20Upgradeable.sol#69-80) is never used and
    ↪ should be removed
SafeERC20Upgradeable.safeIncreaseAllowance(IERC20Upgradeable,address,
    ↪ uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/
    ↪ ERC20/utils/SafeERC20Upgradeable.sol#60-67) is never used and
    ↪ should be removed
SafeERC20Upgradeable.safeTransfer(IERC20Upgradeable,address,uint256) (
    ↪ node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/
    ↪ utils/SafeERC20Upgradeable.sol#21-27) is never used and should be
    ↪  removed
StorageSlotUpgradeable.getBytes32Slot(bytes32) (node_modules/
    ↪ @openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.
    ↪ sol#70-74) is never used and should be removed
StorageSlotUpgradeable.getUint256Slot(bytes32) (node_modules/
    ↪ @openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.
    ↪ sol#79-83) is never used and should be removed
Strings.toHexString(uint256) (node_modules/@openzeppelin/contracts/utils
    ↪ /Strings.sol#40-51) is never used and should be removed
Strings.toHexString(uint256,uint256) (node_modules/@openzeppelin/
    ↪ contracts/utils/Strings.sol#56-66) is never used and should be
    ↪ removed
UUPSUpgradeable.__UUPSUpgradeable_init() (node_modules/@openzeppelin/
    ↪ contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol#23-24) is
    ↪ never used and should be removed
```

```
UUPSUpgradeable.__UUPSUpgradeable_init_unchained() (node_modules/
    ↪ @openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.
    ↪ sol#26-27) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #dead-code

Pragma version0.8.11 (contracts/CreditVault.sol#2) necessitates a
    ↪ version too recent to be trusted. Consider deploying with
    ↪ 0.6.12/0.7.6/0.8.7
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/
    ↪ access/OwnableUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/
    ↪ interfaces/draft-IERC1822Upgradeable.sol#4) allows old versions
Pragma version^0.8.2 (node_modules/@openzeppelin/contracts-upgradeable/
    ↪ proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#4) allows old
    ↪ versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/
    ↪ proxy/beacon/IBeaconUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/
    ↪ proxy/utils/Initializable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/
    ↪ proxy/utils/UUPSUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/
    ↪ security/PausableUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/
    ↪ security/ReentrancyGuardUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/
    ↪ token/ERC20/IERC20Upgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/
    ↪ token/ERC20/utils/SafeERC20Upgradeable.sol#4) allows old versions
Pragma version^0.8.1 (node_modules/@openzeppelin/contracts-upgradeable/
    ↪ utils/AddressUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/
    ↪ utils/ContextUpgradeable.sol#4) allows old versions
```

```
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/
    ↪ utils/StorageSlotUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/
    ↪ ERC20.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/IERC20Metadata.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/IERC721Metadata.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-0.8.11 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #incorrect-versions-of-solidity

PausableUpgradeable.__gap (node_modules/@openzeppelin/contracts-
    ↪ upgradeable/security/PausableUpgradeable.sol#102) is never used
    ↪ in CreditVault (contracts/CreditVault.sol#16-196)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #unused-state-variable
contracts/CreditVault.sol analyzed (26 contracts with 75 detectors), 96
   ↪ result(s) found

// +NftLocker.sol
ERC1967UpgradeUpgradeable._functionDelegateCall(address,bytes) (
   ↪ node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/
   ↪ ERC1967UpgradeUpgradeable.sol#198-204) uses delegatecall to a
   ↪ input-controlled function id
       - (success,returndata) = target.delegatecall(data) (node_modules/
          ↪ @openzeppelin/contracts-upgradeable/proxy/ERC1967/
          ↪ ERC1967UpgradeUpgradeable.sol#202)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #controlled-delegatecall

Reentrancy in NftLocker.unlock(address[],uint256[]) (contracts/NftLocker
   ↪ .sol#60-74):
       External calls:
       - IERC721(nftCollections[i]).safeTransferFrom(address(this),
          ↪ _msgSender(),tokenIds[i]) (contracts/NftLocker.sol#65-69)
       State variables written after the call(s):
       - deleteStakedNfts(nftCollections[i],userNfts[nftCollections[i]][
          ↪ tokenIds[i]],_stakedNfts) (contracts/NftLocker.sol#71)
             - userNfts[nftCollection][lastStakedNFT.tokenId] = index (
                ↪ contracts/NftLocker.sol#55)
       - delete userNfts[nftCollections[i]][tokenIds[i]] (contracts/
          ↪ NftLocker.sol#72)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #reentrancy-vulnerabilities-1

ERC1967UpgradeUpgradeable._upgradeToAndCallUUPS(address,bytes,bool).slot
   ↪  (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/
   ↪ ERC1967UpgradeUpgradeable.sol#98) is a local variable never
```

```
    ↪ initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #uninitialized-local-variables

ERC1967UpgradeUpgradeable._upgradeToAndCallUUPS(address,bytes,bool) (
    ↪ node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/
    ↪ ERC1967UpgradeUpgradeable.sol#87-105) ignores return value by
    ↪ IERC1822ProxiableUpgradeable(newImplementation).proxiableUUID() (
    ↪ node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/
    ↪ ERC1967UpgradeUpgradeable.sol#98-102)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #unused-return

NftLocker.lock(address[],uint256[]) (contracts/NftLocker.sol#29-46) has
    ↪ external calls inside a loop: IERC721(nftCollections[i]).
    ↪ safeTransferFrom(_msgSender(),address(this),tokenIds[i]) (
    ↪ contracts/NftLocker.sol#40-44)
NftLocker.unlock(address[],uint256[]) (contracts/NftLocker.sol#60-74)
    ↪ has external calls inside a loop: IERC721(nftCollections[i]).
    ↪ safeTransferFrom(address(this),_msgSender(),tokenIds[i]) (
    ↪ contracts/NftLocker.sol#65-69)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ /#calls-inside-a-loop

Variable 'ERC1967UpgradeUpgradeable._upgradeToAndCallUUPS(address,bytes,
    ↪ bool).slot (node_modules/@openzeppelin/contracts-upgradeable/
    ↪ proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#98)' in
    ↪ ERC1967UpgradeUpgradeable._upgradeToAndCallUUPS(address,bytes,
    ↪ bool) (node_modules/@openzeppelin/contracts-upgradeable/proxy/
    ↪ ERC1967/ERC1967UpgradeUpgradeable.sol#87-105) potentially used
    ↪ before declaration: require(bool,string)(slot ==
    ↪ _IMPLEMENTATION_SLOT,ERC1967Upgrade: unsupported proxiableUUID) (
    ↪ node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/
    ↪ ERC1967UpgradeUpgradeable.sol#99)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #pre-declaration-usage-of-local-variables

AddressUpgradeable.verifyCallResult(bool,bytes,string) (node_modules/
   ↪ @openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol
   ↪ #174-194) uses assembly
       - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/
          ↪ utils/AddressUpgradeable.sol#186-189)
StorageSlotUpgradeable.getAddressSlot(bytes32) (node_modules/
   ↪ @openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.
   ↪ sol#52-56) uses assembly
       - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/
          ↪ utils/StorageSlotUpgradeable.sol#53-55)
StorageSlotUpgradeable.getBooleanSlot(bytes32) (node_modules/
   ↪ @openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.
   ↪ sol#61-65) uses assembly
       - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/
          ↪ utils/StorageSlotUpgradeable.sol#62-64)
StorageSlotUpgradeable.getBytes32Slot(bytes32) (node_modules/
   ↪ @openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.
   ↪ sol#70-74) uses assembly
       - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/
          ↪ utils/StorageSlotUpgradeable.sol#71-73)
StorageSlotUpgradeable.getUint256Slot(bytes32) (node_modules/
   ↪ @openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.
   ↪ sol#79-83) uses assembly
       - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/
          ↪ utils/StorageSlotUpgradeable.sol#80-82)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #assembly-usage

Different versions of Solidity are used:
       - Version used: ['0.8.11', '^0.8.0', '^0.8.1', '^0.8.2']
       - 0.8.11 (contracts/NftLocker.sol#2)
```

```
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/access
          ↪ /OwnableUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/
          ↪ interfaces/draft-IERC1822Upgradeable.sol#4)
        - ^0.8.2 (node_modules/@openzeppelin/contracts-upgradeable/proxy/
          ↪ ERC1967/ERC1967UpgradeUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/
          ↪ beacon/IBeaconUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/
          ↪ utils/Initializable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/
          ↪ utils/UUPSUpgradeable.sol#4)
        - ^0.8.1 (node_modules/@openzeppelin/contracts-upgradeable/utils/
          ↪ AddressUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/
          ↪ ContextUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/
          ↪ StorageSlotUpgradeable.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/utils/
          ↪ introspection/IERC165.sol#4)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #different-pragma-directives-are-used

AddressUpgradeable.functionCall(address,bytes) (node_modules/
   ↪ @openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol
   ↪ #85-87) is never used and should be removed
AddressUpgradeable.functionCall(address,bytes,string) (node_modules/
   ↪ @openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol
   ↪ #95-101) is never used and should be removed
```

```
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (
   ↪ node_modules/@openzeppelin/contracts-upgradeable/utils/
   ↪ AddressUpgradeable.sol#114-120) is never used and should be
   ↪ removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (
   ↪ node_modules/@openzeppelin/contracts-upgradeable/utils/
   ↪ AddressUpgradeable.sol#128-139) is never used and should be
   ↪ removed
AddressUpgradeable.functionStaticCall(address,bytes) (node_modules/
   ↪ @openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol
   ↪ #147-149) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes,string) (
   ↪ node_modules/@openzeppelin/contracts-upgradeable/utils/
   ↪ AddressUpgradeable.sol#157-166) is never used and should be
   ↪ removed
AddressUpgradeable.sendValue(address,uint256) (node_modules/
   ↪ @openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol
   ↪ #60-65) is never used and should be removed
ContextUpgradeable.__Context_init() (node_modules/@openzeppelin/
   ↪ contracts-upgradeable/utils/ContextUpgradeable.sol#18-19) is
   ↪ never used and should be removed
ContextUpgradeable.__Context_init_unchained() (node_modules/
   ↪ @openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol
   ↪ #21-22) is never used and should be removed
ContextUpgradeable._msgData() (node_modules/@openzeppelin/contracts-
   ↪ upgradeable/utils/ContextUpgradeable.sol#27-29) is never used and
   ↪  should be removed
ERC1967UpgradeUpgradeable.__ERC1967Upgrade_init() (node_modules/
   ↪ @openzeppelin/contracts-upgradeable/proxy/ERC1967/
   ↪ ERC1967UpgradeUpgradeable.sol#21-22) is never used and should be
   ↪ removed
ERC1967UpgradeUpgradeable.__ERC1967Upgrade_init_unchained() (
   ↪ node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/
   ↪ ERC1967UpgradeUpgradeable.sol#24-25) is never used and should be
   ↪ removed
ERC1967UpgradeUpgradeable._changeAdmin(address) (node_modules/
   ↪ @openzeppelin/contracts-upgradeable/proxy/ERC1967/
   ↪ ERC1967UpgradeUpgradeable.sol#139-142) is never used and should
   ↪ be removed
ERC1967UpgradeUpgradeable._getAdmin() (node_modules/@openzeppelin/
   ↪ contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol
   ↪ #122-124) is never used and should be removed
ERC1967UpgradeUpgradeable._getBeacon() (node_modules/@openzeppelin/
   ↪ contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol
   ↪ #158-160) is never used and should be removed
ERC1967UpgradeUpgradeable._setAdmin(address) (node_modules/@openzeppelin
   ↪ /contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.
   ↪ sol#129-132) is never used and should be removed
ERC1967UpgradeUpgradeable._setBeacon(address) (node_modules/
   ↪ @openzeppelin/contracts-upgradeable/proxy/ERC1967/
   ↪ ERC1967UpgradeUpgradeable.sol#165-172) is never used and should
   ↪ be removed
ERC1967UpgradeUpgradeable._upgradeBeaconToAndCall(address,bytes,bool) (
   ↪ node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/
   ↪ ERC1967UpgradeUpgradeable.sol#180-190) is never used and should
   ↪ be removed
Initializable._isConstructor() (node_modules/@openzeppelin/contracts-
   ↪ upgradeable/proxy/utils/Initializable.sol#77-79) is never used
   ↪ and should be removed
OwnableUpgradeable.__Ownable_init() (node_modules/@openzeppelin/
   ↪ contracts-upgradeable/access/OwnableUpgradeable.sol#29-31) is
   ↪ never used and should be removed
OwnableUpgradeable.__Ownable_init_unchained() (node_modules/
   ↪ @openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol
   ↪ #33-35) is never used and should be removed
StorageSlotUpgradeable.getBytes32Slot(bytes32) (node_modules/
   ↪ @openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.
   ↪ sol#70-74) is never used and should be removed
```

```
StorageSlotUpgradeable.getUint256Slot(bytes32) (node_modules/
    ↪ @openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.
    ↪ sol#79-83) is never used and should be removed
UUPSUpgradeable.__UUPSUpgradeable_init() (node_modules/@openzeppelin/
    ↪ contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol#23-24) is
    ↪ never used and should be removed
UUPSUpgradeable.__UUPSUpgradeable_init_unchained() (node_modules/
    ↪ @openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.
    ↪ sol#26-27) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #dead-code

Pragma version0.8.11 (contracts/NftLocker.sol#2) necessitates a version
    ↪ too recent to be trusted. Consider deploying with
    ↪ 0.6.12/0.7.6/0.8.7
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/
    ↪ access/OwnableUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/
    ↪ interfaces/draft-IERC1822Upgradeable.sol#4) allows old versions
Pragma version^0.8.2 (node_modules/@openzeppelin/contracts-upgradeable/
    ↪ proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#4) allows old
    ↪ versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/
    ↪ proxy/beacon/IBeaconUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/
    ↪ proxy/utils/Initializable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/
    ↪ proxy/utils/UUPSUpgradeable.sol#4) allows old versions
Pragma version^0.8.1 (node_modules/@openzeppelin/contracts-upgradeable/
    ↪ utils/AddressUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/
    ↪ utils/ContextUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/
    ↪ utils/StorageSlotUpgradeable.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/utils/
    ↪ introspection/IERC165.sol#4) allows old versions
solc-0.8.11 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #incorrect-versions-of-solidity

OwnableUpgradeable.__gap (node_modules/@openzeppelin/contracts-
    ↪ upgradeable/access/OwnableUpgradeable.sol#87) is never used in
    ↪ NftLocker (contracts/NftLocker.sol#10-84)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #unused-state-variable
contracts/NftLocker.sol analyzed (13 contracts with 75 detectors), 53
    ↪ result(s) found
//LoreToken.sol

LoreToken.tokenRescue(IERC20,address,uint256) (contracts/LoreToken.sol
    ↪ #30-36) ignores return value by token.transfer(recipient,amount)
    ↪ (contracts/LoreToken.sol#35)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #unchecked-transfer

LoreToken.constructor(address)._treasuryAddress (contracts/LoreToken.sol
    ↪ #13) lacks a zero-check on :
                - treasuryAddress = _treasuryAddress (contracts/LoreToken.
                    ↪ sol#14)
LoreToken.setTreasuryAddress(address)._treasuryAddress (contracts/
    ↪ LoreToken.sol#19) lacks a zero-check on :
                - treasuryAddress = _treasuryAddress (contracts/LoreToken.
                    ↪ sol#20)
```

```
LoreToken.etherRescue(address,uint256).recipient (contracts/LoreToken.
   ↪ sol#44) lacks a zero-check on :
              - (success) = address(recipient).call{value: amount}() (
                 ↪ contracts/LoreToken.sol#47)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #missing-zero-address-validation

Different versions of Solidity are used:
       - Version used: ['0.8.11', '^0.8.0']
       - 0.8.11 (contracts/LoreToken.sol#2)
       - ^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol
          ↪ #4)
       - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.
          ↪ sol#4)
       - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20
          ↪ .sol#4)
       - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/
          ↪ extensions/ERC20Burnable.sol#4)
       - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/
          ↪ extensions/IERC20Metadata.sol#4)
       - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol
          ↪ #4)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #different-pragma-directives-are-used

Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.
   ↪ sol#21-23) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #dead-code

Pragma version0.8.11 (contracts/LoreToken.sol#2) necessitates a version
   ↪ too recent to be trusted. Consider deploying with
   ↪ 0.6.12/0.7.6/0.8.7
```

```
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/
    ↪ Ownable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/
    ↪ ERC20.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/ERC20Burnable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/
    ↪ extensions/IERC20Metadata.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context
    ↪ .sol#4) allows old versions
solc-0.8.11 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #incorrect-versions-of-solidity
contracts/LoreToken.sol analyzed (7 contracts with 75 detectors), 14
    ↪ result(s) found
```

Conclusion:

Most of the vulnerabilities found by the analysis have already been addressed by the smart contract code review.

6 Conclusion

In this audit, we examined the design and implementation of NFHeroes contract and discovered several issues of varying severity. NFHeroes team addressed all the issues raised in the initial report and implemented the necessary fixes.

The present code base is well-structured and ready for the mainnet.

SHELLBOXES

For a Contract Audit, contact us at contact@shellboxes.com