CERTIK

Security Assessment

Space Nation: Space NFT Registration

CertiK Assessed on Jun 25th, 2024

CertiK Assessed on Jun 25th, 2024

Space Nation: Space NFT Registration

The security assessment was prepared by CertiK, the leader in Web3.0 security.

Executive Summary

TYPES
DeFi

ECOSYSTEM
Binance Smart Chain (BSC) | Ethereum (ETH)

METHODS
Formal Verification, Manual Review, Static Analysis

LANGUAGE
Solidity

TIMELINE
Delivered on 06/25/2024

KEY COMPONENTS
N/A

CODEBASE
Preliminary: Privately Shared
Fix:
https://github.com/SpaceNationOL/contracts/tree/main/contracts/stake
View All in Codebase Page

COMMITS
Final: ab97aa43bf58ff8dce25cb2c04f13398af053b9c
View All in Codebase Page

Vulnerability Summary

10	6	0	0	4	0
Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined

0	Critical		Critical risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.
1	Major	1 Acknowledged	Major risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.
2	Medium	2 Resolved	Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
3	Minor	2 Resolved, 1 Acknowledged	Minor risks can be any of the above, but on a smaller scale. They generally do not compromise the overall integrity of the project, but they may be less efficient than other solutions.
4	Informational	2 Resolved, 2 Acknowledged	Informational errors are often recommendations to improve the style of the code or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

TABLE OF CONTENTS

SPACE NATION: SPACE NFT REGISTRATION

CODEBASE | SPACE NATION: SPACE NFT REGISTRATION

Repository

Preliminary: Privately Shared

Fix: https://github.com/SpaceNationOL/contracts/tree/main/contracts/stake

Commit

Final: ab97aa43bf58ff8dce25cb2c04f13398af053b9c

AUDIT SCOPE | SPACE NATION: SPACE NFT REGISTRATION

4 files audited ● 1 file with Acknowledged findings ● 2 files with Resolved findings ● 1 file without findings

ID	Repo	File	SHA256 Checksum
● SSN	CertiKProject/certik-audit-projects	projects/SpaceNation/contracts/stakeShip.sol	5d84871bcaff28a191fc0ab68920febf68089a68da72eac6dc70e60813437f70
● SSC	CertiKProject/certik-audit-projects	projects/SpaceNation/contracts/stakeShip.sol	8e2dd4f42f9643541d33b86b904933bbbe100a1fbc1a313bbb9e3506d7eb1930
● SSO	SpaceNationOL/contracts	contracts/stake/stakeShip.sol	4dd0e5129fbffac6d0bc2cd2bd99d276e6f2581075394d659357ba7a01ee4032
● SNF	SpaceNationOL/contracts	contracts/stake/SpaceNFTRegistry.sol	91628af57dfb16e9c1cb0b6e090571ee9425e9e03ce385ffad056dcc4ed0add4

APPROACH & METHODS

SPACE NATION: SPACE NFT REGISTRATION

This report has been prepared for Space Nation to discover issues and vulnerabilities in the source code of the Space Nation: Space NFT Registration project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Testing the smart contracts against both common and uncommon attack vectors;
- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

REVIEW NOTES | SPACE NATION: SPACE NFT REGISTRATION

In commit 1f7f42f16b9ff6039075e62347a9409de280748e, The client decided to change most of the naming for functions and variables. Notably, The contract is renamed from stakeShip.sol to SpaceNFTRegistry.sol. Function `stake` and `unstake` is renamed to `register` and `unregister` respectively. Function `disassemble` is renamed to `burn` .

FINDINGS | SPACE NATION: SPACE NFT REGISTRATION

10	0	1	2	3	4
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for Space Nation: Space NFT Registration. Through this audit, we have uncovered 10 issues ranging from different severity levels. Utilizing the techniques of Static Analysis & Manual Review to complement rigorous manual code reviews, we discovered the following findings:

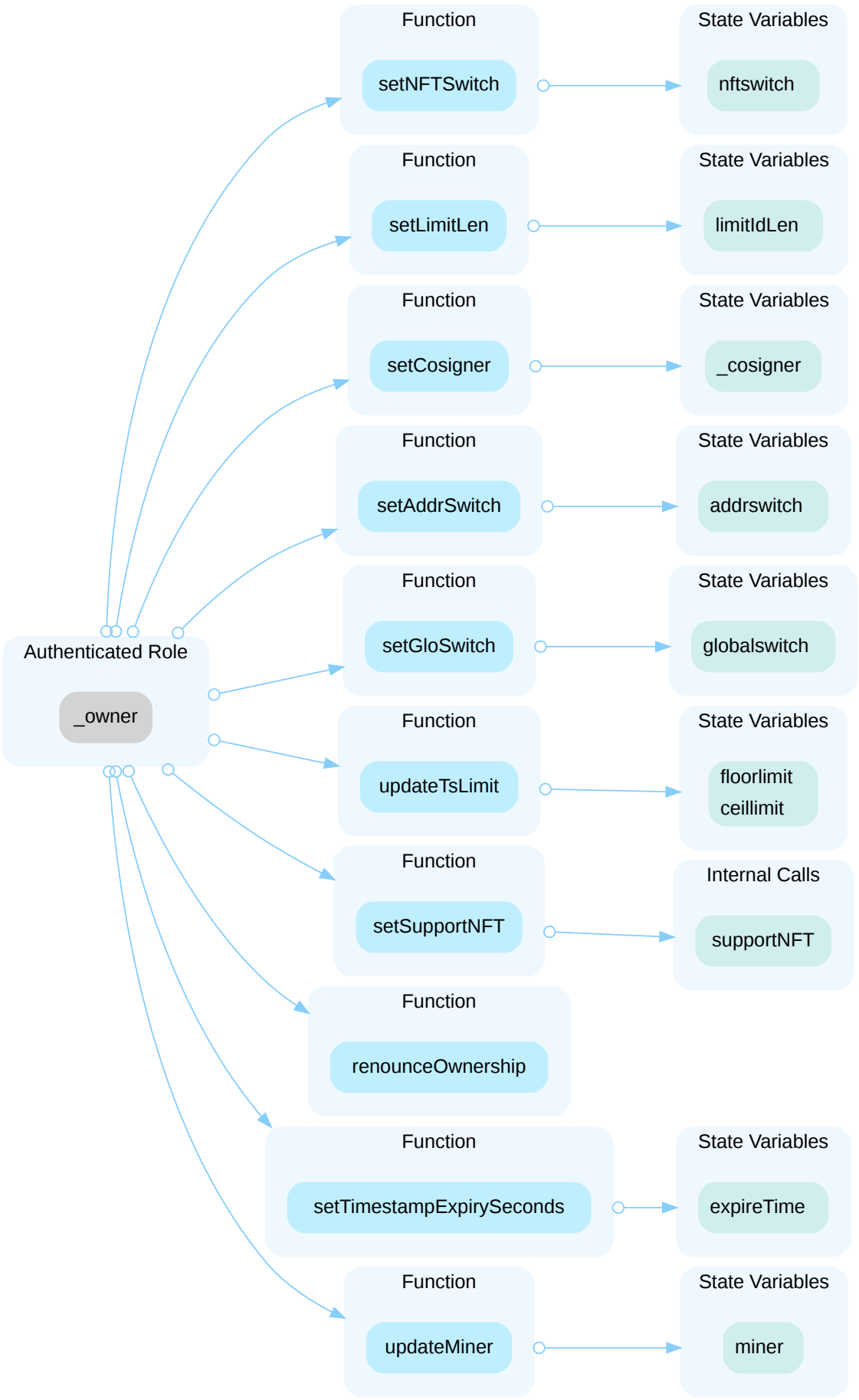
ID	Title	Category	Severity	Status
SSN-04	**Centralization Risks In SpaceNFTRegistry.Sol**	**Centralization**	**Major**	**Acknowledged**
GIT-01	Lack Of Update To `stakeInfo[nft][nftId]` In Functions `unstake` And `disassemble`	Logical Issue	Medium	Resolved
SSC-01	Potential Reentrancy Attack	Logical Issue	Medium	Resolved
GLOBAL-02	Out Of Scope Dependency Usage	Design Issue	Minor	Acknowledged
SSN-06	ERC721 Token May Be Transferred To A Contract That Can Not Handle Them	Logical Issue	Minor	Resolved
SSN-07	Missing Zero Address Validation	Volatile Code	Minor	Resolved
GLOBAL-01	Lack Of Reward For User	Design Issue	Informational	Acknowledged
SSN-02	Inconsistency Between Comments And Implementation	Inconsistency	Informational	Resolved
SSN-03	Purpose Of Function `disassemble`	Design Issue	Informational	Acknowledged
SSO-02	Unused Return Parameter	Inconsistency	Informational	Resolved

SSN-04 | CENTRALIZATION RISKS IN SPACENFTREGISTRY.SOL

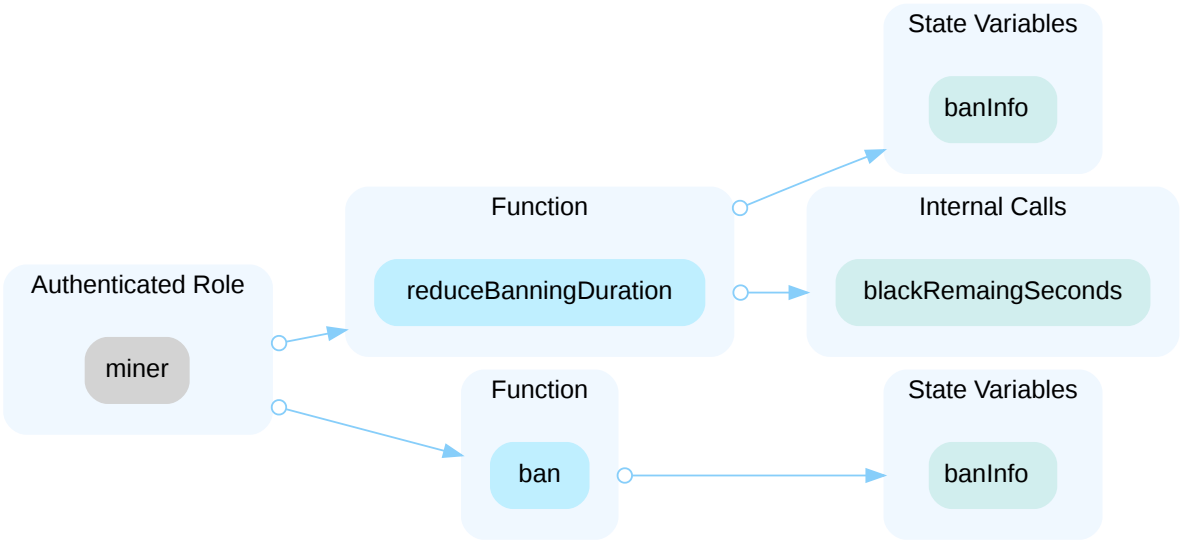
Category	Severity	Location	Status
Centralization	Major	projects/SpaceNation/contracts/stakeShip.sol (pre): 85, 94, 103, 124, 133, 143, 154, 237, 252, 334, 422	Acknowledged

Description

In the contract `SpaceNFTRegistry` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and change switches, set expire time, update miner address, set signer, set NFT address and change time and NFT length limit.


Function
setNFTSwitch
State Variables
nftswitch
Function
setLimitLen
State Variables
limitIdLen
Function
setCosigner
State Variables
_cosigner
Function
setAddrSwitch
State Variables
addrswitch
Function
setGloSwitch
State Variables
globalswitch
Authenticated Role
_owner
Function
updateTsLimit
State Variables
floorlimit
ceillimit
Function
setSupportNFT
Internal Calls
supportNFT
Function
renounceOwnership
Function
setTimestampExpirySeconds
State Variables
expireTime
Function
updateMiner
State Variables
miner

In the contract `SpaceNFTRegistry` the role `miner` has authority over the functions shown in the diagram below. Any compromise to the `miner` account may allow the hacker to take advantage of this authority and change blacklist.



Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multisignature wallets. Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND

- Introduction of a DAO/governance/voting module to increase transparency and user involvement.
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles.
  OR
- Remove the risky functionality.

Alleviation

[Space Nation Team, 06/20/2024]: The team acknowledged this issue. The private key of the privileged address is stored in a computer under a secure environment.

[CertiK, 06/20/2024]: It is suggested to implement the aforementioned methods to avoid centralized failure. Also, CertiK strongly encourages the project team to periodically revisit the private key security management of all addresses related to centralized roles.

GIT-01 LACK OF UPDATE TO `stakeInfo[nft][nftId]` IN FUNCTIONS `unstake` AND `disassemble`

Category	Severity	Location	Status
Logical Issue	Medium	projects/SpaceNation/contracts/stakeShip.sol (pre): 188; contracts/stake/stakeShip.sol (fix2): 314~321	Resolved

Description

The `stakeInfo[nft][nftId]` mapping is used to store user staking information, including the user's address and the end staking time. However, this mapping is not cleared when a user calls the `unstake` function. As a result, the user can still call the `extend` function even after unstaking their NFT. This oversight can lead to inconsistencies in the staking process and potential exploitation by allowing users to manipulate the staking period for NFTs that are no longer staked.

Users can extend the staking period of an NFT that has already been unstaked, leading to potential discrepancies in staking rewards and timelines. This could be exploited to gain unintended benefits from the staking system.

This issue also exists in function `disassemble`, which is used for burning NFTs from user. When a NFT is burned, its corresponding staking record should also be cleared.

Recommendation

Ensure that the `stakeInfo[nft][nftId]` mapping is properly cleared or updated when a user calls the `unstake` and `disassemble` function.

Alleviation

[Space Nation Team, 06/25/2024]: The team heeded the advice and resolved the issue in commit: 1f7f42f16b9ff6039075e62347a9409de280748e.

SSC-01 | POTENTIAL REENTRANCY ATTACK

Category	Severity	Location	Status
Logical Issue	Medium	projects/SpaceNation/contracts/stakeShip.sol (fix): 176, 204	Resolved

Description

In the updated version of the contract, `safeTransferFrom` replaced `transferFrom` when handling NFT transfers. However, several related functions in the contracts do not follow the Checks-Effects-Interactions pattern or lack the `nonReentrant` modifier, potentially exposing them to reentrancy attacks. This risk arises because the ERC721's `safeTransferFrom()` function can call the receiver's `onERC721Received()` function if the receiver is a contract.

For example, the `stake` function is missing the `nonReentrant` modifier, and it performs ERC721 token transfers before updating state variables.

Recommendation

It is recommended to implement both modifications:

1. **Implement `nonReentrant` Modifier:** Apply the `nonReentrant` modifier from OpenZeppelin's `ReentrancyGuard` to all functions that perform external calls, including token transfers, to ensure that no nested (reentrant) calls can occur.

2. **Adhere to Checks-Effects-Interactions Pattern:** Make sure to complete all state changes before executing external calls. Rearrange the code to update state variables prior to making any external function calls. For example, store token IDs in memory variables and position the `ERC721.safeTransferFrom()` function at the conclusion of the contract functions.

Alleviation

[Space Nation Team, 06/25/2024]: The team heeded the advice and resolved the issue in commit: 1f7f42f16b9ff6039075e62347a9409de280748e.

GLOBAL-02 | OUT OF SCOPE DEPENDENCY USAGE

Category	Severity	Location	Status
Design Issue	Minor		Acknowledged

Description

The contract is serving as the underlying entity to interact with one or more out of scope contracts. The scope of the audit treats these contracts as black boxes and assumes their functional correctness. However, in the real world, those contracts can be compromised and this may lead to lost or stolen assets.

```
        address nft,
```

- The function `stakeShip.stake` interacts with third party contract with `IERC721` interface via `nft`.

```
    function unstake(address nft, uint64 nftId) external {
```

- The function `stakeShip.unstake` interacts with third party contract with `IERC721` interface via `nft`.

```
        address nft,
```

- The function `stakeShip.disassemble` interacts with third party contract with `NFTBurn` interface via `nft`.

Recommendation

The auditors understood that the business logic requires interaction with other contracts. It is recommended for the team to constantly monitor the statuses of out of scope dependencies to mitigate the side effects when unexpected activities are observed.

Alleviation

[Space Nation Team, 06/20/2024]: The team acknowledged the finding and decided not to change the current codebase.

The NFT contract addresses of external dependencies have all been reviewed and verified (supportnft) to ensure the external functions are callable.

SSN-06 ERC721 TOKEN MAY BE TRANSFERRED TO A CONTRACT THAT CAN NOT HANDLE THEM

Category	Severity	Location	Status
Logical Issue	Minor	projects/SpaceNation/contracts/stakeShip.sol (pre): 204	Resolved

Description

The EIP-721 standard says the following about `transferFrom()`:

```
    /// @notice Transfer ownership of an NFT -- THE CALLER IS RESPONSIBLE
    ///  TO CONFIRM THAT `_to` IS CAPABLE OF RECEIVING NFTS OR ELSE
    ///  THEY MAY BE PERMANENTLY LOST
    /// @dev Throws unless `msg.sender` is the current owner, an authorized
    ///  operator, or the approved address for this NFT. Throws if `_from` is
    ///  not the current owner. Throws if `_to` is the zero address. Throws if
    ///  `_tokenId` is not a valid NFT.
    /// @param _from The current owner of the NFT
    /// @param _to The new owner
    /// @param _tokenId The NFT to transfer
    function transferFrom(address _from, address _to, uint256 _tokenId) external
payable;
```

The function `unstake` transfers an ERC721 token to the input address via `transferFrom()`. This method does not check that the `unstaker` is a contract, it is designed to support ERC721 tokens. If the `unstaker` is a smart contract not designed to handle ERC721 tokens, they can become locked in the contract forever.

Recommendation

We recommend using `safeTransferFrom()` instead of `transferFrom()`. If the `unstaker` refers to a smart contract, it must implement {IERC721Receiver-onERC721Received}, which is called upon a safe transfer.

Alleviation

[Space Nation Team, 06/20/2024]: The team heeded the advice and resolved the issue.

SSN-07 | MISSING ZERO ADDRESS VALIDATION

Category	Severity	Location	Status
Volatile Code	Minor	projects/SpaceNation/contracts/stakeShip.sol (pre): 334	Resolved

Description

Addresses are not validated before assignment or external calls, potentially allowing the use of zero addresses and leading to unexpected behavior or vulnerabilities. For example, transferring tokens to a zero address can result in a permanent loss of those tokens.

```
334         _cosigner = cosigner;
```

- `cosigner` is not zero-checked before being used.

Recommendation

It is recommended to add a zero-check for the passed-in address value to prevent unexpected errors.

Alleviation

[Space Nation Team, 06/20/2024]: The team heeded the advice and resolved the issue.

GLOBAL-01 | LACK OF REWARD FOR USER

Category	Severity	Location	Status
Design Issue	Informational		Acknowledged

Description

In the current implementation, users do not receive any rewards when staking their NFTs into the contract. Without rewards, users have no financial or intrinsic motivation to stake their NFTs, which undermines the purpose of the staking mechanism. This could result in lower user engagement and participation in the staking program, reducing the overall effectiveness and attractiveness of the platform.

Recommendation

Recommend elaborate the reward mechanism behind the current implementation.

Alleviation

[Space Nation Team, 06/20/2024]: The team acknowledged the finding and decided not to change the current codebase.

This is a staking contract related to a game, where the rewards are distributed within the off-chain game. Players must stake their NFTs into this contract in order for the game to allow them to use the virtual items corresponding to the NFTs within the game.

[CertiK Team, 06/20/2024]: The off-chain component is out of the scope of this audit. CertiK recommend the team to constantly monitor the statuses of those components to mitigate the side effects when unexpected activities are observed.

SSN-02 INCONSISTENCY BETWEEN COMMENTS AND IMPLEMENTATION

Category	Severity	Location	Status
Inconsistency	Informational	projects/SpaceNation/contracts/stakeShip.sol (pre): 209	Resolved

Description

The comment for function `extend` states "Extending will delete the blacklist state". However, in the current implementation, a user will not be able to extend if he is still being blacklisted. If the blacklist did not expire, the transaction will revert which contradicts with the comments.

Recommendation

Recommend the team resolve the inconsistency.

Alleviation

[Space Nation Team, 06/20/2024]: The team heeded the advice and resolved the issue. The comment is modified to align with the implementation.

SSN-03 | PURPOSE OF FUNCTION `disassemble`

Category	Severity	Location	Status
Design Issue	Informational	projects/SpaceNation/contracts/stakeShip.sol (pre): 265~272	Acknowledged

Description

According to the comments, The purpose of function `disassemble` is "Stakers can burn their NFTs based on a valid signed request." We would like to know why a staker would want their NFT being burned, as there is no apparent benefit doing so.

Recommendation

Recommend the team elaborate the design.

Alleviation

[Space Nation Team, 06/20/2024]: The team acknowledged the finding and decided not to change the current codebase.

This is a contract related to a game, where the in-game items (NFTs) may have upgrade or disassembly operations in specific game scenarios.

This will be an operation initiated by the player themselves and signed by the official game wallet.

SSO-02 | UNUSED RETURN PARAMETER

Category	Severity	Location	Status
Inconsistency	Informational	contracts/stake/stakeShip.sol (fix2): 482	Resolved

Description

In function `stakeDataCheck`, `endts` has not been assigned to any value. In addition, `endts` is never returned, `stakets` is returned instead.

Recommendation

Recommend revise the code.

Alleviation

[Space Nation Team, 06/25/2024]: The team heeded the advice and resolved the issue in commit: 1f7f42f16b9ff6039075e62347a9409de280748e.

OPTIMIZATIONS | SPACE NATION: SPACE NFT REGISTRATION

ID	Title	Category	Severity	Status
SSN-01	Inefficient Memory Parameter	Inconsistency	Optimization	Resolved
SSO-01	Event Emission Can Be Optimized	Code Optimization	Optimization	Resolved

SSN-01 | INEFFICIENT MEMORY PARAMETER

Category	Severity	Location	Status
Inconsistency	Optimization	projects/SpaceNation/contracts/stakeShip.sol (pre): 110	Resolved

Description

One or more parameters with `memory` data location are never modified in their functions and those functions are never called internally within the contract. Thus, their data location can be changed to `calldata` to avoid the gas consumption copying from calldata to memory.

```
110     function setSupportNFT(address[] memory nfts, bool status)
```

`setSupportNFT` has memory location parameters: `nfts`.

Recommendation

We recommend changing the parameter's data location to `calldata` to save gas.

- For Solidity versions prior to 0.6.9, since public functions are not allowed to have calldata parameters, the function visibility also needs to be changed to `external`.
- For Solidity versions prior to 0.5.0, since parameter data location is implicit, changing the function visibility to `external` will change the parameter's data location to calldata as well.

Alleviation

[Space Nation Team, 06/20/2024]: The team heeded the advice and resolved the issue.

SSO-01 | EVENT EMISSION CAN BE OPTIMIZED

Category	Severity	Location	Status
Code Optimization	Optimization	contracts/stake/stakeShip.sol (fix2): 255~257, 305~307	Resolved

Description

The following event emission in function `extend` can be optimized:

```
255             uint256[] memory tem = new uint256[](1);
256             tem[0] = nftId;
257             emit UserOp(nft, staker, tem, endts, 1);
```

The event emission can be streamlined by eliminating the need to create a new array and directly using the nftId. The optimized code is as follows:

```
1           emit UserOp(nftId, staker, tem, endts, 1);
```

Similar issue also exists in `redeem`.

Recommendation

Update the `extend` and `redeem` function to use the optimized event emission code to enhance performance and reduce gas costs.

Alleviation

[Space Nation Team, 06/25/2024]: The team heeded the advice and resolved the issue in commit: 1f7f42f16b9ff6039075e62347a9409de280748e.

FORMAL VERIFICATION

SPACE NATION: SPACE NFT REGISTRATION

Formal guarantees about the behavior of smart contracts can be obtained by reasoning about properties relating to the entire contract (e.g. contract invariants) or to specific functions of the contract. Once such properties are proven to be valid, they guarantee that the contract behaves as specified by the property. As part of this audit, we applied formal verification to prove that important functions in the smart contracts adhere to their expected behaviors.

Considered Functions And Scope

In the following, we provide a description of the properties that have been used in this audit. They are grouped according to the type of contract they apply to.

Verification of Standard Ownable Properties

We verified *partial* properties of the public interfaces of those token contracts that implement the Ownable interface. This involves:

- function `owner` that returns the current owner,
- functions `renounceOwnership` that removes ownership,
- function `transferOwnership` that transfers the ownership to a new owner.

The properties that were considered within the scope of this audit are as follows:

Property Name	Title
ownable-renounce-ownership-is-permanent	Once Renounced, Ownership Cannot be Regained
ownable-transferownership-correct	Ownership is Transferred
ownable-owner-succeed-normal	`owner` Always Succeeds
ownable-renounceownership-correct	Ownership is Removed

Verification Results

In the remainder of this section, we list all contracts where formal verification of at least one property was not successful. There are several reasons why this could happen:

- False: The property is violated by the project.
- Inconclusive: The proof engine cannot prove or disprove the property due to timeouts or exceptions.
- Inapplicable: The property does not apply to the project.

Detailed Results For Contract SpaceNFTRegistry (contracts/stake/SpaceNFTRegistry.sol) In

Commit ab97aa43bf58ff8dce25cb2c04f13398af053b9c

Verification of Standard Ownable Properties

Detailed Results for Function `renounceOwnership`

Property Name	Final Result	Remarks
ownable-renounce-ownership-is-permanent	Inconclusive	
ownable-renounceownership-correct	True	

Detailed Results for Function `transferOwnership`

Property Name	Final Result	Remarks
ownable-transferownership-correct	True	

Detailed Results for Function `owner`

Property Name	Final Result	Remarks
ownable-owner-succeed-normal	True	

Detailed Results For Contract stakeShip (contracts/stake/stakeShip.sol) In Commit 05633207fc6f1e0b748d0c9d62a57546cc2212ae

Verification of Standard Ownable Properties

Detailed Results for Function `owner`

Property Name	Final Result	Remarks
ownable-owner-succeed-normal	True	

Detailed Results for Function `transferOwnership`

Property Name	Final Result	Remarks
ownable-transferownership-correct	True	

Detailed Results for Function `renounceOwnership`

Property Name	Final Result	Remarks
ownable-renounceownership-correct	True	
ownable-renounce-ownership-is-permanent	Inconclusive	

Detailed Results For Contract stakeShip (projects/SpaceNation/contracts/stakeShip.sol) In SHA256 Checksum 26aa5401cc76b78ab6596f0bd4f55dc1d919c4c9

Verification of Standard Ownable Properties

Detailed Results for Function `transferOwnership`

Property Name	Final Result	Remarks
ownable-transferownership-correct	True	

Detailed Results for Function `renounceOwnership`

Property Name	Final Result	Remarks
ownable-renounceownership-correct	True	
ownable-renounce-ownership-is-permanent	Inconclusive	

Detailed Results for Function `owner`

Property Name	Final Result	Remarks
ownable-owner-succeed-normal	True	

Detailed Results For Contract stakeShip (projects/SpaceNation/contracts/stakeShip.sol) In SHA256 Checksum fbc749bd6e610eb2ca0263fbd8567fe86be6f3ec

Verification of Standard Ownable Properties

Detailed Results for Function `owner`

Property Name	Final Result	Remarks
ownable-owner-succeed-normal	True	

Detailed Results for Function `transferOwnership`

Property Name	Final Result	Remarks
ownable-transferownership-correct	True	

Detailed Results for Function `renounceOwnership`

Property Name	Final Result	Remarks
ownable-renounceownership-correct	True	
ownable-renounce-ownership-is-permanent	Inconclusive	

APPENDIX | SPACE NATION: SPACE NFT REGISTRATION

Finding Categories

Categories	Description
Inconsistency	Inconsistency findings refer to different parts of code that are not consistent or code that does not behave according to its specification.
Volatile Code	Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases and may result in vulnerabilities.
Logical Issue	Logical Issue findings indicate general implementation issues related to the program logic.
Centralization	Centralization findings detail the design choices of designating privileged roles or other centralized controls over the code.
Design Issue	Design Issue findings indicate general issues at the design level beyond program logic that are not covered by other finding categories.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

Details on Formal Verification

Some Solidity smart contracts from this project have been formally verified. Each such contract was compiled into a mathematical model that reflects all its possible behaviors with respect to the property. The model takes into account the semantics of the Solidity instructions found in the contract. All verification results that we report are based on that model.

The following assumptions and simplifications apply to our model:

- Certain low-level calls and inline assembly are not supported and may lead to a contract not being formally verified.
- We model the semantics of the Solidity source code and not the semantics of the EVM bytecode in a compiled contract.

Formalism for property specifications

All properties are expressed in a behavioral interface specification language that CertiK has developed for Solidity, which allows us to specify the behavior of each function in terms of the contract state and its parameters and return values, as well as contract properties that are maintained by every observable state transition. Observable state transitions occur when the

contract's external interface is invoked and the invocation does not revert, and when the contract's Ether balance is changed by the EVM due to another contract's "self-destruct" invocation. The specification language has the usual Boolean connectives, as well as the operator `\old` (used to denote the state of a variable before a state transition), and several types of specification clause:

Apart from the Boolean connectives and the modal operators "always" (written `[]`) and "eventually" (written `<>`), we use the following predicates to reason about the validity of atomic propositions. They are evaluated on the contract's state whenever a discrete time step occurs:

- `requires [cond]` - the condition `cond`, which refers to a function's parameters, return values, and contract state variables, must hold when a function is invoked in order for it to exhibit a specified behavior.
- `ensures [cond]` - the condition `cond`, which refers to a function's parameters, return values, and both `\old` and current contract state variables, is guaranteed to hold when a function returns if the corresponding requires condition held when it was invoked.
- `invariant [cond]` - the condition `cond`, which refers only to contract state variables, is guaranteed to hold at every observable contract state.
- `constraint [cond]` - the condition `cond`, which refers to both `\old` and current contract state variables, is guaranteed to hold at every observable contract state except for the initial state after construction (because there is no previous state); constraints are used to restrict how contract state can change over time.

Description of the Analyzed Ownable Properties

Properties related to function `renounceOwnership`

ownable-renounce-ownership-is-permanent

The contract must prohibit regaining of ownership once it has been renounced.

Specification:

```
constraint \old(owner()) == address(0) ==> owner() == address(0);
```

ownable-renounceownership-correct

Invocations of `renounceOwnership()` must set ownership to address(0).

Specification:

```
ensures this.owner() == address(0);
```

Properties related to function `transferOwnership`

ownable-transferownership-correct

Invocations of `transferOwnership(newOwner)` must transfer the ownership to the `newOwner`.

Specification:

```
ensures this.owner() == newOwner;
```

Properties related to function `owner`

ownable-owner-succeed-normal

Function `owner` must always succeed if it does not run out of gas.

Specification:

```
reverts_only_when false;
```

DISCLAIMER | CERTIK

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK's prior written consent in each instance.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR

UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

CertiK | Securing the Web3 World

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.