HALBORN

Seascape - Ninja Spin

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **September 2nd, 2022 - September 9th, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	09/02/2022	Luis Arroyo
0.2	Draft Review	09/12/2022	Kubilay Onur Gungor
0.3	Draft Review	09/12/2022	Gabi Urrutia
1.0	Remediation Plan	09/13/2022	Luis Arroyo
1.1	Remediation Plan Review	09/13/2022	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
Kubilay Onur Gungor	Halborn	Kubilay.Gungor@halborn.com
Luis Arroyo	Halborn	Luis.Arroyo@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Seascape engaged Halborn to conduct a security audit on their Ninja Spin smart contracts beginning on September 2nd, 2022 and ending on September 9th, 2022. The security assessment was scoped to some smart contracts provided in the GitHub repository blocklords/bigbang-smartcontracts.

1.2 AUDIT SUMMARY

The team at Halborn was provided a week for the engagement and assigned one 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 were addressed and acknowledged 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 code and can quickly identify items that do not follow the security best practices. The following phases and associated tools were used during the audit:

- Research into architecture and purpose
- Smart contract manual code review and walkthrough
- Graphing out functionality and contract logic/connectivity/functions (solgraph)
- Manual assessment of use and safety for the critical Solidity variables and functions in scope to identify any arithmetic related vulnerability classes
- Manual testing by custom scripts
- Scanning of solidity files for vulnerabilities, security hotspots or bugs. (MythX)
- Static Analysis of security for scoped contract, and imported functions. (Slither)
- Testnet deployment (Brownie, Remix IDE)

RISK METHODOLOGY:

Vulnerabilities or issues observed by Halborn are ranked based on the risk assessment methodology by measuring the **LIKELIHOOD** of a security incident and the **IMPACT** should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. The quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that were used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

RISK SCALE - LIKELIHOOD

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

RISK SCALE - IMPACT

5 - May cause devastating and unrecoverable impact or loss.
4 - May cause a significant level of impact or loss.

3 - May cause a partial impact or loss to many.
2 - May cause temporary impact or loss.
1 - May cause minimal or un-noticeable impact.

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

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

1.4 SCOPE

IN-SCOPE:
The security assessment was scoped to the following smart contracts:

- BigBangNFT.sol
- BigBangFactory.sol
- BigBangGame.sol

Located on the Commit ID: 4e6b040ec981df4f8ce9993bea75563ca5b9fa07

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	1	1	2	3

LIKELIHOOD

IMPACT

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

EXECUTIVE OVERVIEW

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL01 - CONTRACT ADMIN CAN REVOKE AND RENOUNCE HIMSELF	High	SOLVED - 09/13/2022
HAL02 - CONTRACT DOES NOT ALLOW MINTING NFT WITH THE ID 0	Medium	SOLVED - 09/13/2022
HAL03 - NATIVE TOKEN FUNCTIONALITY COULD NOT BE USED	Low	SOLVED - 09/13/2022
HAL04 - MISSING ZERO ADDRESS CHECK	Low	SOLVED - 09/13/2022
HAL05 - UNUSED PARAMETERS	Informational	ACKNOWLEDGED
HAL06 - PRAGMA VERSION	Informational	ACKNOWLEDGED
HAL07 - UNSAFE MATH CALCULATIONS	Informational	ACKNOWLEDGED

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 that deploys the contract. In the BigBangNFTFactory.sol smart contract, Only Admin can perform some privileged actions such as setNft(), addAdmin(), addGenerator() 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 could revoke his role via renounceAdmin(). If an admin is mistakenly renounced, administrative access will result in the contract having no admin, eliminating the ability to call privileged functions. In such a case, contracts would have to be redeployed.

Code Location:

Listing 1: BigBangNFTFactory.sol (Line 55)

```
53    function renounceAdmin() public virtual
54    {
55        renounceRole(DEFAULT_ADMIN_ROLE, msg.sender);
56    }
```

Proof of Concept:

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

Risk Level:

Likelihood - 3
Impact - 5

Recommendation:

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

Remediation Plan:

SOLVED: Now the owner cannot renounce his role. This issue was fixed in commit ID 7d4d7a60ef10d499be2c672a6b623211ef288246

3.2 (HAL-02) CONTRACT DOES NOT ALLOW MINTING NFT WITH THE ID 0 - MEDIUM

Description:

The BigBangNFT.sol contract constructor contains an increment of the counter variable, making the IDs begin on 1.

Listing 2: ScapeStore.sol (Line 26)

```
25      constructor() public ERC721("BigBang NFT", "BB") {
26          nftId.increment();
27      }
```

It is known that some collections start at the NFT ID 0.

Risk Level:

Likelihood - 3
Impact - 3

Recommendation:

It is recommended to remove the increment so users can also mint the NFT with ID 0.

Remediation Plan:

SOLVED: The Ninja Scape Game can now start minting NFTs at ID 0. This issue was fixed in commit ID 63f46b35467cd3865416b1dae2aa99ea4363cf63

3.3 (HAL-03) NATIVE TOKEN FUNCTIONALITY COULD NOT BE USED - LOW

Description:

In the _safeTransfer() function from BigBangGame.sol smart contract, two different types of transfers could be performed. if _token address sent as parameter is different from 0, the function will perform a token.transfer(). If _token is address(0), then the contract will send the native coin to the sender.

It is not possible to add address(0) to the list of allowed tokens, as the addresses are validating this parameter to be different from 0. This makes it impossible to operate with native tokens.

Code Location:

Listing 3: BigBangGame.sol (Line 38)

```
37 constructor(address _token, address _nft, address _factory,
↳ address _verifier) public {
38     require(_token != address(0), "BBGame: Token can't be zero
↳ address");
39     require(_nft != address(0), "BBGame: Nft can't be zero address
↳ ");
40     require(_verifier != address(0), "BBGame: Verifier can't be
↳ zero address");
41
42     BBNft     = _nft;
43     BBFactory = _factory;
44     verifier    = _verifier;
45
46     changeAllowed[_token] = true;
47     token[typeId]     = _token;
48   }
```

Listing 4: BigBangGame.sol (Line 212)

```
211   function addToken(address _token) public onlyOwner {
212     require(_token != address(0), "BBGame: Token can't be zero
 ↳ address");
213     require(!changeAllowed[_token], "BBGame: This token is exist")
 ↳ ;
214
215     changeAllowed[_token] = true;
216     token[++typeId] = _token;
217
218     emit AddToken(_token, typeId, block.timestamp);
219   }
```

Listing 5: BigBangGame.sol (Lines 168,178)

```
167   function _safeTransfer(address _token, address _to, uint256
 ↳ _amount) internal {
168     if (_token != address(0)) {
169       IERC20 _rewardToken = IERC20(_token);
170
171       uint256 _balance = _rewardToken.balanceOf(address(this));
172       require(_amount <= _balance, "BBGame: Do not have enough
 ↳ token to reward");
173
174       uint256 _beforBalance = _rewardToken.balanceOf(_to);
175       _rewardToken.transfer(_to, _amount);
176
177       require(_rewardToken.balanceOf(_to) == _beforBalance +
 ↳ _amount, "BBGame: Invalid transfer");
178     } else {
179
180       uint256 _balance = address(this).balance;
181       require(_amount <= _balance, "BBGame: Do not have enough
 ↳ token to reward");
182
183       payable(_to).transfer(_amount);
184     }
185   }
186
187   // Accept native tokens.
188   receive() external payable {
189     //React to receiving ether
190   }
```

Risk Level:

Likelihood - 3
Impact - 2

Recommendation:

It is recommended to implement a function to add address(0) in the array of allowed tokens in order to allow transactions with native token.

Remediation Plan:

SOLVED: The SeaScape Team now allows the BigBangGame.sol contract to add the address(0) to specify that the native token could be used. This issue was fixed in commit ID 6fbb5845900735349c19c5e751991b9854d9a2a5

3.4 (HAL-04) MISSING ZERO ADDRESS CHECK - LOW

Description:

There is no validation of the addresses in the mint() and access control functions. Addresses should be validated and checked that are different from zero when necessary. This issue is present in all the smart contracts, in the constructors and functions that use addresses as parameters. These examples show how the factory could be set up with a wrong address and how mint could burn NFTs if _owner is address(0)

Some code location examples:

Listing 6: BigBangNFTFactory.sol

```
25     constructor(address _nft) public {
26         nft = BigBangNFT(_nft);
27         _setupRole(DEFAULT_ADMIN_ROLE, msg.sender);
28     }
```

Listing 7: BigBangNFTFactory.sol

```
34     function mint(address _owner, uint256 _quality, uint256 _image)
↳  public onlyGenerator returns(uint256) {
35         require (_quality > 0 && _quality < 6, "NFT Factory: invalid
↳  quality");
36         return nft.mint(_owner, _quality, _image);
37     }
```

Risk Level:

Likelihood - 3
Impact - 2

Recommendation:

Validate that necessary address inputs are different from zero.

Remediation Plan:

SOLVED: The SeaScape Team now checks the address inputs to ensure they are non-zero. This issue has been fixed in commit ID [7d4d7a60ef10d499be2c672a6b623211ef288246](https://github.com/Seastarinteractive/mo

3.5 (HAL-05) UNUSED PARAMETERS - INFORMATIONAL

Description:

There are functions whose parameters are never used in some smart contracts. These parameters have no effect on the code.

Code Location:

- operator, from, tokenId, data (BigBangGame.sol#190)
- BBFactory (BigBangGame.sol#231)

Risk Level:

Likelihood - 1
Impact - 2

Recommendation:

All parameters given to a function should affect the state of the code. The operator, from, tokenId, and data from onERC721Received() functions should implement any logic with that parameters, but they are not used for any contract state operations.

Remediation Plan:

ACKNOWLEDGED: The SeaScape team acknowledged this finding.

3.6 (HAL-06) PRAGMA VERSION - INFORMATIONAL

Description:

The BigBangGame smart contracts use the pragma version 0.6.7 which was released on May 4, 2020. latest pragma version 0.8.16 released on August 8, 2022, solves different issues. It is also noticeable that Solidity versions after 0.8.0 also implement default overflow protection on arithmetic operations.

Reference: Solidity Releases

Code Location:

note: All Ninja Spin smart contracts implement same pragma version.

Listing 8: BigBangGame.sol

```
1 pragma solidity 0.6.7;
```

Risk Level:

Likelihood - 2
Impact - 1

Recommendation:

It is recommended to update the pragma version used in the BigBangGame smart contracts to versions above or equal to 0.8.0.

Remediation Plan:

ACKNOWLEDGED: The SeaScape team acknowledged this finding.

3.7 (HAL-07) UNSAFE MATH CALCULATIONS - INFORMATIONAL

Description:

Testing revealed that calculations within the smart contract did not make use of a safe math library. While Solidity pragma > 0.8.0 reverts to overflows by default, the code was making use of an older compiler version; thus it is vulnerable to integer overflows and underflows.

Halborn could not find a path to exploit the integer overflows; however, there might be some extreme test cases where this can be exploited by an attacker.

Code Location:

Listing 9: BigBangGame.sol (Line 148)

```
128   function goldChangeToken(uint256 _gold, uint256 _typeId, uint8
 ↳ _v, bytes32 _r, bytes32 _s) external {
129     require(_gold > 0, "BBGame: The exchange amount must greater
 ↳ than zero");
130     require(checkToken(_typeId), "BBGame: Do not have this token
 ↳ type");
131
132     uint256 chainId;
133     assembly {
134         chainId := chainid()
135     }
136
137     {
138       bytes memory prefix     = "\x19Ethereum Signed Message:\n32"
 ↳ ;
139       bytes32 message         = keccak256(abi.encodePacked(_gold,
 ↳ msg.sender, nonce[msg.sender], address(this), chainId));
140       bytes32 hash            = keccak256(abi.encodePacked(prefix,
 ↳  message));
141       address recover         = ecrecover(hash, _v, _r, _s);
142
143       require(recover == verifier, "BBGame: Verification failed
```

```
 ↳ about goldChangeToken");
144     }
145
146     nonce[msg.sender]++;
147
148     uint256 _tokenAmount = _gold * MULTIPLIER / ratio;
149     _safeTransfer(token[_typeId], msg.sender, _tokenAmount);
150
151     emit GoldChangeToken(msg.sender, _typeId, _gold, _tokenAmount,
 ↳  block.timestamp);
152
153   }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended that either the compiler version is upgraded to pragma >= 0.8.0 or that a library such as OpenZeppelin's SafeMath is used.

Remediation Plan:

ACKNOWLEDGED: The SeaScape team acknowledged this finding.

AUTOMATED TESTING

4.1 STATIC ANALYSIS REPORT

Description:

Halborn used automated testing techniques to enhance the coverage of certain areas of the smart contracts in scope. Among the tools used was Slither, a Solidity static analysis framework. After Halborn verified the smart contracts in the repository and was able to compile them correctly into their ABIs and binary format, Slither was run against the 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:

```
BigBangGame._safeTransfer(address,address,uint256) (contracts/BigBangGame.sol#168-186) sends eth to arbitrary user
        Dangerous calls:
        - address(_to).transfer(_amount) (contracts/BigBangGame.sol#184)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations

BigBangGame.tokenChangeGold(uint256,uint256) (contracts/BigBangGame.sol#115-125) ignores return value by _token.transferFrom(msg.sender,address(this),_amount) (contracts/BigBangGame.sol#122)
BigBangGame._safeTransfer(address,address,uint256) (contracts/BigBangGame.sol#168-186) ignores return value by _rewardToken.transfer(_to,_amount) (contracts/BigBangGame.sol#176)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

BigBangGame._safeTransfer(address,address,uint256) (contracts/BigBangGame.sol#168-186) uses a dangerous strict equality:
        - require(bool,string)(_rewardToken.balanceOf(_to) == _beforBalance + _amount,BBGame: Invalid transfer) (contracts/BigBangGame.sol#178)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in BigBangGame.exportNft(uint256,uint256,uint256,uint8,bytes32,bytes32) (contracts/BigBangGame.sol#80-112):
        External calls:
        - nft.safeTransferFrom(address(this),msg.sender,player[msg.sender][_id]) (contracts/BigBangGame.sol#98)
        State variables written after the call(s):
        - delete player[msg.sender][_id] (contracts/BigBangGame.sol#100)
Reentrancy in BigBangGame.exportNft(uint256,uint256,uint256,uint8,bytes32,bytes32) (contracts/BigBangGame.sol#80-112):
        External calls:
        - nft.safeTransferFrom(address(this),msg.sender,player[msg.sender][_id]) (contracts/BigBangGame.sol#98)
        - nftId = factory.mint(msg.sender,_quality,_image) (contracts/BigBangGame.sol#104)
        State variables written after the call(s):
        - nonce[msg.sender] ++ (contracts/BigBangGame.sol#109)
Reentrancy in BigBangGame.importNft(uint256,uint256,uint8,bytes32,bytes32) (contracts/BigBangGame.sol#52-77):
        External calls:
        - nft.safeTransferFrom(msg.sender,address(this),_nftId) (contracts/BigBangGame.sol#69)
        State variables written after the call(s):
        - nonce[msg.sender] ++ (contracts/BigBangGame.sol#71)
        - player[msg.sender][_id] = _nftId (contracts/BigBangGame.sol#72)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

ERC721._mint(address,uint256) (openzeppelin/contracts/token/ERC721/ERC721.sol#333-344) ignores return value by _holderTokens[to].add(tokenId) (openzeppelin/contracts/token/ERC721/ERC721.sol#339)
ERC721._mint(address,uint256) (openzeppelin/contracts/token/ERC721/ERC721.sol#333-344) ignores return value by _tokenOwners.set(tokenId,to) (openzeppelin/contracts/token/ERC721/ERC721.sol#341)
ERC721._burn(uint256) (openzeppelin/contracts/token/ERC721/ERC721.sol#356-374) ignores return value by _holderTokens[owner].remove(tokenId) (openzeppelin/contracts/token/ERC721/ERC721.sol#369)
ERC721._burn(uint256) (openzeppelin/contracts/token/ERC721/ERC721.sol#356-374) ignores return value by _tokenOwners.remove(tokenId) (openzeppelin/contracts/token/ERC721/ERC721.sol#371)
ERC721._transfer(address,address,uint256) (openzeppelin/contracts/token/ERC721/ERC721.sol#387-402) ignores return value by _holderTokens[from].remove(tokenId) (openzeppelin/contracts/token/ERC721/ERC721.sol#396)
ERC721._transfer(address,address,uint256) (openzeppelin/contracts/token/ERC721/ERC721.sol#387-402) ignores return value by _holderTokens[to].add(tokenId) (openzeppelin/contracts/token/ERC721/ERC721.sol#397)
ERC721._transfer(address,address,uint256) (openzeppelin/contracts/token/ERC721/ERC721.sol#387-402) ignores return value by _tokenOwners.set(tokenId,to) (openzeppelin/contracts/token/ERC721/ERC721.sol#399)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

```
BigBangGame.setNftFactory(address).BBFactory (contracts/BigBangGame.sol#231) shadows:
        - BigBangGame.BBFactory (contracts/BigBangGame.sol#18) (state variable)
BigBangNFT.setOwner(address)._owner (contracts/BigBangNFT.sol#47) shadows:
        - Ownable._owner (openzeppelin/contracts/access/Ownable.sol#19) (state variable)
ERC721.constructor(string,string).name (openzeppelin/contracts/token/ERC721/ERC721.sol#93) shadows:
        - ERC721.name() (openzeppelin/contracts/token/ERC721/ERC721.sol#122-124) (function)
        - IERC721Metadata.name() (openzeppelin/contracts/token/ERC721/IERC721Metadata.sol#16) (function)
ERC721.constructor(string,string).symbol (openzeppelin/contracts/token/ERC721/ERC721.sol#93) shadows:
        - ERC721.symbol() (openzeppelin/contracts/token/ERC721/ERC721.sol#129-131) (function)
        - IERC721Metadata.symbol() (openzeppelin/contracts/token/ERC721/IERC721Metadata.sol#21) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

BigBangNFT.setFactory(address) (contracts/BigBangNFT.sol#51-53) should emit an event for:
        - factory = _factory (contracts/BigBangNFT.sol#52)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control

BigBangGame.constructor(address,address,address,address)._factory (contracts/BigBangGame.sol#38) lacks a zero-check on :
                - BBFactory = _factory (contracts/BigBangGame.sol#44)
BigBangNFT.setFactory(address)._factory (contracts/BigBangNFT.sol#51) lacks a zero-check on :
                - factory = _factory (contracts/BigBangNFT.sol#52)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in BigBangGame.exportNft(uint256,uint256,uint256,uint8,bytes32,bytes32) (contracts/BigBangGame.sol#80-112):
        External calls:
        - nft.safeTransferFrom(address(this),msg.sender,player[msg.sender][_id]) (contracts/BigBangGame.sol#98)
        State variables written after the call(s):
        - totalStake[msg.sender] -- (contracts/BigBangGame.sol#101)
Reentrancy in BigBangGame.importNft(uint256,uint256,uint8,bytes32,bytes32) (contracts/BigBangGame.sol#52-77):
        External calls:
        - nft.safeTransferFrom(msg.sender,address(this),_nftId) (contracts/BigBangGame.sol#69)
        State variables written after the call(s):
        - nftOwner[_id] = msg.sender (contracts/BigBangGame.sol#74)
        - totalStake[msg.sender] ++ (contracts/BigBangGame.sol#73)
Reentrancy in BigBangNFT.mint(address,uint256,uint256) (contracts/BigBangNFT.sol#34-45):
        External calls:
        - _safeMint(_to,_nftId) (contracts/BigBangNFT.sol#38)
                - returndata = to.functionCall(abi.encodeWithSelector(IERC721Receiver(to).onERC721Received.selector,_msgSender(),from,tokenId,_data),ERC721: transfer to non ERC721Receiver implementer) (openzeppelin/contracts/token/ERC721/ERC721.sol#441-447)
                - (success,returndata) = target.call{value: weiValue}(data) (openzeppelin/contracts/utils/Address.sol#123)
        External calls sending eth:
        - _safeMint(_to,_nftId) (contracts/BigBangNFT.sol#38)
                - (success,returndata) = target.call{value: weiValue}(data) (openzeppelin/contracts/utils/Address.sol#123)
        State variables written after the call(s):
        - paramsOf[_nftId] = Params(_quality,_image) (contracts/BigBangNFT.sol#40)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```

```
Reentrancy in BigBangGame.exportNft(uint256,uint256,uint256,uint8,bytes32,bytes32) (contracts/BigBangGame.sol#80-112):
        External calls:
        - nft.safeTransferFrom(address(this),msg.sender,player[msg.sender][_id]) (contracts/BigBangGame.sol#98)
        - nftId = factory.mint(msg.sender,_quality,_image) (contracts/BigBangGame.sol#104)
        Event emitted after the call(s):
        - ExportNft(_id,msg.sender,nftId,_quality,_image,block.timestamp) (contracts/BigBangGame.sol#111)
Reentrancy in BigBangGame.goldChangeToken(uint256,uint256,uint8,bytes32,bytes32) (contracts/BigBangGame.sol#129-154):
        External calls:
        - _safeTransfer(token[_typeId],msg.sender,_tokenAmount) (contracts/BigBangGame.sol#150)
                - _rewardToken.transfer(_to,_amount) (contracts/BigBangGame.sol#176)
        External calls sending eth:
        - _safeTransfer(token[_typeId],msg.sender,_tokenAmount) (contracts/BigBangGame.sol#150)
                - address(_to).transfer(_amount) (contracts/BigBangGame.sol#184)
        Event emitted after the call(s):
        - GoldChangeToken(msg.sender,_typeId,_gold,_tokenAmount,block.timestamp) (contracts/BigBangGame.sol#152)
Reentrancy in BigBangGame.importNft(uint256,uint256,uint8,bytes32,bytes32) (contracts/BigBangGame.sol#52-77):
        External calls:
        - nft.safeTransferFrom(msg.sender,address(this),_nftId) (contracts/BigBangGame.sol#69)
        Event emitted after the call(s):
        - ImportNft(_id,msg.sender,_nftId,block.timestamp) (contracts/BigBangGame.sol#76)
Reentrancy in BigBangNFT.mint(address,uint256,uint256) (contracts/BigBangNFT.sol#34-45):
        External calls:
        - _safeMint(_to,_nftId) (contracts/BigBangNFT.sol#38)
                - returndata = to.functionCall(abi.encodeWithSelector(IERC721Receiver(to).onERC721Received.selector,_msgSender(),from,tokenId,_data),ERC721: transfer to non ERC721Receiver
 implementer) (openzeppelin/contracts/token/ERC721/ERC721.sol#441-447)
                - (success,returndata) = target.call{value: weiValue}(data) (openzeppelin/contracts/utils/Address.sol#123)
        External calls sending eth:
        - _safeMint(_to,_nftId) (contracts/BigBangNFT.sol#38)
                - (success,returndata) = target.call{value: weiValue}(data) (openzeppelin/contracts/utils/Address.sol#123)
        Event emitted after the call(s):
        - Minted(_to,_nftId,_quality,_image,block.timestamp) (contracts/BigBangNFT.sol#43)
Reentrancy in BigBangGame.tokenChangeGold(uint256,uint256) (contracts/BigBangGame.sol#115-125):
        External calls:
        - _token.transferFrom(msg.sender,address(this),_amount) (contracts/BigBangGame.sol#122)
        Event emitted after the call(s):
        - TokenChangeGold(msg.sender,_typeId,_amount,block.timestamp) (contracts/BigBangGame.sol#124)
Reentrancy in BigBangGame.withdraw(address,uint256) (contracts/BigBangGame.sol#202-209):
        External calls:
        - _safeTransfer(_token,owner(),_amount) (contracts/BigBangGame.sol#206)
                - _rewardToken.transfer(_to,_amount) (contracts/BigBangGame.sol#176)
        External calls sending eth:
        - _safeTransfer(_token,owner(),_amount) (contracts/BigBangGame.sol#206)
                - address(_to).transfer(_amount) (contracts/BigBangGame.sol#184)
        Event emitted after the call(s):
        - Withdraw(_token,_amount,msg.sender,block.timestamp) (contracts/BigBangGame.sol#208)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```

```
BigBangGame.goldChangeToken(uint256,uint256,uint8,bytes32,bytes32) (contracts/BigBangGame.sol#129-154) uses assembly
        - INLINE ASM (contracts/BigBangGame.sol#134-136)
console._sendLogPayload(bytes) (node_modules/hardhat/console.sol#7-14) uses assembly
        - INLINE ASM (node_modules/hardhat/console.sol#10-13)
Address.isContract(address) (openzeppelin/contracts/utils/Address.sol#26-35) uses assembly
        - INLINE ASM (openzeppelin/contracts/utils/Address.sol#33)
Address._functionCallWithValue(address,bytes,uint256,string) (openzeppelin/contracts/utils/Address.sol#119-140) uses assembly
        - INLINE ASM (openzeppelin/contracts/utils/Address.sol#132-135)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity are used:
        - Version used: ['0.6.7', '>=0.4.22<0.9.0', '^0.6.0', '^0.6.2']
        - 0.6.7 (contracts/BigBangGame.sol#2)
        - 0.6.7 (contracts/BigBangNFT.sol#2)
        - 0.6.7 (contracts/BigBangNFTFactory.sol#2)
        - >=0.4.22<0.9.0 (node_modules/hardhat/console.sol#2)
        - ^0.6.0 (openzeppelin/contracts/GSN/Context.sol#3)
        - ^0.6.0 (openzeppelin/contracts/access/AccessControl.sol#3)
        - ^0.6.0 (openzeppelin/contracts/access/Ownable.sol#3)
        - ^0.6.0 (openzeppelin/contracts/introspection/ERC165.sol#3)
        - ^0.6.0 (openzeppelin/contracts/introspection/IERC165.sol#3)
        - ^0.6.0 (openzeppelin/contracts/math/SafeMath.sol#3)
        - ^0.6.0 (openzeppelin/contracts/token/ERC20/IERC20.sol#3)
        - ^0.6.0 (openzeppelin/contracts/token/ERC20/SafeERC20.sol#3)
        - ^0.6.0 (openzeppelin/contracts/token/ERC721/ERC721.sol#3)
        - ^0.6.0 (openzeppelin/contracts/token/ERC721/ERC721Burnable.sol#3)
        - ^0.6.2 (openzeppelin/contracts/token/ERC721/IERC721.sol#3)
        - ^0.6.2 (openzeppelin/contracts/token/ERC721/IERC721Enumerable.sol#3)
        - ^0.6.2 (openzeppelin/contracts/token/ERC721/IERC721Metadata.sol#3)
        - ^0.6.0 (openzeppelin/contracts/token/ERC721/IERC721Receiver.sol#3)
        - ^0.6.2 (openzeppelin/contracts/utils/Address.sol#3)
        - ^0.6.0 (openzeppelin/contracts/utils/Counters.sol#3)
        - ^0.6.0 (openzeppelin/contracts/utils/EnumerableMap.sol#3)
        - ^0.6.0 (openzeppelin/contracts/utils/EnumerableSet.sol#3)
        - ^0.6.0 (openzeppelin/contracts/utils/Strings.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
```

```
AccessControl._setRoleAdmin(bytes32,bytes32) (openzeppelin/contracts/access/AccessControl.sol#201-204) is never used and should be removed
Address.functionCall(address,bytes) (openzeppelin/contracts/utils/Address.sol#79-81) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (openzeppelin/contracts/utils/Address.sol#104-106) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (openzeppelin/contracts/utils/Address.sol#114-117) is never used and should be removed
Address.sendValue(address,uint256) (openzeppelin/contracts/utils/Address.sol#53-59) is never used and should be removed
Context._msgData() (openzeppelin/contracts/GSN/Context.sol#20-23) is never used and should be removed
Counters.decrement(Counters.Counter) (openzeppelin/contracts/utils/Counters.sol#37-39) is never used and should be removed
ERC721._setTokenURI(uint256,string) (openzeppelin/contracts/token/ERC721/ERC721.sol#411-414) is never used and should be removed
EnumerableMap._get(EnumerableMap.Map,bytes32) (openzeppelin/contracts/utils/EnumerableMap.sol#153-155) is never used and should be removed
EnumerableMap.get(EnumerableMap.UintToAddressMap,uint256) (openzeppelin/contracts/utils/EnumerableMap.sol#227-229) is never used and should be removed
EnumerableSet.contains(EnumerableSet.UintSet,uint256) (openzeppelin/contracts/utils/EnumerableSet.sol#219-221) is never used and should be removed
SafeERC20._callOptionalReturn(IERC20,bytes) (openzeppelin/contracts/token/ERC20/SafeERC20.sol#64-74) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (openzeppelin/contracts/token/ERC20/SafeERC20.sol#37-46) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (openzeppelin/contracts/token/ERC20/SafeERC20.sol#53-56) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (openzeppelin/contracts/token/ERC20/SafeERC20.sol#48-51) is never used and should be removed
SafeERC20.safeTransfer(IERC20,address,uint256) (openzeppelin/contracts/token/ERC20/SafeERC20.sol#22-24) is never used and should be removed
SafeERC20.safeTransferFrom(IERC20,address,address,uint256) (openzeppelin/contracts/token/ERC20/SafeERC20.sol#26-28) is never used and should be removed
SafeMath.add(uint256,uint256) (openzeppelin/contracts/math/SafeMath.sol#29-34) is never used and should be removed
SafeMath.div(uint256,uint256) (openzeppelin/contracts/math/SafeMath.sol#103-105) is never used and should be removed
SafeMath.div(uint256,uint256,string) (openzeppelin/contracts/math/SafeMath.sol#119-125) is never used and should be removed
SafeMath.mod(uint256,uint256) (openzeppelin/contracts/math/SafeMath.sol#139-141) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (openzeppelin/contracts/math/SafeMath.sol#155-158) is never used and should be removed
SafeMath.mul(uint256,uint256) (openzeppelin/contracts/math/SafeMath.sol#77-85) is never used and should be removed
SafeMath.sub(uint256,uint256) (openzeppelin/contracts/math/SafeMath.sol#46-48) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (openzeppelin/contracts/math/SafeMath.sol#60-65) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version0.6.7 (contracts/BigBangGame.sol#2) allows old versions
Pragma version0.6.7 (contracts/BigBangNFT.sol#2) allows old versions
Pragma version0.6.7 (contracts/BigBangNFTFactory.sol#2) allows old versions
Pragma version>=0.4.22<0.9.0 (node_modules/hardhat/console.sol#2) is too complex
Pragma version^0.6.0 (openzeppelin/contracts/GSN/Context.sol#3) allows old versions
Pragma version^0.6.0 (openzeppelin/contracts/access/AccessControl.sol#3) allows old versions
Pragma version^0.6.0 (openzeppelin/contracts/access/Ownable.sol#3) allows old versions
Pragma version^0.6.0 (openzeppelin/contracts/introspection/ERC165.sol#3) allows old versions
Pragma version^0.6.0 (openzeppelin/contracts/introspection/IERC165.sol#3) allows old versions
Pragma version^0.6.0 (openzeppelin/contracts/math/SafeMath.sol#3) allows old versions
Pragma version^0.6.0 (openzeppelin/contracts/token/ERC20/IERC20.sol#3) allows old versions
Pragma version^0.6.0 (openzeppelin/contracts/token/ERC20/SafeERC20.sol#3) allows old versions
Pragma version^0.6.0 (openzeppelin/contracts/token/ERC721/ERC721.sol#3) allows old versions
Pragma version^0.6.0 (openzeppelin/contracts/token/ERC721/ERC721Burnable.sol#3) allows old versions
Pragma version^0.6.2 (openzeppelin/contracts/token/ERC721/IERC721.sol#3) allows old versions
Pragma version^0.6.2 (openzeppelin/contracts/token/ERC721/IERC721Enumerable.sol#3) allows old versions
Pragma version^0.6.2 (openzeppelin/contracts/token/ERC721/IERC721Metadata.sol#3) allows old versions
Pragma version^0.6.0 (openzeppelin/contracts/token/ERC721/IERC721Receiver.sol#3) allows old versions
Pragma version^0.6.2 (openzeppelin/contracts/utils/Address.sol#3) allows old versions
Pragma version^0.6.0 (openzeppelin/contracts/utils/Counters.sol#3) allows old versions
Pragma version^0.6.0 (openzeppelin/contracts/utils/EnumerableMap.sol#3) allows old versions
Pragma version^0.6.0 (openzeppelin/contracts/utils/EnumerableSet.sol#3) allows old versions
Pragma version^0.6.0 (openzeppelin/contracts/utils/Strings.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
```

AUTOMATED TESTING

```
Low level call in Address.sendValue(address,uint256) (openzeppelin/contracts/utils/Address.sol#53-59):
        - (success) = recipient.call{value: amount}() (openzeppelin/contracts/utils/Address.sol#57)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (openzeppelin/contracts/utils/Address.sol#119-140):
        - (success,returndata) = target.call{value: weiValue}(data) (openzeppelin/contracts/utils/Address.sol#123)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter BigBangGame.importNft(uint256,uint256,uint8,bytes32,bytes32)._id (contracts/BigBangGame.sol#52) is not in mixedCase
Parameter BigBangGame.importNft(uint256,uint256,uint8,bytes32,bytes32)._nftId (contracts/BigBangGame.sol#52) is not in mixedCase
Parameter BigBangGame.importNft(uint256,uint256,uint8,bytes32,bytes32)._v (contracts/BigBangGame.sol#52) is not in mixedCase
Parameter BigBangGame.importNft(uint256,uint256,uint8,bytes32,bytes32)._r (contracts/BigBangGame.sol#52) is not in mixedCase
Parameter BigBangGame.importNft(uint256,uint256,uint8,bytes32,bytes32)._s (contracts/BigBangGame.sol#52) is not in mixedCase
Parameter BigBangGame.exportNft(uint256,uint256,uint256,uint8,bytes32,bytes32)._id (contracts/BigBangGame.sol#80) is not in mixedCase
Parameter BigBangGame.exportNft(uint256,uint256,uint256,uint8,bytes32,bytes32)._quality (contracts/BigBangGame.sol#80) is not in mixedCase
Parameter BigBangGame.exportNft(uint256,uint256,uint256,uint8,bytes32,bytes32)._image (contracts/BigBangGame.sol#80) is not in mixedCase
Parameter BigBangGame.exportNft(uint256,uint256,uint256,uint8,bytes32,bytes32)._v (contracts/BigBangGame.sol#80) is not in mixedCase
Parameter BigBangGame.exportNft(uint256,uint256,uint256,uint8,bytes32,bytes32)._r (contracts/BigBangGame.sol#80) is not in mixedCase
Parameter BigBangGame.exportNft(uint256,uint256,uint256,uint8,bytes32,bytes32)._s (contracts/BigBangGame.sol#80) is not in mixedCase
Parameter BigBangGame.tokenChangeGold(uint256,uint256)._typeId (contracts/BigBangGame.sol#115) is not in mixedCase
Parameter BigBangGame.tokenChangeGold(uint256,uint256)._amount (contracts/BigBangGame.sol#115) is not in mixedCase
Parameter BigBangGame.goldChangeToken(uint256,uint256,uint8,bytes32,bytes32)._gold (contracts/BigBangGame.sol#129) is not in mixedCase
Parameter BigBangGame.goldChangeToken(uint256,uint256,uint8,bytes32,bytes32)._typeId (contracts/BigBangGame.sol#129) is not in mixedCase
Parameter BigBangGame.goldChangeToken(uint256,uint256,uint8,bytes32,bytes32)._v (contracts/BigBangGame.sol#129) is not in mixedCase
Parameter BigBangGame.goldChangeToken(uint256,uint256,uint8,bytes32,bytes32)._r (contracts/BigBangGame.sol#129) is not in mixedCase
Parameter BigBangGame.goldChangeToken(uint256,uint256,uint8,bytes32,bytes32)._s (contracts/BigBangGame.sol#129) is not in mixedCase
Parameter BigBangGame.checkToken(uint256)._typeId (contracts/BigBangGame.sol#157) is not in mixedCase
Parameter BigBangGame.withdraw(address,uint256)._token (contracts/BigBangGame.sol#202) is not in mixedCase
Parameter BigBangGame.withdraw(address,uint256)._amount (contracts/BigBangGame.sol#202) is not in mixedCase
Parameter BigBangGame.addToken(address)._token (contracts/BigBangGame.sol#212) is not in mixedCase
Parameter BigBangGame.setScale(uint256)._ratio (contracts/BigBangGame.sol#223) is not in mixedCase
Parameter BigBangGame.setNftFactory(address)._factoryAddress (contracts/BigBangGame.sol#229) is not in mixedCase
Variable BigBangGame.BBNft (contracts/BigBangGame.sol#17) is not in mixedCase
Variable BigBangGame.BBFactory (contracts/BigBangGame.sol#18) is not in mixedCase
Parameter BigBangNFT.mint(address,uint256,uint256)._to (contracts/BigBangNFT.sol#34) is not in mixedCase
Parameter BigBangNFT.mint(address,uint256,uint256)._quality (contracts/BigBangNFT.sol#34) is not in mixedCase
Parameter BigBangNFT.mint(address,uint256,uint256)._image (contracts/BigBangNFT.sol#34) is not in mixedCase
Parameter BigBangNFT.setOwner(address)._owner (contracts/BigBangNFT.sol#47) is not in mixedCase
Parameter BigBangNFT.setFactory(address)._factory (contracts/BigBangNFT.sol#51) is not in mixedCase
Parameter BigBangNFT.setBaseUri(string)._uri (contracts/BigBangNFT.sol#55) is not in mixedCase
Parameter BigBangNFTFactory.mint(address,uint256,uint256)._owner (contracts/BigBangNFTFactory.sol#34) is not in mixedCase
Parameter BigBangNFTFactory.mint(address,uint256,uint256)._quality (contracts/BigBangNFTFactory.sol#34) is not in mixedCase
Parameter BigBangNFTFactory.mint(address,uint256,uint256)._image (contracts/BigBangNFTFactory.sol#34) is not in mixedCase
Parameter BigBangNFTFactory.setNft(address)._nft (contracts/BigBangNFTFactory.sol#42) is not in mixedCase
Contract console (node_modules/hardhat/console.sol#4-1532) is not in CapWords
Parameter ERC721.safeTransferFrom(address,address,uint256,bytes)._data (openzeppelin/contracts/token/ERC721/ERC721.sol#245) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (openzeppelin/contracts/GSN/Context.sol#21)" inContext (openzeppelin/contracts/GSN/Context.sol#15-24)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

Reentrancy in BigBangGame.goldChangeToken(uint256,uint256,uint8,bytes32,bytes32) (contracts/BigBangGame.sol#129-154):
        External calls:
        - _safeTransfer(token[_typeId],msg.sender,_tokenAmount) (contracts/BigBangGame.sol#150)
                - address(_to).transfer(_amount) (contracts/BigBangGame.sol#184)
        Event emitted after the call(s):
        - GoldChangeToken(msg.sender,_typeId,_gold,_tokenAmount,block.timestamp) (contracts/BigBangGame.sol#152)
Reentrancy in BigBangGame.withdraw(address,uint256) (contracts/BigBangGame.sol#202-209):
        External calls:
        - _safeTransfer(_token,owner(),_amount) (contracts/BigBangGame.sol#206)
                - address(_to).transfer(_amount) (contracts/BigBangGame.sol#184)
        Event emitted after the call(s):
        - Withdraw(_token,_amount,msg.sender,block.timestamp) (contracts/BigBangGame.sol#208)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4

console.slitherConstructorConstantVariables() (node_modules/hardhat/console.sol#4-1532) uses literals with too many digits:
        - CONSOLE_ADDRESS = address(0x000000000000000000636F6e736F6c652e6c6f67) (node_modules/hardhat/console.sol#5)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

withdraw(address,uint256) should be declared external:
        - BigBangGame.withdraw(address,uint256) (contracts/BigBangGame.sol#202-209)
addToken(address) should be declared external:
        - BigBangGame.addToken(address) (contracts/BigBangGame.sol#212-220)
setScale(uint256) should be declared external:
        - BigBangGame.setScale(uint256) (contracts/BigBangGame.sol#223-226)
mint(address,uint256,uint256) should be declared external:
        - BigBangNFT.mint(address,uint256,uint256) (contracts/BigBangNFT.sol#34-45)
setOwner(address) should be declared external:
        - BigBangNFT.setOwner(address) (contracts/BigBangNFT.sol#47-49)
setFactory(address) should be declared external:
        - BigBangNFT.setFactory(address) (contracts/BigBangNFT.sol#51-53)
setBaseUri(string) should be declared external:
        - BigBangNFT.setBaseUri(string) (contracts/BigBangNFT.sol#55-57)
mint(address,uint256,uint256) should be declared external:
        - BigBangNFTFactory.mint(address,uint256,uint256) (contracts/BigBangNFTFactory.sol#34-37)
setNft(address) should be declared external:
        - BigBangNFTFactory.setNft(address) (contracts/BigBangNFTFactory.sol#42-44)
addAdmin(address) should be declared external:
        - BigBangNFTFactory.addAdmin(address) (contracts/BigBangNFTFactory.sol#47-50)
renounceAdmin() should be declared external:
        - BigBangNFTFactory.renounceAdmin() (contracts/BigBangNFTFactory.sol#53-56)
addGenerator(address) should be declared external:
        - BigBangNFTFactory.addGenerator(address) (contracts/BigBangNFTFactory.sol#85-88)
removeGenerator(address) should be declared external:
        - BigBangNFTFactory.removeGenerator(address) (contracts/BigBangNFTFactory.sol#91-94)
getRoleMemberCount(bytes32) should be declared external:
        - AccessControl.getRoleMemberCount(bytes32) (openzeppelin/contracts/access/AccessControl.sol#95-97)
getRoleMember(bytes32,uint256) should be declared external:
        - AccessControl.getRoleMember(bytes32,uint256) (openzeppelin/contracts/access/AccessControl.sol#111-113)
getRoleAdmin(bytes32) should be declared external:
        - AccessControl.getRoleAdmin(bytes32) (openzeppelin/contracts/access/AccessControl.sol#121-123)
getRoleAdmin(bytes32) should be declared external:
        - AccessControl.getRoleAdmin(bytes32) (openzeppelin/contracts/access/AccessControl.sol#121-123)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (openzeppelin/contracts/access/Ownable.sol#54-57)
supportsInterface(bytes4) should be declared external:
        - ERC165.supportsInterface(bytes4) (openzeppelin/contracts/introspection/ERC165.sol#35-37)
balanceOf(address) should be declared external:
        - ERC721.balanceOf(address) (openzeppelin/contracts/token/ERC721/ERC721.sol#106-110)
name() should be declared external:
        - ERC721.name() (openzeppelin/contracts/token/ERC721/ERC721.sol#122-124)
symbol() should be declared external:
        - ERC721.symbol() (openzeppelin/contracts/token/ERC721/ERC721.sol#129-131)
tokenURI(uint256) should be declared external:
        - ERC721.tokenURI(uint256) (openzeppelin/contracts/token/ERC721/ERC721.sol#136-151)
baseURI() should be declared external:
        - ERC721.baseURI() (openzeppelin/contracts/token/ERC721/ERC721.sol#158-160)
tokenOfOwnerByIndex(address,uint256) should be declared external:
        - ERC721.tokenOfOwnerByIndex(address,uint256) (openzeppelin/contracts/token/ERC721/ERC721.sol#165-167)
totalSupply() should be declared external:
        - ERC721.totalSupply() (openzeppelin/contracts/token/ERC721/ERC721.sol#172-175)
tokenByIndex(uint256) should be declared external:
        - ERC721.tokenByIndex(uint256) (openzeppelin/contracts/token/ERC721/ERC721.sol#180-183)
approve(address,uint256) should be declared external:
        - ERC721.approve(address,uint256) (openzeppelin/contracts/token/ERC721/ERC721.sol#188-197)
setApprovalForAll(address,bool) should be declared external:
        - ERC721.setApprovalForAll(address,bool) (openzeppelin/contracts/token/ERC721/ERC721.sol#211-216)
transferFrom(address,address,uint256) should be declared external:
        - ERC721.transferFrom(address,address,uint256) (openzeppelin/contracts/token/ERC721/ERC721.sol#228-233)
safeTransferFrom(address,address,uint256) should be declared external:
        - ERC721.safeTransferFrom(address,address,uint256) (openzeppelin/contracts/token/ERC721/ERC721.sol#238-240)
burn(uint256) should be declared external:
        - ERC721Burnable.burn(uint256) (openzeppelin/contracts/token/ERC721/ERC721Burnable.sol#20-24)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
. analyzed (23 contracts with 78 detectors), 159 result(s) found
```

- No major issues found by Slither.

4.2 AUTOMATED SECURITY SCAN

Description:

Halborn used automated security scanners to assist with detection of well-known security issues and to identify low-hanging fruits on the targets for this engagement. Among the tools used was MythX, a security analysis service for Ethereum smart contracts. MythX performed a scan on the smart contracts and sent the compiled results to the analyzers in order to locate any vulnerabilities.

MythX results:

- No issues found by MythX.

THANK YOU FOR CHOOSING

// HALBORN