// HALBORN

Seascape - Riverboat NFT

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **November 1st, 2021 - November 12th, 2021**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	11/12/2021	Michal Bazyli
0.2	Document Edits	11/12/2021	Michal Bazyli
0.3	Final Draft	11/12/2021	Michal Bazyli
0.4	Final Draft Review	11/15/2021	Gabi Urrutia
1.0	Remediation Plan	11/17/2021	Michal Bazyli
1.1	Remediation Plan Review	11/18/2021	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
Michal Bazyli	Halborn	michal.bazyli@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Seascape engaged Halborn to conduct a security audit on their Riverboat NFT beginning on November 1st, 2021 and ending on November 12th, 2021. The security assessment was scoped to the smart contracts provided in the Github repository blocklords/seascape-smartcontracts/tree/nft-market/contracts/riverboat

1.2 AUDIT SUMMARY

The team at Halborn was provided two 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 some security risks that should be addressed by the Seascape 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
- 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. It's quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that was used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

RISK SCALE - LIKELIHOOD

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

RISK SCALE - IMPACT

5 - May cause devastating and unrecoverable impact or loss.
4 - May cause a significant level of impact or loss.
3 - May cause a partial impact or loss to many.
2 - May cause temporary impact or loss.

1 - May cause minimal or un-noticeable impact.

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

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

1.4 SCOPE

The security assessment was scoped to the following
blocklords/seascape-smartcontracts/tree/nft-market/contracts/riverboat
- LighthouseTierInterface.sol
- RiverboatFactory.sol
- RiverboatNft.sol
- Riverboat.sol
- All contracts inherited by these contracts

Commit ID: 312419847e775c73838d15af71ab877087f7a49d
Fixed Commit ID: c5fd888b95f40e3c3c60f594708887db1922b663

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	1	1	3	2

LIKELIHOOD

IMPACT

		(HAL-01)		
(HAL-05)	(HAL-04)	(HAL-02)		
		(HAL-03)		
(HAL-06) (HAL-07)				

EXECUTIVE OVERVIEW

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL01 - CONTRACT ADMIN CAN REVOKE AND RENOUNCE HIMSELF	High	SOLVED - 11/18/2021
HAL02 - IMPROPER ACCESS CONTROL POLICY	Medium	SOLVED - 11/18/2021
HAL03 - MISSING ZERO ADDRESS CHECK	Low	SOLVED - 11/18/2021
HAL04 - USAGE OF BLOCK-TIMESTAMP	Low	NOT APPLICABLE
HAL05 - FLOATING PRAGMA	Low	SOLVED - 11/18/2021
HAL06 - INTEGER OVERFLOW	Informational	ACKNOWLEDGED
HAL07 - UNUSED IMPORT	Informational	SOLVED - 11/18/2021

FINDINGS & TECH DETAILS

3.1 (HAL-01) CONTRACT ADMIN CAN REVOKE AND RENOUNCE HIMSELF - HIGH

Description:

The Owner of the contract is usually the account which deploys the contract. In the RiverboatFactory.sol smart contract, Only Admin can perform some privileged actions such as setNft, addStaticUser, removeStaticUser etc. . . , the addAdmin function is used to add an Admin role, and the renounceAdmin function is used to renounce being an Admin. It was observed that admin can revoke his role via renounceAdmin. If an admin is mistakenly renounced, administrative access would result in the contract having no admin, eliminating the ability to call privileged functions. In such a case, contracts would have to be redeployed.

PoC Steps:

- Deploy a RiverboatFactory using the owner address.
- Execute riverboatfactory.renounceAdmin() function as using the owner address.

```
>>> riverboatfactory.isAdmin(accounts[0])
True
>>> riverboatfactory.renounceAdmin()
Transaction sent: 0xe305392594a6a1d97d56fc42d01744144b7873aa71ca29cb6989f4509164aef3
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 4
  RiverboatFactory.renounceAdmin confirmed   Block: 5   Gas used: 22436 (0.19%)

<Transaction '0xe305392594a6a1d97d56fc42d01744144b7873aa71ca29cb6989f4509164aef3'>
>>> riverboatfactory.isAdmin(accounts[0])
False
>>> riverboatfactory.setNft(riverboatnft)
Transaction sent: 0xb8bf55b9fdead852dcab94d83ff5fa48f3f9c6d859f57016942f2f83e6a96aa0
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 5
  RiverboatFactory.setNft confirmed (Restricted to admins.)   Block: 6   Gas used: 23031 (0.19%)

<Transaction '0xb8bf55b9fdead852dcab94d83ff5fa48f3f9c6d859f57016942f2f83e6a96aa0'>
```

Code Location:

Listing 1: contracts/RiverboatFactory (Lines 43)

```
43      function renounceAdmin() public virtual {
44          renounceRole(DEFAULT_ADMIN_ROLE, msg.sender);
45      }
```

Risk Level:

Likelihood - 3
Impact - 5

Recommendations:

It is recommended that the contract Admin cannot call renounceAdmin without transferring the Ownership to other address before. In addition, if a multi-signature wallet is used, calling renounceAdmin function should be confirmed for two or more users.

Remediation Plan:

SOLVED: The issue was fixed in commit: c5fd888b95f40e3c3c60f594708887db1922b663

3.2 (HAL-02) IMPROPER ACCESS CONTROL POLICY - MEDIUM

Description:

In the smart contracts, implementing a correct Access Control policy is essential to maintain security and decentralization of permissions on a token. The features to mint/burn tokens and pause contracts are given by Access Control. For instance, Ownership is the most common form of Access Control. In other words, the owner of a contract (the account that deployed it by default) can do some administrative tasks on it. Nevertheless, other authorization levels are required to keep the principle of least privilege, also known as least authority. Briefly, any process, user, or program only can access to the necessary resources or information. Otherwise, the ownership role is useful in simple systems, but more complex projects require the use of more roles using Role-based access control.

The mintType function on RiverboatFactory have no restrictions and can be called by any user. An adversary could leverage this to mint NFT's instead of buying.

PoC Steps:

- Deploy a RiverboatFactory and RiverboatNft using the owner address.
- Call function setFactory and provide address of the RiverboatFactory using owner user address.
- Execute mintType function using any user address.

```
>>> riverboatfactory.isAdmin(accounts[0])
True
>>> riverboatfactory.isAdmin(accounts[1])
False
>>> riverboatfactory.mintType(accounts[1], 1, {'from':accounts[1]})
Transaction sent: 0xa4d1c4ae778733a1f120ca12108f750f855d84ab23165bb498b359fb341b99ba
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 0
  RiverboatFactory.mintType confirmed   Block: 5   Gas used: 203821 (1.70%)

<Transaction '0xa4d1c4ae778733a1f120ca12108f750f855d84ab23165bb498b359fb341b99ba'>
>>> riverboatnft.ownerOf(1)
'0x33A4622B82D4c04a53e170c638B944ce27cffce3'
```

Code Location:

Listing 2: contracts/RiverboatFactory (Lines 24)

```
24      function mintType(address _owner, uint256 _type) public
            returns(uint256) {
25          require (_type < 5, "invalid type");
26          return nft.mint(_owner, _type);
27      }
```

Risk Level:

Likelihood - 3
Impact - 3

Recommendations:

It is recommended to limit the function with an appropriate modifier, which will allow only authorized users to execute the function.

Remediation Plan:

SOLVED: The issue was fixed in commit: c5fd888b95f40e3c3c60f594708887db1922b663

3.3 (HAL-03) MISSING ZERO ADDRESS CHECK - LOW

Description:

There is no validation of the addresses anywhere in the code. Every address should be validated and checked that is different from zero. This issue is present in all the smart contracts, in the constructors and functions that use addresses as parameters.

Some code location examples:

RiverboatFactory.sol

Listing 3: RiverboatFactory.sol

```
145     function setNft(address _nft) public onlyAdmin {
146         nft = RiverboatNft(_nft);
147     }
```

RiverboatNft.sol

Listing 4: RiverboatNft.sol

```
56     function setFactory(address _factory) public onlyOwner {
57         factory = _factory;
58     }
```

Risk Level:

Likelihood - 3
Impact - 2

Recommendation:

Validate that every address input is different from zero.

Remediation Plan:

SOLVED: The issue was fixed in commit: c5fd888b95f40e3c3c60f594708887db1922b663

3.4 (HAL-04) USAGE OF BLOCK-TIMESTAMP - LOW

Description:

During a manual static review, the tester noticed the use of “now” and “block.timestamp”. The contract developers should be aware that this does not mean current time. “block.timestamp” can be influenced by miners to a certain degree, so the testers should be warned that this may have some risk if miners collude on time manipulation to influence the price oracles.

Code Location:

Listing 5: contracts/RiverboatFactory (Lines 121)

```
115             if (sessionId > 0)
116              require(isFinished(sessionId), "last session hasnt
                    finished yet");
117         require(_currencyAddress != address(0), "invalid currency
                address");
118         require(_nftAddress != address(0), "invalid nft address");
119         require(_startPrice > 0, "start price can't be 0");
120         require(_priceIncrease > 0, "price increase can't be 0");
121         require(_startTime > block.timestamp, "session should
                start in future");
122         require(_intervalDuration > 0, "interval duration can't be
                0");
123         require(_intervalsAmount > 0, "intervals amount can't be 0
                ");
124         require(_slotsAmount > 0, "slots amount can't be 0");
125     }
```

Listing 6: contracts/Riverboat

```
1 contracts/Riverboat.sol:234: uint256 _currentInterval = (now -
      sessions[_sessionId]
2 contracts/Riverboat.sol:262: if(now >= session.startTime && now <
      session
```

```
3 contracts/Riverboat.sol:274: if(now > session.startTime + session.
    intervalsAmount * session.intervalDuration)
```

Risk Level:

Likelihood - 2
Impact - 3

Recommendations:

Use block.number instead of block.timestamp or now reduce the influence of miners. If possible, It’s recommended to use Oracles.

Remediation Plan:

NOT APPLICABLE: The Seascape claims that the timescales is higher than 900 seconds.

3.5 (HAL-05) FLOATING PRAGMA - LOW

Description:

Riverboat contract Riverboat.sol, RiverboatFactory.sol and RiverboatNft.sol uses the floating pragma ^0.6.7. Contract should be deployed with the same compiler version and flags that they have been tested with thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, either an outdated compiler version that might introduce bugs that affect the contract system negatively or a pragma version too new which has not been extensively tested.

Code Location:

Listing 7: Riverboat.sol

```
1 pragma solidity ^0.6.7;
```

Listing 8: RiverboatFactory.sol

```
1 pragma solidity 0.6.7;
2
```

Listing 9: RiverboatNft.soll

```
1 // Riverboats NFT
2 // SPDX-License-Identifier: MIT
3 pragma solidity 0.6.7;
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendations:

Consider locking the pragma version with known bugs for the compiler version. When possible, do not use floating pragma in the final live deployment. Specifying a fixed compiler version ensures that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Remediation Plan:

SOLVED: The issue was fixed in commit: c5fd888b95f40e3c3c60f594708887db1922b663

3.6 (HAL-06) INTEGER OVERFLOW - INFORMATIONAL

Description:

An overflow happens when an arithmetic operation reaches the maximum size of a type. In computer programming, an integer overflow occurs when an arithmetic operation attempts to create a numeric value that is outside the range that can be represented with a given number of bits -- either larger than the maximum or lower than the minimum representable value

Code Location:

Listing 10: contracts/Riverboat (Lines 126)

```
101     function startSession(
102         address _currencyAddress,
103         address _nftAddress,
104         address _lighthouseTierAddress,
105         uint256 _startPrice,
106         uint256 _priceIncrease,
107         uint32 _startTime,
108         uint32 _intervalDuration,
109         uint32 _intervalsAmount,
110         uint32 _slotsAmount
111     )
112         external
113         onlyOwner
114     {
115         if (sessionId > 0)
116             require(isFinished(sessionId), "last session hasnt
                    finished yet");
117         require(_currencyAddress != address(0), "invalid currency
                address");
118         require(_nftAddress != address(0), "invalid nft address");
119         require(_startPrice > 0, "start price can't be 0");
120         require(_priceIncrease > 0, "price increase can't be 0");
121         require(_startTime > block.timestamp, "session should
                start in future");
```

```
122         require(_intervalDuration > 0, "interval duration can't be
                0");
123         require(_intervalsAmount > 0, "intervals amount can't be 0
                ");
124         require(_slotsAmount > 0, "slots amount can't be 0");
125
126         sessionId++;
127         sessions[sessionId] = Session(
128             _currencyAddress,
129             _nftAddress,
130             _lighthouseTierAddress,
131             _startPrice,
132             _priceIncrease,
133             _startTime,
134             _intervalDuration,
135             _intervalsAmount,
136             _slotsAmount
137         );
```

Listing 11: contracts/Riverboat

```
contracts/Riverboat.sol:252: uint256 _currentPrice = sessions[
    _sessionId].startPrice + sessions[_sessionId]
contracts/Riverboat.sol:253: .priceIncrease * _currentInterval;
contracts/Riverboat.sol:263: .startTime + session.intervalsAmount
    * session.intervalDuration){
contracts/Riverboat.sol:274 if(now > session.startTime + session.
    intervalsAmount * session.intervalDuration)
```

Risk Level:

Likelihood - 1

Impact - 1

Recommendation:

It is recommended to use vetted safe math libraries for arithmetic operations consistently throughout the smart contract system.

Remediation Plan:

ACKNOWLEDGED: Since `sessionId` is type uint256, the likelihood of reaching 10^77 is very low. Therefore, overflow is highly unlikely.

3.7 (HAL-07) UNUSED IMPORT - INFORMATIONAL

Description:

During the test, it was determined that one of the import on the contract was not used. This situation does not pose any risk in terms of security. But it is important for the readability and applicability of the code

Code Location:

Listing 12: contracts/Riverboat (Lines 9)

```
1 pragma solidity ^0.6.7;
2
3 import "./../openzeppelin/contracts/token/ERC20/IERC20.sol";
4 import "./../openzeppelin/contracts/token/ERC20/SafeERC20.sol";
5 import "./../openzeppelin/contracts/token/ERC721/IERC721.sol";
6 import "./../openzeppelin/contracts/token/ERC721/IERC721Receiver.
      sol";
7 import "./../openzeppelin/contracts/access/Ownable.sol";
8 import "./LighthouseTierInterface.sol";
9 import "./RiverboatNft.sol";
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to remove the RiverboatNft.sol import from the contract if it will not be used for any purpose.

Remediation Plan:

SOLVED: The issue was fixed in commit: c5fd888b95f40e3c3c60f594708887db1922b663

AUTOMATED TESTING

4.1 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:

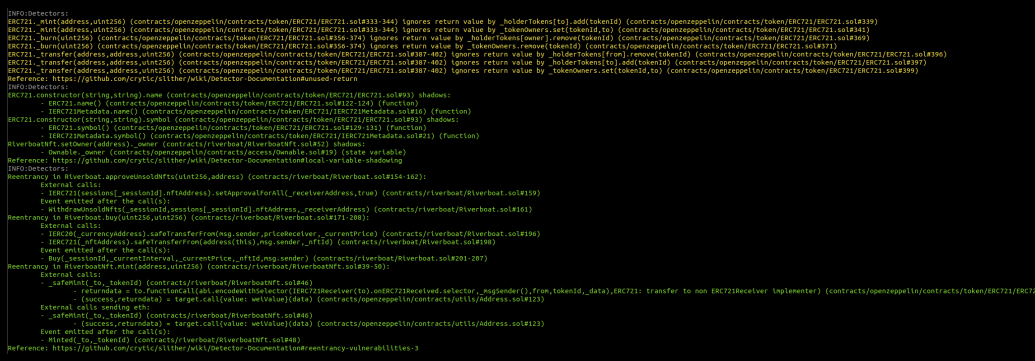

Figure 1: contracts/Riverboat.sol

All reentrancy issues were false positive. Furthermore, all the vulnerabilities found were already described on the report

```
INFO:Detectors:
Riverboat.startSession(address,address,address,uint256,uint256,uint32,uint32,uint32,uint32) (contracts/riverboat/Riverboat.sol#101-149) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(_startTime > block.timestamp,session should start in future) (contracts/riverboat/Riverboat.sol#121)
Riverboat.buy(uint256,uint256) (contracts/riverboat/Riverboat.sol#171-208) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(! nftMinters[_sessionId][_currentInterval][msg.sender],cant buy more nfts per interval) (contracts/riverboat/Riverboat.sol#179-180)
Riverboat.getCurrentInterval(uint256) (contracts/riverboat/Riverboat.sol#232-240) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(_currentInterval < sessions[_sessionId].intervalsAmount,_currentInterval > intervalsAmount, session is finished) (contracts/riverboat/Riverboat.sol#237-238)
Riverboat.isActive(uint256) (contracts/riverboat/Riverboat.sol#260-267) uses timestamp for comparisons
        Dangerous comparisons:
        - now >= session.startTime && now < session.startTime + session.intervalsAmount * session.intervalDuration (contracts/riverboat/Riverboat.sol#262-263)
Riverboat.isFinished(uint256) (contracts/riverboat/Riverboat.sol#272-277) uses timestamp for comparisons
        Dangerous comparisons:
        - now > session.startTime + session.intervalsAmount * session.intervalDuration (contracts/riverboat/Riverboat.sol#274)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (contracts/openzeppelin/contracts/utils/Address.sol#26-35) uses assembly
        - INLINE ASM (contracts/openzeppelin/contracts/utils/Address.sol#33)
Address._functionCallWithValue(address,bytes,uint256,string) (contracts/openzeppelin/contracts/utils/Address.sol#119-140) uses assembly
        - INLINE ASM (contracts/openzeppelin/contracts/utils/Address.sol#132-135)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Different versions of Solidity is used in :
        - Version used: ['0.6.7', '^0.6.0', '^0.6.2', '^0.6.7']
        - ^0.6.0 (contracts/openzeppelin/contracts/GSN/Context.sol#3)
        - ^0.6.0 (contracts/openzeppelin/contracts/access/Ownable.sol#3)
        - ^0.6.0 (contracts/openzeppelin/contracts/introspection/ERC165.sol#3)
        - ^0.6.0 (contracts/openzeppelin/contracts/introspection/IERC165.sol#3)
        - ^0.6.0 (contracts/openzeppelin/contracts/math/SafeMath.sol#3)
        - ^0.6.0 (contracts/openzeppelin/contracts/token/ERC20/IERC20.sol#3)
        - ^0.6.0 (contracts/openzeppelin/contracts/token/ERC20/SafeERC20.sol#3)
        - ^0.6.0 (contracts/openzeppelin/contracts/token/ERC721/ERC721.sol#3)
        - ^0.6.0 (contracts/openzeppelin/contracts/token/ERC721/ERC721Burnable.sol#3)
        - ^0.6.2 (contracts/openzeppelin/contracts/token/ERC721/IERC721.sol#3)
        - ^0.6.2 (contracts/openzeppelin/contracts/token/ERC721/IERC721Enumerable.sol#3)
        - ^0.6.2 (contracts/openzeppelin/contracts/token/ERC721/IERC721Metadata.sol#3)
        - ^0.6.0 (contracts/openzeppelin/contracts/token/ERC721/IERC721Receiver.sol#3)
        - ^0.6.2 (contracts/openzeppelin/contracts/utils/Address.sol#3)
        - ^0.6.0 (contracts/openzeppelin/contracts/utils/Counters.sol#3)
        - ^0.6.0 (contracts/openzeppelin/contracts/utils/EnumerableMap.sol#3)
        - ^0.6.0 (contracts/openzeppelin/contracts/utils/EnumerableSet.sol#3)
        - ^0.6.0 (contracts/openzeppelin/contracts/utils/Strings.sol#3)
        - ^0.6.7 (contracts/riverboat/LighthouseTierInterface.sol#1)
        - ^0.6.7 (contracts/riverboat/Riverboat.sol#1)
        - 0.6.7 (contracts/riverboat/RiverboatNft.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
```

Figure 2: contracts/Riverboat.sol

```
INFO:Detectors:
Pragma version^0.6.0 (contracts/openzeppelin/contracts/GSN/Context.sol#3) allows old versions
Pragma version^0.6.0 (contracts/openzeppelin/contracts/access/Ownable.sol#3) allows old versions
Pragma version^0.6.0 (contracts/openzeppelin/contracts/introspection/ERC165.sol#3) allows old versions
Pragma version^0.6.0 (contracts/openzeppelin/contracts/introspection/IERC165.sol#3) allows old versions
Pragma version^0.6.0 (contracts/openzeppelin/contracts/math/SafeMath.sol#3) allows old versions
Pragma version^0.6.0 (contracts/openzeppelin/contracts/token/ERC20/IERC20.sol#3) allows old versions
Pragma version^0.6.0 (contracts/openzeppelin/contracts/token/ERC20/SafeERC20.sol#3) allows old versions
Pragma version^0.6.0 (contracts/openzeppelin/contracts/token/ERC721/ERC721.sol#3) allows old versions
Pragma version^0.6.0 (contracts/openzeppelin/contracts/token/ERC721/ERC721Burnable.sol#3) allows old versions
Pragma version^0.6.2 (contracts/openzeppelin/contracts/token/ERC721/IERC721.sol#3) allows old versions
Pragma version^0.6.2 (contracts/openzeppelin/contracts/token/ERC721/IERC721Enumerable.sol#3) allows old versions
Pragma version^0.6.2 (contracts/openzeppelin/contracts/token/ERC721/IERC721Metadata.sol#3) allows old versions
Pragma version^0.6.0 (contracts/openzeppelin/contracts/token/ERC721/IERC721Receiver.sol#3) allows old versions
Pragma version^0.6.2 (contracts/openzeppelin/contracts/utils/Address.sol#3) allows old versions
Pragma version^0.6.0 (contracts/openzeppelin/contracts/utils/Counters.sol#3) allows old versions
Pragma version^0.6.0 (contracts/openzeppelin/contracts/utils/EnumerableMap.sol#3) allows old versions
Pragma version^0.6.0 (contracts/openzeppelin/contracts/utils/EnumerableSet.sol#3) allows old versions
Pragma version^0.6.0 (contracts/openzeppelin/contracts/utils/Strings.sol#3) allows old versions
Pragma version^0.6.7 (contracts/riverboat/LighthouseTierInterface.sol#1) allows old versions
Pragma version^0.6.7 (contracts/riverboat/Riverboat.sol#1) allows old versions
Pragma version0.6.7 (contracts/riverboat/RiverboatNft.sol#3) allows old versions
solc-0.6.7 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (contracts/openzeppelin/contracts/utils/Address.sol#53-59):
        - (success) = recipient.call{value: amount}() (contracts/openzeppelin/contracts/utils/Address.sol#57)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (contracts/openzeppelin/contracts/utils/Address.sol#119-140):
        - (success,returndata) = target.call{value: weiValue}(data) (contracts/openzeppelin/contracts/utils/Address.sol#123)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

Figure 3: contracts/Riverboat.sol

```
INFO:Detectors:
Parameter ERC721.safeTransferFrom(address,address,uint256,bytes)._data (contracts/openzeppelin/contracts/token/ERC721/ERC721.sol#245) is not in mixedCase
Parameter Riverboat.enableTrade(bool)._tradeEnabled (contracts/riverboat/Riverboat.sol#80) is not in mixedCase
Parameter Riverboat.setPriceReceiver(address)._priceReceiver (contracts/riverboat/Riverboat.sol#86) is not in mixedCase
Parameter Riverboat.startSession(address,address,address,uint256,uint256,uint32,uint32,uint32,uint32)._currencyAddress (contracts/riverboat/Riverboat.sol#102) is not in mixedCase
Parameter Riverboat.startSession(address,address,address,uint256,uint256,uint32,uint32,uint32,uint32)._nftAddress (contracts/riverboat/Riverboat.sol#103) is not in mixedCase
Parameter Riverboat.startSession(address,address,address,uint256,uint256,uint32,uint32,uint32,uint32)._lighthouseTierAddress (contracts/riverboat/Riverboat.sol#104) is not in mixedCase
Parameter Riverboat.startSession(address,address,address,uint256,uint256,uint32,uint32,uint32,uint32)._startPrice (contracts/riverboat/Riverboat.sol#105) is not in mixedCase
Parameter Riverboat.startSession(address,address,address,uint256,uint256,uint32,uint32,uint32,uint32)._priceIncrease (contracts/riverboat/Riverboat.sol#106) is not in mixedCase
Parameter Riverboat.startSession(address,address,address,uint256,uint256,uint32,uint32,uint32,uint32)._startTime (contracts/riverboat/Riverboat.sol#107) is not in mixedCase
Parameter Riverboat.startSession(address,address,address,uint256,uint256,uint32,uint32,uint32,uint32)._intervalDuration (contracts/riverboat/Riverboat.sol#108) is not in mixedCase
Parameter Riverboat.startSession(address,address,address,uint256,uint256,uint32,uint32,uint32,uint32)._intervalsAmount (contracts/riverboat/Riverboat.sol#109) is not in mixedCase
Parameter Riverboat.startSession(address,address,address,uint256,uint256,uint32,uint32,uint32,uint32)._slotsAmount (contracts/riverboat/Riverboat.sol#110) is not in mixedCase
Parameter Riverboat.approveUnsoldNfts(uint256,address)._sessionId (contracts/riverboat/Riverboat.sol#154) is not in mixedCase
Parameter Riverboat.approveUnsoldNfts(uint256,address)._receiverAddress (contracts/riverboat/Riverboat.sol#154) is not in mixedCase
Parameter Riverboat.buy(uint256,uint256)._sessionId (contracts/riverboat/Riverboat.sol#171) is not in mixedCase
Parameter Riverboat.buy(uint256,uint256)._nftId (contracts/riverboat/Riverboat.sol#171) is not in mixedCase
Parameter Riverboat.getCurrentInterval(uint256)._sessionId (contracts/riverboat/Riverboat.sol#232) is not in mixedCase
Parameter Riverboat.getCurrentPrice(uint256,uint256)._sessionId (contracts/riverboat/Riverboat.sol#246) is not in mixedCase
Parameter Riverboat.getCurrentPrice(uint256,uint256)._currentInterval (contracts/riverboat/Riverboat.sol#246) is not in mixedCase
Parameter Riverboat.isActive(uint256)._sessionId (contracts/riverboat/Riverboat.sol#260) is not in mixedCase
Parameter Riverboat.isFinished(uint256)._sessionId (contracts/riverboat/Riverboat.sol#272) is not in mixedCase
Parameter Riverboat.debbug(uint256,uint256)._sessionId (contracts/riverboat/Riverboat.sol#278) is not in mixedCase
Parameter Riverboat.debbug(uint256,uint256)._currentInterval (contracts/riverboat/Riverboat.sol#278) is not in mixedCase
Parameter RiverboatNft.mint(address,uint256)._to (contracts/riverboat/RiverboatNft.sol#39) is not in mixedCase
Parameter RiverboatNft.mint(address,uint256)._type (contracts/riverboat/RiverboatNft.sol#39) is not in mixedCase
Parameter RiverboatNft.setOwner(address)._owner (contracts/riverboat/RiverboatNft.sol#52) is not in mixedCase
Parameter RiverboatNft.setFactory(address)._factory (contracts/riverboat/RiverboatNft.sol#56) is not in mixedCase
Parameter RiverboatNft.setBaseUri(string)._uri (contracts/riverboat/RiverboatNft.sol#60) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformity-to-solidity-naming-conventions
```

Figure 4: contracts/Riverboat.sol

```
INFO:Detectors:
owner() should be declared external:
        - Ownable.owner() (contracts/openzeppelin/contracts/access/Ownable.sol#35-37)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (contracts/openzeppelin/contracts/access/Ownable.sol#54-57)
supportsInterface(bytes4) should be declared external:
        - ERC165.supportsInterface(bytes4) (contracts/openzeppelin/contracts/introspection/ERC165.sol#35-37)
balanceOf(address) should be declared external:
        - ERC721.balanceOf(address) (contracts/openzeppelin/contracts/token/ERC721/ERC721.sol#106-110)
name() should be declared external:
        - ERC721.name() (contracts/openzeppelin/contracts/token/ERC721/ERC721.sol#122-124)
symbol() should be declared external:
        - ERC721.symbol() (contracts/openzeppelin/contracts/token/ERC721/ERC721.sol#129-131)
tokenURI(uint256) should be declared external:
        - ERC721.tokenURI(uint256) (contracts/openzeppelin/contracts/token/ERC721/ERC721.sol#136-151)
baseURI() should be declared external:
        - ERC721.baseURI() (contracts/openzeppelin/contracts/token/ERC721/ERC721.sol#158-160)
tokenOfOwnerByIndex(address,uint256) should be declared external:
        - ERC721.tokenOfOwnerByIndex(address,uint256) (contracts/openzeppelin/contracts/token/ERC721/ERC721.sol#165-167)
totalSupply() should be declared external:
        - ERC721.totalSupply() (contracts/openzeppelin/contracts/token/ERC721/ERC721.sol#172-175)
tokenByIndex(uint256) should be declared external:
        - ERC721.tokenByIndex(uint256) (contracts/openzeppelin/contracts/token/ERC721/ERC721.sol#180-183)
approve(address,uint256) should be declared external:
        - ERC721.approve(address,uint256) (contracts/openzeppelin/contracts/token/ERC721/ERC721.sol#188-197)
setApprovalForAll(address,bool) should be declared external:
        - ERC721.setApprovalForAll(address,bool) (contracts/openzeppelin/contracts/token/ERC721/ERC721.sol#211-216)
transferFrom(address,address,uint256) should be declared external:
        - ERC721.transferFrom(address,address,uint256) (contracts/openzeppelin/contracts/token/ERC721/ERC721.sol#228-233)
safeTransferFrom(address,address,uint256) should be declared external:
        - ERC721.safeTransferFrom(address,address,uint256) (contracts/openzeppelin/contracts/token/ERC721/ERC721.sol#238-240)
burn(uint256) should be declared external:
        - ERC721Burnable.burn(uint256) (contracts/openzeppelin/contracts/token/ERC721/ERC721Burnable.sol#20-24)
debbug(uint256,uint256) should be declared external:
        - Riverboat.debbug(uint256,uint256) (contracts/riverboat/Riverboat.sol#278-280)
mint(address,uint256) should be declared external:
        - RiverboatNft.mint(address,uint256) (contracts/riverboat/RiverboatNft.sol#39-50)
setOwner(address) should be declared external:
        - RiverboatNft.setOwner(address) (contracts/riverboat/RiverboatNft.sol#52-54)
setFactory(address) should be declared external:
        - RiverboatNft.setFactory(address) (contracts/riverboat/RiverboatNft.sol#56-58)
setBaseUrl(string) should be declared external:
        - RiverboatNft.setBaseUrl(string) (contracts/riverboat/RiverboatNft.sol#60-62)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

Figure 5: contracts/Riverboat.sol

THANK YOU FOR CHOOSING

// HALBORN