DeSpace Smart Contract Preliminary Audit Report

Project Synopsis

Project Name	**DeSpace**
Platform	Ethereum, Solidity
Github Repo	Not Provided
Deployed Contract	Not Deployed
Total Duration	10 Days
Timeline of Audit	**25th August 2021 to 5th September 2021**

Contract Details

Total Contract(s)	7
Name of Contract Folder	Des-NFT-Main, NFT_Auction-Contract
Language	Solidity
Commit Hash	**Null**

Contract Vulnerabilities Synopsis

Issues	Open Issues	Closed Issues
Critical Severity	1	0
Medium Severity	2	0
Low Severity	6	0
Information	1	0
Total Found	10	0

Detailed Results

The contract has gone through several stages of the audit procedure that includes structural analysis, automated testing, manual code review etc.

All the issues have been explained and discussed in detail below. Along with the explanation of the issue found during the audit, the recommended way to overcome the issue or improve the code quality has also been mentioned.

A.1 Contract Name: DeSpaceAuction.sol
Contract Folder: NFT_Auction_Contracts

High Severity Issues

A.1.1 Sending Ether to External Entity with Transfer method could freeze the contract

Line no - 259

Explanation:

In ***bidWithEther(uint)*** method, an external call transferring ether to the *highest bidder* is being made. The function executes an external call before updating imperative state variables and this might lead to unwanted scenarios.?

Since the `**transfer**` method used a non-adjustable 2300 amount of forwarded gas; the transaction could run out of gas, since there is no check that highestBidder is not an EOA (externally-owned-account) when being set in line 263, & contracts can be set as payable (see here) it could be a contract with a fallback function which could perform a reentrancy attack.
Fallback functions requiring more gas than the stipend could otherwise freeze a contract.

```
255                 msg.value == amount,
256                 "Error: must bid 10 percent more than previous bid"
257             );
258             //return ether to the prevous highest bidder
259             auction.highestBidder.transfer(auction.highestBidAmount);
260         }
261
```

Recommendation:

Pull Over Push pattern can be used, which lets the users request payments instead of sending them, in order to avoid unexpected behavior. This can be done by storing the remaining user balances in a mapping (with the complaint token addresses) and adding another function for users to withdraw their funds.

Medium Severity Issues

A.1.2 Better error handling in case token is not complaint with registry

Line no - 701

Explanation:

In function ***_nextBigAmountToken***, if this is the first bid, (i.e. current == 0) there is no check if the complaint token used is supported by the registry (i.e. its chainlink feed exists and is stored in the registry)
This would result in `decimals` and `ethPerToken` being returned as **ZERO** and line 701 will trigger the INVALID_OPCODE (instead of revert since solidity 0.8 see here) because of **division by zero**.

```
696             if (current == 0) {
697                 (,uint8 decimals) = registry.getProxy(_compliantToken);
698                 uint ethPerToken = _getThePrice(_compliantToken);
699                 return (
700                 //get the equivilent based on chainlink oracle price and token decimal
701                     ((10 ** uint(decimals)) * initialBidAmount) / ethPerToken
702                 );
```

Recommendation:

It is recommended to add a require statement after initializing ethPerToken on line 698, such as

```
require(ethPerToken != 0, "Error: _complaintToken not supported");
```

This ensures adequate error messages being forwarded to users.

A.1.3 Return Value of an External Call is never used Effectively

Line no - 325, 327, 345, 349

Explanation:

The external calls made in the above-mentioned lines do return a boolean value that indicates whether or not the external call made was successful.
These boolean return values can be used in the function as a check to ensure that the further execution of the function is only allowed if the external is successfully made.

However, during the automated testing of the **Auction** contract, it was found that the protocol never uses these returns adequately.
The following functions avoid the return values of external calls:
a. ***closeBid()***
b. ***bidWithToken()***

Recommendation:
Effective use of all the return values from external calls must be ensured within the contract.

Low Severity Issues

A.1.4 Violation of Check_Effects_Interaction Pattern

Line no - 267, 359, 429, 451,

Explanation:

As per the Check_Effects_Interaction Pattern in Solidity, external calls should be made at the very end of the function and event emission, as well as any state variable modification, must be done before the external call is made.

However, the following functions in the DeSpaceAuction contract emit events as well as change imperative state variables after the external call has been made at the line number mentioned above:

- ***bidWithEther()***
- ***bidWithToken()***
- ***closeBid()***

Recommendation:

Check Effects Interaction Pattern must be followed while implementing external calls in a function.

A.1.5 No minimum or maximum threshold is validated for *_newFee*

Line no - 747-752

The ***_setFeePercentage*** function in the protocol includes a validation that ensures that the *_newFee* argument passed must not be equal to the already existing ***feePercentage***.
However, during the manual review, it was found that the function doesn't include

any validation to check the maximum or minimum threshold of the ***_newFee*** argument being passed. For instance, the ***_newFee*** can never be as low as zero.

```
747     function _setFeePercentage(
748         uint _newFee
749         ) private {
750         require(_newFee != feePercentage, "Error: already set");
751         feePercentage = _newFee;
752     }
```

Keeping in mind the fact that the ***_feePercentage*** state variable plays a significant role during fee calculation while closing a bid, it's imperative to validate the arguments passed adequately.

Recommendation:
Adequate input validations must be included before modifying imperative state variables.

A.1.6 Reuse function result instead of evaluating again

Line no - 283, 375

Explanation:

In both ***bidWithEther***, and ***bidWithToken*** methods, while increasing the countdown to set the auction end period, *timeLeft* is used to store the return value of ***_bidTimeRemaining***(uint) and is not modified.
However, the ***_bidTimeRemaining*** function is being redundantly called again while emitting the ***AuctionUpdated*** event at the lines mentioned above.

```
274         //increase countdown clock
275         uint timeLeft = _bidTimeRemaining(_tokenId);
276         if(timeLeft < 1 hours) {
277             timeLeft + 10 minutes <= 1 hours
278             ? auction.endPeriod += 10 minutes
279             : auction.endPeriod += 1 hours - timeLeft;
280
281             emit AuctionUpdated(
282                 _tokenId,
283                 block.timestamp + _bidTimeRemaining(_tokenId)
284             );
285         }
286
```

Recommendation:
It is common practice to use local variables inside functions as they don't require any storage slots, and since the `***timeLeft***` variable is unchanged, it can be used at the above-mentioned lines to save function execution gas cost.

A.1.7 Prefer using constant on variables which aren't being reassigned

Line no - 32, 34

Explanation:
In DeSpaceAuction, the following variables: initialBidAmount, and DIVISOR are being set once in the constructor and never being reassigned.

```
32      uint public initialBidAmount;
33      uint public feePercentage; // 1% = 1000
34      uint private DIVISOR;
```

Recommendation:
They can be assigned their values using the `constant` keyword to save storage slots for this specific contract unless intended by design.

```
uint public constant initialBidAmount = 1 ether;
uint private constant DIVISOR = 100 * 1000;
```

Informational

A.1.8 Order of layout

Explanation:
As per the Solidity Style Guide, the order of elements and statements should be according to the following layout:

a. Pragma statements
b. Import statements
c. Interfaces
d. Libraries
e. Contracts

Inside each contract, library or interface, use the following order:

a. Type declarations
b. State variables
c. Events

d. Functions

The following documentation links can be used as a reference to understand the correct order: -
https://solidity.readthedocs.io/en/v0.8.7/style-guide.html#order-of-layout
https://solidity.readthedocs.io/en/v0.8.7/style-guide.html#order-of-functions

A.2 Contract Name: DesLinkRegistry.sol
Contract Folder: NFT_Auction_Contracts

___No Issues Found

Automated Test Results

1. DeSpaceAuction.sol

```
Compiled with solc
Number of lines: 1201 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 10 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 1
Number of informational issues: 25
Number of low issues: 11
Number of medium issues: 11
Number of high issues: 6
ERCs: ERC20, ERC165, ERC721

+-------------------------+------------+--------------+--------------------+--------------+--------------------+
|          Name           | # functions |      ERCS    |     ERC20 info     | Complex code |      Features      |
+-------------------------+------------+--------------+--------------------+--------------+--------------------+
|         IERC20          |      6     |     ERC20    |     No Minting     |      No      |                    |
|                         |            |              | Approve Race Cond. |              |                    |
|                         |            |              |                    |              |                    |
|  AggregatorV3Interface  |      5     |              |                    |      No      |                    |
| DesLinkRegistryInterface |      3     |              |                    |      No      |                    |
|         IDESNFT         |      12    | ERC165,ERC721 |                    |      No      |                    |
|      DeSpaceAuction     |      23    |              |                    |      No      |    Receive ETH     |
|                         |            |              |                    |              |      Send ETH      |
|                         |            |              |                    |              | Tokens interaction |
|                         |            |              |                    |              |    Upgradeable     |
+-------------------------+------------+--------------+--------------------+--------------+--------------------+
INFO:Slither:myFlats/AuctionFlat.sol analyzed (10 contracts)
```

2. DesLinkRegistry.sol

```
Compiled with solc
Number of lines: 202 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 3 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 2
Number of informational issues: 9
Number of low issues: 0
Number of medium issues: 0
Number of high issues: 0


+-----------------+-------------+------+------------+--------------+----------+
|       Name      | # functions | ERCS | ERC20 info | Complex code | Features |
+-----------------+-------------+------+------------+--------------+----------+
| DesLinkRegistry |      10     |      |            |      No      |          |
+-----------------+-------------+------+------------+--------------+----------+
INFO:Slither:myFlats/RegistryFlat.sol analyzed (3 contracts)
```

B.1 Contract Name: DeSpaceNFT.sol

Contract Folder: Des-NFT

Low Severity Issues

B.1.1 Coding Style Issues in the Contract

Explanation:

Code readability of a Smart Contract is largely influenced by the Coding Style issues and in some specific scenarios may lead to bugs in the future.

```
Variable ERC721BurnableUpgradeable.__gap (FlatNFT.sol#2109) is not in mixedCase
Parameter DeSpaceNFT.initialize(address)._defaultAdmin (FlatNFT.sol#2148) is not in mixedCase
Parameter DeSpaceNFT.createNFT(string,address)._tokenURI (FlatNFT.sol#2158) is not in mixedCase
Parameter DeSpaceNFT.addMarketplace(address)._marketplace (FlatNFT.sol#2193) is not in mixedCase
Parameter DeSpaceNFT.removeMarketplace(address)._marketplace (FlatNFT.sol#2220) is not in mixedCase
Parameter DeSpaceNFT.isMarketplace(address)._marketplace (FlatNFT.sol#2247) is not in mixedCase
```

During the automated testing, it was found that the **DeSpaceNFT** contract had quite a few code style issues.

Recommendation:

Therefore, it is recommended to fix the issues like naming convention, indentation, and code layout issues in a smart contract.

B.1.2 NatSpec Annotations must be included

Description:

The smart contracts do not include the NatSpec annotations adequately.

Recommendation:

Cover by NatSpec all Contract methods.

Automated Test Results

```
Compiled with solc
Number of lines: 2377 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 21 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 11
Number of informational issues: 93
Number of low issues: 3
Number of medium issues: 5
Number of high issues: 6
ERCs: ERC721, ERC165
```

Name	# functions	ERCS	ERC20 info	Complex code	Features
StringsUpgradeable	4			Yes	
EnumerableSetUpgradeable	24			No	Assembly
CountersUpgradeable	4			No	
IERC721ReceiverUpgradeable	1			No	
AddressUpgradeable	9			No	Send ETH Assembly
DeSpaceNFT	114	ERC165,ERC721		No	Assembly Upgradeable