SHELLBOXES

Gnars V2

Smart Contract Security Audit

Prepared by ShellBoxes

April 25th, 2022 - April 29th, 2022

Shellboxes.com

contact@shellboxes.com

Document Properties

Client	Gnars V2
Version	1.0
Classification	Public

Scope

The Gnars V2 Contracts in the Gnars V2 Repository

Repo	Commit Hash
`https://github.com/artdothaus/gnars-contracts`	5cc1793946938c439d952730c6156c10562a9d08

Re-Audit

Contract Name	Contract Address
SkateContractV2	0x558BFFF0D583416f7C4e380625c7865821b8E95C
SkateContractV2AuctionHouse	0x46e57901ea1edd8a200cf982af4fd8cc3a957974

Contacts

COMPANY	EMAIL
ShellBoxes	contact@shellboxes.com

Contents

1 Introduction

Bitlabs.dev engaged ShellBoxes to conduct a security assessment on the Gnars V2 beginning on April 25th, 2022 and ending April 29th, 2022. In this report, we detail our methodical approach to evaluate potential security issues associated with the implementation of smart contracts, by exposing possible semantic discrepancies between the smart contract code and design document, and by recommending additional ideas to optimize the existing code. Our findings indicate that the current version of smart contracts can still be enhanced further due to the presence of many security and performance concerns.

This document summarizes the findings of our audit.

1.1 About Bitlabs.dev

Bitlabs is contributing to decentralised ecosystems by developing software products. The most known brand from Bitlabs is art.haus, which is one of the leading art curators in the NFT space.

Issuer	Bitlabs.dev
Website	https://bitlabs.dev
Type	Solidity Smart Contract
Audit Method	Whitebox

1.2 Approach & Methodology

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

1.2.1 Risk Methodology

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

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

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

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

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

2 Findings Overview

2.1 Summary

The following is a synopsis of our conclusions from our analysis of the Gnars V2 implementation. During the first part of our audit, we examine the smart contract source code and run the codebase via a static code analyzer. The objective here is to find known coding problems statically and then manually check (reject or confirm) issues highlighted by the tool. Additionally, we check business logics, system processes, and DeFi-related components manually to identify potential hazards and/or defects.

2.2 Key Findings

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

Vulnerabilities	Severity	Status
Missing Transfer Verification	MEDIUM	Fixed
Missing Value Verification	LOW	Fixed
Missing Address Verification	LOW	Fixed
Renounce Ownership	LOW	Acknowledged
Floating Pragma	LOW	Fixed
Missing Address Verification	LOW	Fixed
Renounce Ownership	LOW	Acknowledged
Floating Pragma	LOW	Fixed

3 Findings Details

A SkateContractV2AuctionHouse.sol

A.1 Missing Transfer Verification [MEDIUM]

Description:

The ERC20 standard token implementation functions return the transaction status as a Boolean. It is good practice to check for the return status of the function call to ensure that the transaction was successful. It is the developer's responsibility to enclose these function calls with require() to ensure that, when the intended ERC20 function call returns false , the caller transaction also fails. However, it is mostly missed by developers when they carry out checks in effect, the transaction would always succeed, even if the token transfer did not.

Code:

Listing 1: SkateContractV2AuctionHouse (Line 308)

```
function _safeTransferETHWithFallback(address to, uint256 amount) internal {
    if (!_safeTransferETH(to, amount)) {
        IWETH(weth).deposit{value: amount}();
        IERC20(weth).transfer(to, amount);
    }
}
```

Recommendation:

It is recommended to put the transfer call inside an assert or require to verify that the transfer has passed successfully.

Status - Fixed

The issue was fixed by adding a require in the transfer function.

Code:

Listing 2: SkateContractV2AuctionHouse (Line 280)

```
    function _safeTransferETHWithFallback(address to, uint256 amount)
    internal {
        if (!_safeTransferETH(to, amount)) {
            IWETH(weth).deposit{value: amount}();
            require(IERC20(weth).transfer(to, amount), "Transfer failed");
        }
    }
```

A.2 Missing Value Verification [LOW]

Description:

Certain functions lack a safety check in the values, the values of the arguments should be verified to allow only the ones that go with the contract's logic.

- The `_reservePrice` variable should be different from zero.

- The `_minBidIncrementPercentage` variable should be less than 100%.

Code:

Listing 3: SkateContractV2AuctionHouse (Line 62)

```
function initialize(
    address _skate,
    address _dao,
    ISkateContractV2 _gnars,
    address _weth,
    uint256 _reservePrice,
    uint8 _minBidIncrementPercentage
) external initializer {
    __Pausable_init();
    __ReentrancyGuard_init();
    __Ownable_init();
    __UUPSUpgradeable_init();

    _pause();

    skate = _skate;
    dao = _dao;
    gnars = _gnars;
    weth = _weth;
    reservePrice = _reservePrice;
    minBidIncrementPercentage = _minBidIncrementPercentage;
    auctionPeriodBlocks = 666;
}
```

Recommendation:

It is recommended to verify the values provided in the arguments. The concerns can be resolved by utilizing a require statement.

Status - Fixed

The issue was fixed by validating the following variable `_reservePrice` and `_minBidIncrementPercentage` .

Code:

Listing 4: SkateContractV2AuctionHouse (Line 90)

```
    function initialize(
        address _skate,
        address _dao,
        ISkateContractV2 _gnars,
        address _weth,
        uint256 _reservePrice,
        uint8 _minBidIncrementPercentage,
        uint256 _baseAuctionTime,
        uint256 _timeDoublingCount
    ) external initializer {
        __Pausable_init();
        __ReentrancyGuard_init();
        __Ownable_init();
        __UUPSUpgradeable_init();

        _pause();

        require(
            _skate != address(0) && _dao != address(0) &&
            address(_gnars) != address(0) && _weth != address(0),
            "ZERO ADDRESS"
        );
        require(_reservePrice > 0, "Reserve price is zero");
        require(_minBidIncrementPercentage <= 100, "Min bid increment
         percentange too high");
        require(_baseAuctionTime > 0, "Base auction time is zero");
        require(_timeDoublingCount > 0, "Time doubling count is zero");
  }
```

A.3 Missing Address Verification [LOW]

Description:

Certain functions lack a safety check in the address, the address-type argument should include a zero-address test, otherwise, some of the contract's functionality may become inaccessible. The `_skate`, `_dao`, `_gnars` and `_weth` addresses should be verified to be different from `address(0)`.

Code:

Listing 5: SkateContractV2AuctionHouse (Line 62)

```
function initialize(
   address _skate,
   address _dao,
   ISkateContractV2 _gnars,
   address _weth,
   uint256 _reservePrice,
   uint8 _minBidIncrementPercentage
) external initializer {
   __Pausable_init();
   __ReentrancyGuard_init();
   __Ownable_init();
   __UUPSUpgradeable_init();

   _pause();

   skate = _skate;
   dao = _dao;
   gnars = _gnars;
   weth = _weth;
   reservePrice = _reservePrice;
   minBidIncrementPercentage = _minBidIncrementPercentage;
   auctionPeriodBlocks = 666;
}
```

Recommendation:

It is recommended to verify that the addresses provided in the arguments are different from the `address(0)` .

Status - Fixed

The issue was fixed by validating the following variables `_skate` , `_dao` , `_gnars` and `_weth` .

Code:

Listing 6: SkateContractV2AuctionHouse (Line 90)

```
   function initialize(
       address _skate,
       address _dao,
       ISkateContractV2 _gnars,
       address _weth,
       uint256 _reservePrice,
       uint8 _minBidIncrementPercentage,
       uint256 _baseAuctionTime,
       uint256 _timeDoublingCount
   ) external initializer {
       __Pausable_init();
       __ReentrancyGuard_init();
       __Ownable_init();
       __UUPSUpgradeable_init();

       _pause();

       require(
           _skate != address(0) && _dao != address(0) &&
           address(_gnars) != address(0) && _weth != address(0),
           "ZERO ADDRESS"
       );
       require(_reservePrice > 0, "Reserve price is zero");
       require(_minBidIncrementPercentage <= 100, "Min bid increment
        percentange too high");
       require(_baseAuctionTime > 0, "Base auction time is zero");
       require(_timeDoublingCount > 0, "Time doubling count is zero");
}
```

A.4 Renounce Ownership [LOW]

Description:

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

Code:

Listing 7: SkateContractV2AuctionHouse (Line 24)

```
contract SkateContractV2AuctionHouse is
   ISkateContractV2AuctionHouse,
   PausableUpgradeable,
   ReentrancyGuardUpgradeable,
   OwnableUpgradeable,
   UUPSUpgradeable
{
```

Recommendation:

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

Status - Acknowledged

The team decided to acknowledge the risk.

A.5 Floating Pragma [LOW]

Description:

The contract makes use of the floating-point pragma 0.8.6 . Contracts should be deployed using the same compiler version and flags that were used during the testing process. Locking the pragma helps ensure that contracts are not unintentionally deployed using another pragma, such as an obsolete version, that may introduce issues in the contract system.

Code:

Listing 8: SkateContractV2AuctionHouse (Line 13)

```
pragma solidity ^0.8.6;
```

Recommendation:

Consider locking the pragma version. It is advised that floating pragma should not be used in production. Both truffle-config.js and hardhat.config.js support locking the pragma version.

Status - Fixed

The issue has been fixed by locking the pragma version to 0.8.6.

B SkateContractV2.sol

B.1 Missing Address Verification [LOW]

Description:

Certain functions lack a safety check in the address, the address-type argument should include a zero-address test, otherwise, some of the contract's functionality may become inaccessible. The `_minter` , `_descriptor` and `_seeder` variables should be verified to be different from the `address(0)` .

Code:

Listing 9: SkateContractV2 (Line 79)

```
constructor(
   address _minter,
   IGnarDescriptor _descriptor,
   IGnarSeeder _seeder,
   uint256 initialGnarId
) ERC721("Skate or DAO", "GNAR") {
   minter = _minter;
   descriptor = _descriptor;
   seeder = _seeder;
   _currentGnarId = initialGnarId;
}
```

Listing 10: SkateContractV2 (Line 127)

```
function setMinter(address _minter)
   external
   override
   onlyOwner
   whenMinterNotLocked
{
   minter = _minter;

   emit MinterUpdated(_minter);
}
```

Listing 11: SkateContractV2 (Line 152)

```
function setDescriptor(IGnarDescriptor _descriptor)
   external
   override
   onlyOwner
   whenDescriptorNotLocked
{
   descriptor = _descriptor;

   emit DescriptorUpdated(_descriptor);
}
```

Listing 12: SkateContractV2 (Line 182)

```
function setSeeder(IGnarSeeder _seeder)
    external
    override
    onlyOwner
    whenSeederNotLocked
{
    seeder = _seeder;

    emit SeederUpdated(_seeder);
}
```

Recommendation:

It is recommended to verify that the addresses provided in the arguments are different from the address(0) .

Status - Fixed

The issue was fixed by validating the following variables _minter , _descriptor and _seeder .

Code:

Listing 13: SkateContractV2 (Line 102)

```
    constructor(
        address _noundersDAO,
        address _minter,
        IGnarDescriptorV2 _descriptor,
        IGnarSeederV2 _seeder,
        IProxyRegistry _proxyRegistry,
        uint256 _initialGnarId
    ) ERC721("Gnars", "GNAR") {
        require(
            _noundersDAO != address(0) &&
                _minter != address(0) &&
                address(_descriptor) != address(0) &&
                address(_seeder) != address(0) &&
                address(_proxyRegistry) != address(0),
            "ZERO ADDRESS"
        );
```

B.2 Renounce Ownership [LOW]

Description:

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

Code:

Listing 14: SkateContractV2.sol (Line 22)

```
contract SkateContractV2 is ISkateContractV2, Ownable, ERC721Enumerable {
```

Recommendation:

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

Status - Acknowledged

The team decided to acknowledge the risk.

B.3 Floating Pragma [LOW]

Description:

The contract makes use of the floating-point pragma 0.8.6 . Contracts should be deployed using the same compiler version and flags that were used during the testing process.Locking the pragma helps ensure that contracts are not unintentionally deployed using another pragma, such as an obsolete version, that may introduce issues in the contract system.

Code:

Listing 15: SkateContractV2 (Line 12)

```
pragma solidity ^0.8.6;
```

Recommendation:

Consider locking the pragma version. It is advised that floating pragma should not be used in production. Both truffle-config.js and hardhat.config.js support locking the pragma version.

Status - Fixed

The issue has been fixed by locking the pragma version to 0.8.6.

4 Best Practices

BP.1 Low-level function calls

Description:

The low-level function call is being utilized in this contract, which can introduce several unexpected risks or errors. Even though we did not find any potential hacks via this low-level function for this contract. Nevertheless, please ensure to avoid using low-level calls.

Code:

Listing 16: SkateContractV2AuctionHouse (Line 319)

```
function _safeTransferETH(address to, uint256 value)
    internal
    returns (bool)
{
    (bool success, ) = to.call{value: value, gas: 30_000}(new bytes(0));
    return success;
}
```

5 Static Analysis (Slither)

Description:

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

Results:

```
ERC721._checkOnERC721Received(address,address,uint256,bytes) (../../openzep
pelin-contracts/contracts/token/ERC721/ERC721.sol#369-390) ignores return v
alue by IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,_dat
a) (../../openzeppelin-contracts/contracts/token/ERC721/ERC721.sol#376-386)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#un
used-return

IGnarDescriptor.addManyBackgrounds(string[]).backgrounds (../interfaces/IGN
ARDescriptor.sol#49) shadows:
        - IGnarDescriptor.backgrounds(uint256) (../interfaces/IGNARDescript
or.sol#24) (function)
IGnarDescriptor.addManyBodies(bytes[]).bodies (../interfaces/IGNARDescripto
r.sol#51) shadows:
        - IGnarDescriptor.bodies(uint256) (../interfaces/IGNARDescriptor.so
l#26) (function)
IGnarDescriptor.addManyAccessories(bytes[]).accessories (../interfaces/IGNA
RDescriptor.sol#53) shadows:
        - IGnarDescriptor.accessories(uint256) (../interfaces/IGNARDescript
or.sol#28) (function)
IGnarDescriptor.addManyHeads(bytes[]).heads (../interfaces/IGNARDescriptor.
sol#55) shadows:
	- IGnarDescriptor.heads(uint256) (../interfaces/IGNARDescriptor.sol
#30) (function)
IGnarDescriptor.addManyGlasses(bytes[]).glasses (../interfaces/IGNARDescrip
tor.sol#57) shadows:
	- IGnarDescriptor.glasses(uint256) (../interfaces/IGNARDescriptor.s
ol#32) (function)
IGnarDescriptor.addGlasses(bytes).glasses (../interfaces/IGNARDescriptor.so
l#70) shadows:
	- IGnarDescriptor.glasses(uint256) (../interfaces/IGNARDescriptor.s
ol#32) (function)
IGnarDescriptor.setBaseURI(string).baseURI (../interfaces/IGNARDescriptor.s
ol#76) shadows:
	- IGnarDescriptor.baseURI() (../interfaces/IGNARDescriptor.sol#17)
(function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#lo
cal-variable-shadowing

SkateContractV2.constructor(address,IGnarDescriptor,IGnarSeeder,uint256)._m
inter (SkateContractV2.sol#80) lacks a zero-check on :
		- minter = _minter (SkateContractV2.sol#85)
SkateContractV2.setMinter(address)._minter (SkateContractV2.sol#127) lacks
a zero-check on :
		- minter = _minter (SkateContractV2.sol#133)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#mi
ssing-zero-address-validation

Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).retv
al (../../openzeppelin-contracts/contracts/token/ERC721/ERC721.sol#376)' in
 ERC721._checkOnERC721Received(address,address,uint256,bytes) (../../openze
ppelin-contracts/contracts/token/ERC721/ERC721.sol#369-390) potentially use
d before declaration: retval == IERC721Receiver.onERC721Received.selector (
../../openzeppelin-contracts/contracts/token/ERC721/ERC721.sol#377)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reas
on (../../openzeppelin-contracts/contracts/token/ERC721/ERC721.sol#378)' in
```

```
 ERC721._checkOnERC721Received(address,address,uint256,bytes) (../../openze
ppelin-contracts/contracts/token/ERC721/ERC721.sol#369-390) potentially use
d before declaration: reason.length == 0 (../../openzeppelin-contracts/cont
racts/token/ERC721/ERC721.sol#379)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reas
on (../../openzeppelin-contracts/contracts/token/ERC721/ERC721.sol#378)' in
 ERC721._checkOnERC721Received(address,address,uint256,bytes) (../../openze
ppelin-contracts/contracts/token/ERC721/ERC721.sol#369-390) potentially use
d before declaration: revert(uint256,uint256)(32 + reason,mload(uint256)(re
ason)) (../../openzeppelin-contracts/contracts/token/ERC721/ERC721.sol#383)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pr
e-declaration-usage-of-local-variables

ERC721._checkOnERC721Received(address,address,uint256,bytes) (../../openzep
pelin-contracts/contracts/token/ERC721/ERC721.sol#369-390) uses assembly
       - INLINE ASM (../../openzeppelin-contracts/contracts/token/ERC721/E
RC721.sol#382-384)
Address.isContract(address) (../../openzeppelin-contracts/contracts/utils/A
ddress.sol#26-36) uses assembly
       - INLINE ASM (../../openzeppelin-contracts/contracts/utils/Address.
sol#32-34)
Address.verifyCallResult(bool,bytes,string) (../../openzeppelin-contracts/c
ontracts/utils/Address.sol#195-215) uses assembly
       - INLINE ASM (../../openzeppelin-contracts/contracts/utils/Address.
sol#207-210)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#as
sembly-usage

Different versions of Solidity is used:
       - Version used: ['^0.8.0', '^0.8.6']
       - ^0.8.6 (../interfaces/IGNARDescriptor.sol#2)
       - ^0.8.6 (../interfaces/IGNARSeeder.sol#2)
       - ^0.8.6 (../interfaces/ISkateContractV2.sol#18)
       - ^0.8.0 (../../openzeppelin-contracts/contracts/access/Ownable.sol
#3)
        - ^0.8.0 (../../openzeppelin-contracts/contracts/token/ERC721/ERC72
1.sol#3)
        - ^0.8.0 (../../openzeppelin-contracts/contracts/token/ERC721/IERC7
21.sol#3)
        - ^0.8.0 (../../openzeppelin-contracts/contracts/token/ERC721/IERC7
21Receiver.sol#3)
        - ^0.8.0 (../../openzeppelin-contracts/contracts/token/ERC721/exten
sions/ERC721Enumerable.sol#3)
        - ^0.8.0 (../../openzeppelin-contracts/contracts/token/ERC721/exten
sions/IERC721Enumerable.sol#3)
        - ^0.8.0 (../../openzeppelin-contracts/contracts/token/ERC721/exten
sions/IERC721Metadata.sol#3)
        - ^0.8.0 (../../openzeppelin-contracts/contracts/utils/Address.sol#
3)
        - ^0.8.0 (../../openzeppelin-contracts/contracts/utils/Context.sol#
3)
        - ^0.8.0 (../../openzeppelin-contracts/contracts/utils/Strings.sol#
3)
        - ^0.8.0 (../../openzeppelin-contracts/contracts/utils/introspectio
n/ERC165.sol#3)
        - ^0.8.0 (../../openzeppelin-contracts/contracts/utils/introspectio
n/IERC165.sol#3)
        - ^0.8.6 (SkateContractV2.sol#12)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#di
fferent-pragma-directives-are-used

Address.functionCall(address,bytes) (../../openzeppelin-contracts/contracts
/utils/Address.sol#79-81) is never used and should be removed
Address.functionCall(address,bytes,string) (../../openzeppelin-contracts/co
ntracts/utils/Address.sol#89-95) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (../../openzeppelin-co
ntracts/contracts/utils/Address.sol#108-114) is never used and should be re
moved
```

```
Address.functionCallWithValue(address,bytes,uint256,string) (../../openzepp
elin-contracts/contracts/utils/Address.sol#122-133) is never used and shoul
d be removed
Address.functionDelegateCall(address,bytes) (../../openzeppelin-contracts/c
ontracts/utils/Address.sol#168-170) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (../../openzeppelin-cont
racts/contracts/utils/Address.sol#178-187) is never used and should be remo
ved
Address.functionStaticCall(address,bytes) (../../openzeppelin-contracts/con
tracts/utils/Address.sol#141-143) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (../../openzeppelin-contra
cts/contracts/utils/Address.sol#151-160) is never used and should be remove
d
Address.sendValue(address,uint256) (../../openzeppelin-contracts/contracts/
utils/Address.sol#54-59) is never used and should be removed
Address.verifyCallResult(bool,bytes,string) (../../openzeppelin-contracts/c
ontracts/utils/Address.sol#195-215) is never used and should be removed
Context._msgData() (../../openzeppelin-contracts/contracts/utils/Context.so
l#20-22) is never used and should be removed
ERC721._baseURI() (../../openzeppelin-contracts/contracts/token/ERC721/ERC7
21.sol#104-106) is never used and should be removed
ERC721._safeMint(address,uint256) (../../openzeppelin-contracts/contracts/t
oken/ERC721/ERC721.sol#250-252) is never used and should be removed
ERC721._safeMint(address,uint256,bytes) (../../openzeppelin-contracts/contr
acts/token/ERC721/ERC721.sol#258-268) is never used and should be removed
Strings.toHexString(uint256) (../../openzeppelin-contracts/contracts/utils/
Strings.sol#39-50) is never used and should be removed
Strings.toHexString(uint256,uint256) (../../openzeppelin-contracts/contract
s/utils/Strings.sol#55-65) is never used and should be removed
Strings.toString(uint256) (../../openzeppelin-contracts/contracts/utils/Str
ings.sol#14-34) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#de
ad-code
```

```
Pragma version^0.8.6 (../interfaces/IGNARDescriptor.sol#2) necessitates a v
ersion too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.6 (../interfaces/IGNARSeeder.sol#2) necessitates a versi
on too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.6 (../interfaces/ISkateContractV2.sol#18) necessitates a
 version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (../../openzeppelin-contracts/contracts/access/Ownable
.sol#3) necessitates a version too recent to be trusted. Consider deploying
 with 0.6.12/0.7.6
Pragma version^0.8.0 (../../openzeppelin-contracts/contracts/token/ERC721/E
RC721.sol#3) necessitates a version too recent to be trusted. Consider depl
oying with 0.6.12/0.7.6
Pragma version^0.8.0 (../../openzeppelin-contracts/contracts/token/ERC721/I
ERC721.sol#3) necessitates a version too recent to be trusted. Consider dep
loying with 0.6.12/0.7.6
Pragma version^0.8.0 (../../openzeppelin-contracts/contracts/token/ERC721/I
ERC721Receiver.sol#3) necessitates a version too recent to be trusted. Cons
ider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (../../openzeppelin-contracts/contracts/token/ERC721/e
xtensions/ERC721Enumerable.sol#3) necessitates a version too recent to be t
rusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (../../openzeppelin-contracts/contracts/token/ERC721/e
xtensions/IERC721Enumerable.sol#3) necessitates a version too recent to be
trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (../../openzeppelin-contracts/contracts/token/ERC721/e
xtensions/IERC721Metadata.sol#3) necessitates a version too recent to be tr
usted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (../../openzeppelin-contracts/contracts/utils/Address.
sol#3) necessitates a version too recent to be trusted. Consider deploying
with 0.6.12/0.7.6
Pragma version^0.8.0 (../../openzeppelin-contracts/contracts/utils/Context.
sol#3) necessitates a version too recent to be trusted. Consider deploying
with 0.6.12/0.7.6
Pragma version^0.8.0 (../../openzeppelin-contracts/contracts/utils/Strings.
sol#3) necessitates a version too recent to be trusted. Consider deploying
with 0.6.12/0.7.6
Pragma version^0.8.0 (../../openzeppelin-contracts/contracts/utils/introspe
ction/ERC165.sol#3) necessitates a version too recent to be trusted. Consid
er deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (../../openzeppelin-contracts/contracts/utils/introspe
ction/IERC165.sol#3) necessitates a version too recent to be trusted. Consi
der deploying with 0.6.12/0.7.6
Pragma version^0.8.6 (SkateContractV2.sol#12) necessitates a version too re
cent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.6 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#in
correct-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (../../openzeppelin-co
ntracts/contracts/utils/Address.sol#54-59):
	- (success) = recipient.call{value: amount}() (../../openzeppelin-c
ontracts/contracts/utils/Address.sol#57)
Low level call in Address.functionCallWithValue(address,bytes,uint256,strin
g) (../../openzeppelin-contracts/contracts/utils/Address.sol#122-133):
	- (success,returndata) = target.call{value: value}(data) (../../ope
nzeppelin-contracts/contracts/utils/Address.sol#131)
Low level call in Address.functionStaticCall(address,bytes,string) (../../o
penzeppelin-contracts/contracts/utils/Address.sol#151-160):
	- (success,returndata) = target.staticcall(data) (../../openzeppeli
n-contracts/contracts/utils/Address.sol#158)
Low level call in Address.functionDelegateCall(address,bytes,string) (../..
/openzeppelin-contracts/contracts/utils/Address.sol#178-187):
	- (success,returndata) = target.delegatecall(data) (../../openzeppe
lin-contracts/contracts/utils/Address.sol#185)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#lo
w-level-calls

Parameter ERC721.safeTransferFrom(address,address,uint256,bytes)._data (../
../openzeppelin-contracts/contracts/token/ERC721/ERC721.sol#181) is not in
mixedCase
Parameter SkateContractV2.setMinter(address)._minter (SkateContractV2.sol#1
27) is not in mixedCase
Parameter SkateContractV2.setDescriptor(IGnarDescriptor)._descriptor (Skate
ContractV2.sol#152) is not in mixedCase
Parameter SkateContractV2.setSeeder(IGnarSeeder)._seeder (SkateContractV2.s
ol#182) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#co
nformance-to-solidity-naming-conventions

renounceOwnership() should be declared external:
	- Ownable.renounceOwnership() (../../openzeppelin-contracts/contrac
ts/access/Ownable.sol#53-55)
transferOwnership(address) should be declared external:
	- Ownable.transferOwnership(address) (../../openzeppelin-contracts/
contracts/access/Ownable.sol#61-64)
name() should be declared external:
	- ERC721.name() (../../openzeppelin-contracts/contracts/token/ERC72
1/ERC721.sol#78-80)
symbol() should be declared external:
	- ERC721.symbol() (../../openzeppelin-contracts/contracts/token/ERC
721/ERC721.sol#85-87)
tokenURI(uint256) should be declared external:
	- ERC721.tokenURI(uint256) (../../openzeppelin-contracts/contracts/
token/ERC721/ERC721.sol#92-97)
	- SkateContractV2.tokenURI(uint256) (SkateContractV2.sol#113-121)
approve(address,uint256) should be declared external:
	- ERC721.approve(address,uint256) (../../openzeppelin-contracts/con
tracts/token/ERC721/ERC721.sol#111-121)
setApprovalForAll(address,bool) should be declared external:
	- ERC721.setApprovalForAll(address,bool) (../../openzeppelin-contra
cts/contracts/token/ERC721/ERC721.sol#135-140)
transferFrom(address,address,uint256) should be declared external:
        - ERC721.transferFrom(address,address,uint256) (../../openzeppelin-
contracts/contracts/token/ERC721/ERC721.sol#152-161)
safeTransferFrom(address,address,uint256) should be declared external:
        - ERC721.safeTransferFrom(address,address,uint256) (../../openzeppe
lin-contracts/contracts/token/ERC721/ERC721.sol#166-172)
tokenOfOwnerByIndex(address,uint256) should be declared external:
        - ERC721Enumerable.tokenOfOwnerByIndex(address,uint256) (../../open
zeppelin-contracts/contracts/token/ERC721/extensions/ERC721Enumerable.sol#3
6-39)
tokenByIndex(uint256) should be declared external:
        - ERC721Enumerable.tokenByIndex(uint256) (../../openzeppelin-contra
cts/contracts/token/ERC721/extensions/ERC721Enumerable.sol#51-54)
mint() should be declared external:
        - SkateContractV2.mint() (SkateContractV2.sol#95-97)
burn(uint256) should be declared external:
        - SkateContractV2.burn(uint256) (SkateContractV2.sol#102-107)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pu
blic-function-that-could-be-declared-external
SkateContractV2.sol analyzed (16 contracts with 75 detectors), 72 result(s)
 found

ERC1967UpgradeUpgradeable._functionDelegateCall(address,bytes) (../../openz
eppelin-contracts-upgradeable/contracts/proxy/ERC1967/ERC1967UpgradeUpgrade
able.sol#198-204) uses delegatecall to a input-controlled function id
        - (success,returndata) = target.delegatecall(data) (../../openzeppe
lin-contracts-upgradeable/contracts/proxy/ERC1967/ERC1967UpgradeUpgradeable
.sol#202)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#co
ntrolled-delegatecall

Reentrancy in SkateContractV2AuctionHouse.createBid(uint256,uint8,uint8) (S
kateContractV2AuctionHouse.sol#111-147):
        External calls:
        - _safeTransferETHWithFallback(lastBidder,_auction.amount) (SkateCo
ntractV2AuctionHouse.sol#133)
              - (success) = to.call{gas: 30_000,value: value}(new bytes(0
)) (SkateContractV2AuctionHouse.sol#323)
              - IWETH(weth).deposit{value: amount}() (SkateContractV2Auct
ionHouse.sol#310)
              - IERC20(weth).transfer(to,amount) (SkateContractV2AuctionH
ouse.sol#311)
       External calls sending eth:
       - _safeTransferETHWithFallback(lastBidder,_auction.amount) (SkateCo
ntractV2AuctionHouse.sol#133)
              - (success) = to.call{gas: 30_000,value: value}(new bytes(0
)) (SkateContractV2AuctionHouse.sol#323)
              - IWETH(weth).deposit{value: amount}() (SkateContractV2Auct
ionHouse.sol#310)
       State variables written after the call(s):
       - auction.amount = msg.value (SkateContractV2AuctionHouse.sol#136)
       - auction.bidder = address(msg.sender) (SkateContractV2AuctionHouse
.sol#137)
       - auction.skatePercent = skatePercent (SkateContractV2AuctionHouse.
sol#138)
       - auction.daoPercent = daoPercent (SkateContractV2AuctionHouse.sol#
139)
Reentrancy in SkateContractV2AuctionHouse.settleCurrentAndCreateNewAuction(
) (SkateContractV2AuctionHouse.sol#89-97):
       External calls:
       - _settleAuction() (SkateContractV2AuctionHouse.sol#95)
              - (success) = to.call{gas: 30_000,value: value}(new bytes(0
)) (SkateContractV2AuctionHouse.sol#323)
              - IWETH(weth).deposit{value: amount}() (SkateContractV2Auct
ionHouse.sol#310)
              - IERC20(weth).transfer(to,amount) (SkateContractV2AuctionH
ouse.sol#311)
              - gnars.burn(_auction.gnarId) (SkateContractV2AuctionHouse.
sol#281)
```

```
              - gnars.transferFrom(address(this),_auction.bidder,_auction
.gnarId) (SkateContractV2AuctionHouse.sol#283)
       - _createAuction() (SkateContractV2AuctionHouse.sol#96)
              - gnars.mint() (SkateContractV2AuctionHouse.sol#246-264)
       External calls sending eth:
       - _settleAuction() (SkateContractV2AuctionHouse.sol#95)
              - (success) = to.call{gas: 30_000,value: value}(new bytes(0
)) (SkateContractV2AuctionHouse.sol#323)
              - IWETH(weth).deposit{value: amount}() (SkateContractV2Auct
ionHouse.sol#310)
       State variables written after the call(s):
       - _createAuction() (SkateContractV2AuctionHouse.sol#96)
              - _paused = true (../../openzeppelin-contracts-upgradeable/
contracts/security/PausableUpgradeable.sol#81)
       - _createAuction() (SkateContractV2AuctionHouse.sol#96)
              - auction = Auction(gnarId,0,startBlock,endBlock,address(0)
,50,50,false) (SkateContractV2AuctionHouse.sol#250-259)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#re
entrancy-vulnerabilities

OwnableUpgradeable.__gap (../../openzeppelin-contracts-upgradeable/contract
s/access/OwnableUpgradeable.sol#87) shadows:
       - ContextUpgradeable.__gap (../../openzeppelin-contracts-upgradeabl
e/contracts/utils/ContextUpgradeable.sol#36)
UUPSUpgradeable.__gap (../../openzeppelin-contracts-upgradeable/contracts/p
roxy/utils/UUPSUpgradeable.sol#107) shadows:
       - ERC1967UpgradeUpgradeable.__gap (../../openzeppelin-contracts-upg
radeable/contracts/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#211)
PausableUpgradeable.__gap (../../openzeppelin-contracts-upgradeable/contrac
ts/security/PausableUpgradeable.sol#102) shadows:
       - ContextUpgradeable.__gap (../../openzeppelin-contracts-upgradeabl
e/contracts/utils/ContextUpgradeable.sol#36)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#st
ate-variable-shadowing
```

```
SkateContractV2AuctionHouse._safeTransferETHWithFallback(address,uint256) (
SkateContractV2AuctionHouse.sol#308-313) ignores return value by IERC20(wet
h).transfer(to,amount) (SkateContractV2AuctionHouse.sol#311)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#un
checked-transfer

SkateContractV2AuctionHouse (SkateContractV2AuctionHouse.sol#24-326) is an
upgradeable contract that does not protect its initiliaze functions: SkateC
ontractV2AuctionHouse.initialize(address,address,ISkateContractV2,address,u
int256,uint8) (SkateContractV2AuctionHouse.sol#62-84). Anyone can delete th
e contract with: UUPSUpgradeable.upgradeTo(address) (../../openzeppelin-con
tracts-upgradeable/contracts/proxy/utils/UUPSUpgradeable.sol#72-75)UUPSUpgr
adeable.upgradeToAndCall(address,bytes) (../../openzeppelin-contracts-upgra
deable/contracts/proxy/utils/UUPSUpgradeable.sol#85-88)Reference: https://g
ithub.com/crytic/slither/wiki/Detector-Documentation#unprotected-upgradeabl
e-contract

SkateContractV2AuctionHouse._settleAuction() (SkateContractV2AuctionHouse.s
ol#271-303) uses a dangerous strict equality:
       - _auction.bidder == address(0) (SkateContractV2AuctionHouse.sol#28
0)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#da
ngerous-strict-equalities

SkateContractV2AuctionHouse._createAuction().gnarId (SkateContractV2Auction
House.sol#246) is a local variable never initialized
ERC1967UpgradeUpgradeable._upgradeToAndCallUUPS(address,bytes,bool).slot (.
./../openzeppelin-contracts-upgradeable/contracts/proxy/ERC1967/ERC1967Upgr
adeUpgradeable.sol#98) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#un
initialized-local-variables

ERC1967UpgradeUpgradeable._upgradeToAndCallUUPS(address,bytes,bool) (../../
openzeppelin-contracts-upgradeable/contracts/proxy/ERC1967/ERC1967UpgradeUp
gradeable.sol#87-105) ignores return value by IERC1822ProxiableUpgradeable(
newImplementation).proxiableUUID() (../../openzeppelin-contracts-upgradeabl
e/contracts/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#98-102)
SkateContractV2AuctionHouse._createAuction() (SkateContractV2AuctionHouse.s
ol#245-265) ignores return value by gnars.mint() (SkateContractV2AuctionHou
se.sol#246-264)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#un
used-return

IGnarDescriptor.addManyBackgrounds(string[]).backgrounds (../interfaces/IGN
ARDescriptor.sol#49) shadows:
       - IGnarDescriptor.backgrounds(uint256) (../interfaces/IGNARDescript
or.sol#24) (function)
IGnarDescriptor.addManyBodies(bytes[]).bodies (../interfaces/IGNARDescripto
r.sol#51) shadows:
       - IGnarDescriptor.bodies(uint256) (../interfaces/IGNARDescriptor.so
l#26) (function)
IGnarDescriptor.addManyAccessories(bytes[]).accessories (../interfaces/IGNA
RDescriptor.sol#53) shadows:
       - IGnarDescriptor.accessories(uint256) (../interfaces/IGNARDescript
or.sol#28) (function)
IGnarDescriptor.addManyHeads(bytes[]).heads (../interfaces/IGNARDescriptor.
sol#55) shadows:
       - IGnarDescriptor.heads(uint256) (../interfaces/IGNARDescriptor.sol
#30) (function)
IGnarDescriptor.addManyGlasses(bytes[]).glasses (../interfaces/IGNARDescrip
tor.sol#57) shadows:
       - IGnarDescriptor.glasses(uint256) (../interfaces/IGNARDescriptor.s
ol#32) (function)
IGnarDescriptor.addGlasses(bytes).glasses (../interfaces/IGNARDescriptor.so
l#70) shadows:
       - IGnarDescriptor.glasses(uint256) (../interfaces/IGNARDescriptor.s
ol#32) (function)
```

```
IGnarDescriptor.setBaseURI(string).baseURI (../interfaces/IGNARDescriptor.s
ol#76) shadows:
       - IGnarDescriptor.baseURI() (../interfaces/IGNARDescriptor.sol#17)
(function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#lo
cal-variable-shadowing

SkateContractV2AuctionHouse.initialize(address,address,ISkateContractV2,add
ress,uint256,uint8)._skate (SkateContractV2AuctionHouse.sol#63) lacks a zer
o-check on :
              - skate = _skate (SkateContractV2AuctionHouse.sol#77)
SkateContractV2AuctionHouse.initialize(address,address,ISkateContractV2,add
ress,uint256,uint8)._dao (SkateContractV2AuctionHouse.sol#64) lacks a zero-
check on :
              - dao = _dao (SkateContractV2AuctionHouse.sol#78)
SkateContractV2AuctionHouse.initialize(address,address,ISkateContractV2,add
ress,uint256,uint8)._weth (SkateContractV2AuctionHouse.sol#66) lacks a zero
-check on :
              - weth = _weth (SkateContractV2AuctionHouse.sol#80)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#mi
ssing-zero-address-validation

Variable 'ERC1967UpgradeUpgradeable._upgradeToAndCallUUPS(address,bytes,boo
l).slot (../../openzeppelin-contracts-upgradeable/contracts/proxy/ERC1967/E
RC1967UpgradeUpgradeable.sol#98)' in ERC1967UpgradeUpgradeable._upgradeToAn
dCallUUPS(address,bytes,bool) (../../openzeppelin-contracts-upgradeable/con
tracts/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#87-105) potentially used
 before declaration: require(bool,string)(slot == _IMPLEMENTATION_SLOT,ERC1
967Upgrade: unsupported proxiableUUID) (../../openzeppelin-contracts-upgrad
eable/contracts/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#99)
Variable 'SkateContractV2AuctionHouse._createAuction().gnarId (SkateContrac
tV2AuctionHouse.sol#246)' in SkateContractV2AuctionHouse._createAuction() (
SkateContractV2AuctionHouse.sol#245-265) potentially used before declaratio
n: auction = Auction(gnarId,0,startBlock,endBlock,address(0),50,50,false) (
SkateContractV2AuctionHouse.sol#250-259)
Variable 'SkateContractV2AuctionHouse._createAuction().gnarId (SkateContrac
tV2AuctionHouse.sol#246)' in SkateContractV2AuctionHouse._createAuction() (
SkateContractV2AuctionHouse.sol#245-265) potentially used before declaratio
n: AuctionCreated(gnarId,startBlock,endBlock,block.timestamp) (SkateContrac
tV2AuctionHouse.sol#261)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pr
e-declaration-usage-of-local-variables

Reentrancy in SkateContractV2AuctionHouse._createAuction() (SkateContractV2
AuctionHouse.sol#245-265):
        External calls:
        - gnars.mint() (SkateContractV2AuctionHouse.sol#246-264)
        State variables written after the call(s):
        - _pause() (SkateContractV2AuctionHouse.sol#263)
                - _paused = true (../../openzeppelin-contracts-upgradeable/
contracts/security/PausableUpgradeable.sol#81)
        - auction = Auction(gnarId,0,startBlock,endBlock,address(0),50,50,f
alse) (SkateContractV2AuctionHouse.sol#250-259)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#re
entrancy-vulnerabilities-2

Reentrancy in SkateContractV2AuctionHouse._createAuction() (SkateContractV2
AuctionHouse.sol#245-265):
        External calls:
        - gnars.mint() (SkateContractV2AuctionHouse.sol#246-264)
        Event emitted after the call(s):
        - AuctionCreated(gnarId,startBlock,endBlock,block.timestamp) (Skate
ContractV2AuctionHouse.sol#261)
        - Paused(_msgSender()) (../../openzeppelin-contracts-upgradeable/co
ntracts/security/PausableUpgradeable.sol#82)
                - _pause() (SkateContractV2AuctionHouse.sol#263)
Reentrancy in SkateContractV2AuctionHouse._settleAuction() (SkateContractV2
AuctionHouse.sol#271-303):
        External calls:
        - gnars.burn(_auction.gnarId) (SkateContractV2AuctionHouse.sol#281)
        - gnars.transferFrom(address(this),_auction.bidder,_auction.gnarId)
 (SkateContractV2AuctionHouse.sol#283)
        - _safeTransferETHWithFallback(skate,(_auction.amount * _auction.sk
atePercent) / 100) (SkateContractV2AuctionHouse.sol#287-290)
                - (success) = to.call{gas: 30_000,value: value}(new bytes(0
)) (SkateContractV2AuctionHouse.sol#323)
                - IWETH(weth).deposit{value: amount}() (SkateContractV2Auct
ionHouse.sol#310)
                - IERC20(weth).transfer(to,amount) (SkateContractV2AuctionH
ouse.sol#311)
        - _safeTransferETHWithFallback(dao,(_auction.amount * _auction.daoP
ercent) / 100) (SkateContractV2AuctionHouse.sol#291-294)
                - (success) = to.call{gas: 30_000,value: value}(new bytes(0
)) (SkateContractV2AuctionHouse.sol#323)
                - IWETH(weth).deposit{value: amount}() (SkateContractV2Auct
ionHouse.sol#310)
                - IERC20(weth).transfer(to,amount) (SkateContractV2AuctionH
ouse.sol#311)
        External calls sending eth:
        - _safeTransferETHWithFallback(skate,(_auction.amount * _auction.sk
atePercent) / 100) (SkateContractV2AuctionHouse.sol#287-290)
                - (success) = to.call{gas: 30_000,value: value}(new bytes(0
)) (SkateContractV2AuctionHouse.sol#323)
                - IWETH(weth).deposit{value: amount}() (SkateContractV2Auct
ionHouse.sol#310)
        - _safeTransferETHWithFallback(dao,(_auction.amount * _auction.daoP
ercent) / 100) (SkateContractV2AuctionHouse.sol#291-294)
                - (success) = to.call{gas: 30_000,value: value}(new bytes(0
)) (SkateContractV2AuctionHouse.sol#323)
                - IWETH(weth).deposit{value: amount}() (SkateContractV2Auct
ionHouse.sol#310)
        Event emitted after the call(s):
```

```
        - AuctionSettled(_auction.gnarId,_auction.bidder,_auction.amount,bl
ock.timestamp) (SkateContractV2AuctionHouse.sol#297-302)
Reentrancy in SkateContractV2AuctionHouse.createBid(uint256,uint8,uint8) (S
kateContractV2AuctionHouse.sol#111-147):
        External calls:
        - _safeTransferETHWithFallback(lastBidder,_auction.amount) (SkateCo
ntractV2AuctionHouse.sol#133)
                - (success) = to.call{gas: 30_000,value: value}(new bytes(0
)) (SkateContractV2AuctionHouse.sol#323)
                - IWETH(weth).deposit{value: amount}() (SkateContractV2Auct
ionHouse.sol#310)
                - IERC20(weth).transfer(to,amount) (SkateContractV2AuctionH
ouse.sol#311)
        External calls sending eth:
        - _safeTransferETHWithFallback(lastBidder,_auction.amount) (SkateCo
ntractV2AuctionHouse.sol#133)
                - (success) = to.call{gas: 30_000,value: value}(new bytes(0
)) (SkateContractV2AuctionHouse.sol#323)
                - IWETH(weth).deposit{value: amount}() (SkateContractV2Auct
ionHouse.sol#310)
        Event emitted after the call(s):
        - AuctionBid(_auction.gnarId,msg.sender,msg.value,block.timestamp)
(SkateContractV2AuctionHouse.sol#141-146)
Reentrancy in SkateContractV2AuctionHouse.settleCurrentAndCreateNewAuction(
) (SkateContractV2AuctionHouse.sol#89-97):
        External calls:
        - _settleAuction() (SkateContractV2AuctionHouse.sol#95)
                - (success) = to.call{gas: 30_000,value: value}(new bytes(0
)) (SkateContractV2AuctionHouse.sol#323)
                - IWETH(weth).deposit{value: amount}() (SkateContractV2Auct
ionHouse.sol#310)
                - IERC20(weth).transfer(to,amount) (SkateContractV2AuctionH
ouse.sol#311)
                - gnars.burn(_auction.gnarId) (SkateContractV2AuctionHouse.
sol#281)
              - gnars.transferFrom(address(this),_auction.bidder,_auction
.gnarId) (SkateContractV2AuctionHouse.sol#283)
       - _createAuction() (SkateContractV2AuctionHouse.sol#96)
              - gnars.mint() (SkateContractV2AuctionHouse.sol#246-264)
       External calls sending eth:
       - _settleAuction() (SkateContractV2AuctionHouse.sol#95)
              - (success) = to.call{gas: 30_000,value: value}(new bytes(0
)) (SkateContractV2AuctionHouse.sol#323)
              - IWETH(weth).deposit{value: amount}() (SkateContractV2Auct
ionHouse.sol#310)
       Event emitted after the call(s):
       - AuctionCreated(gnarId,startBlock,endBlock,block.timestamp) (Skate
ContractV2AuctionHouse.sol#261)
              - _createAuction() (SkateContractV2AuctionHouse.sol#96)
       - Paused(_msgSender()) (../../openzeppelin-contracts-upgradeable/co
ntracts/security/PausableUpgradeable.sol#82)
              - _createAuction() (SkateContractV2AuctionHouse.sol#96)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#re
entrancy-vulnerabilities-3

AddressUpgradeable.verifyCallResult(bool,bytes,string) (../../openzeppelin-
contracts-upgradeable/contracts/utils/AddressUpgradeable.sol#174-194) uses
assembly
       - INLINE ASM (../../openzeppelin-contracts-upgradeable/contracts/ut
ils/AddressUpgradeable.sol#186-189)
StorageSlotUpgradeable.getAddressSlot(bytes32) (../../openzeppelin-contract
s-upgradeable/contracts/utils/StorageSlotUpgradeable.sol#52-56) uses assemb
ly
       - INLINE ASM (../../openzeppelin-contracts-upgradeable/contracts/ut
ils/StorageSlotUpgradeable.sol#53-55)
StorageSlotUpgradeable.getBooleanSlot(bytes32) (../../openzeppelin-contract
s-upgradeable/contracts/utils/StorageSlotUpgradeable.sol#61-65) uses assemb
ly
```

```
        - INLINE ASM (../../openzeppelin-contracts-upgradeable/contracts/ut
ils/StorageSlotUpgradeable.sol#62-64)
StorageSlotUpgradeable.getBytes32Slot(bytes32) (../../openzeppelin-contract
s-upgradeable/contracts/utils/StorageSlotUpgradeable.sol#70-74) uses assemb
ly
        - INLINE ASM (../../openzeppelin-contracts-upgradeable/contracts/ut
ils/StorageSlotUpgradeable.sol#71-73)
StorageSlotUpgradeable.getUint256Slot(bytes32) (../../openzeppelin-contract
s-upgradeable/contracts/utils/StorageSlotUpgradeable.sol#79-83) uses assemb
ly
        - INLINE ASM (../../openzeppelin-contracts-upgradeable/contracts/ut
ils/StorageSlotUpgradeable.sol#80-82)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#as
sembly-usage

Different versions of Solidity is used:
        - Version used: ['^0.8.0', '^0.8.1', '^0.8.2', '^0.8.6']
        - ^0.8.6 (../interfaces/IGNARDescriptor.sol#2)
        - ^0.8.6 (../interfaces/IGNARSeeder.sol#2)
        - ^0.8.6 (../interfaces/ISkateContractV2.sol#18)
        - ^0.8.6 (../interfaces/ISkateContractV2AuctionHouse.sol#18)
        - ^0.8.6 (../interfaces/IWETH.sol#3)
        - ^0.8.0 (../../openzeppelin-contracts-upgradeable/contracts/access
/OwnableUpgradeable.sol#4)
        - ^0.8.0 (../../openzeppelin-contracts-upgradeable/contracts/interf
aces/draft-IERC1822Upgradeable.sol#4)
        - ^0.8.2 (../../openzeppelin-contracts-upgradeable/contracts/proxy/
ERC1967/ERC1967UpgradeUpgradeable.sol#4)
        - ^0.8.0 (../../openzeppelin-contracts-upgradeable/contracts/proxy/
beacon/IBeaconUpgradeable.sol#4)
        - ^0.8.0 (../../openzeppelin-contracts-upgradeable/contracts/proxy/
utils/Initializable.sol#4)
        - ^0.8.0 (../../openzeppelin-contracts-upgradeable/contracts/proxy/
utils/UUPSUpgradeable.sol#4)
```

```
	- ^0.8.0 (../../openzeppelin-contracts-upgradeable/contracts/securi
ty/PausableUpgradeable.sol#4)
	- ^0.8.0 (../../openzeppelin-contracts-upgradeable/contracts/securi
ty/ReentrancyGuardUpgradeable.sol#4)
	- ^0.8.1 (../../openzeppelin-contracts-upgradeable/contracts/utils/
AddressUpgradeable.sol#4)
	- ^0.8.0 (../../openzeppelin-contracts-upgradeable/contracts/utils/
ContextUpgradeable.sol#4)
	- ^0.8.0 (../../openzeppelin-contracts-upgradeable/contracts/utils/
StorageSlotUpgradeable.sol#4)
	- ^0.8.0 (../../openzeppelin-contracts/contracts/token/ERC20/IERC20
.sol#3)
	- ^0.8.0 (../../openzeppelin-contracts/contracts/token/ERC721/IERC7
21.sol#3)
	- ^0.8.0 (../../openzeppelin-contracts/contracts/utils/introspectio
n/IERC165.sol#3)
	- ^0.8.6 (SkateContractV2AuctionHouse.sol#13)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#di
fferent-pragma-directives-are-used

AddressUpgradeable.functionCall(address,bytes) (../../openzeppelin-contract
s-upgradeable/contracts/utils/AddressUpgradeable.sol#85-87) is never used a
nd should be removed
AddressUpgradeable.functionCall(address,bytes,string) (../../openzeppelin-c
ontracts-upgradeable/contracts/utils/AddressUpgradeable.sol#95-101) is neve
r used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (../../open
zeppelin-contracts-upgradeable/contracts/utils/AddressUpgradeable.sol#114-1
20) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (../
../openzeppelin-contracts-upgradeable/contracts/utils/AddressUpgradeable.so
l#128-139) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes) (../../openzeppelin-co
ntracts-upgradeable/contracts/utils/AddressUpgradeable.sol#147-149) is neve
```

```
r used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes,string) (../../openzepp
elin-contracts-upgradeable/contracts/utils/AddressUpgradeable.sol#157-166)
is never used and should be removed
AddressUpgradeable.sendValue(address,uint256) (../../openzeppelin-contracts
-upgradeable/contracts/utils/AddressUpgradeable.sol#60-65) is never used an
d should be removed
ContextUpgradeable.__Context_init() (../../openzeppelin-contracts-upgradeab
le/contracts/utils/ContextUpgradeable.sol#18-19) is never used and should b
e removed
ContextUpgradeable.__Context_init_unchained() (../../openzeppelin-contracts
-upgradeable/contracts/utils/ContextUpgradeable.sol#21-22) is never used an
d should be removed
ContextUpgradeable._msgData() (../../openzeppelin-contracts-upgradeable/con
tracts/utils/ContextUpgradeable.sol#27-29) is never used and should be remo
ved
ERC1967UpgradeUpgradeable.__ERC1967Upgrade_init() (../../openzeppelin-contr
acts-upgradeable/contracts/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#21-2
2) is never used and should be removed
ERC1967UpgradeUpgradeable.__ERC1967Upgrade_init_unchained() (../../openzepp
elin-contracts-upgradeable/contracts/proxy/ERC1967/ERC1967UpgradeUpgradeabl
e.sol#24-25) is never used and should be removed
ERC1967UpgradeUpgradeable._changeAdmin(address) (../../openzeppelin-contrac
ts-upgradeable/contracts/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#139-14
2) is never used and should be removed
ERC1967UpgradeUpgradeable._getAdmin() (../../openzeppelin-contracts-upgrade
able/contracts/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#122-124) is neve
r used and should be removed
ERC1967UpgradeUpgradeable._getBeacon() (../../openzeppelin-contracts-upgrad
eable/contracts/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#158-160) is nev
er used and should be removed
ERC1967UpgradeUpgradeable._setAdmin(address) (../../openzeppelin-contracts-
upgradeable/contracts/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#129-132)
is never used and should be removed
```

```
ERC1967UpgradeUpgradeable._setBeacon(address) (../../openzeppelin-contracts
-upgradeable/contracts/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#165-172)
 is never used and should be removed
ERC1967UpgradeUpgradeable._upgradeBeaconToAndCall(address,bytes,bool) (../.
./openzeppelin-contracts-upgradeable/contracts/proxy/ERC1967/ERC1967Upgrade
Upgradeable.sol#180-190) is never used and should be removed
Initializable._disableInitializers() (../../openzeppelin-contracts-upgradea
ble/contracts/proxy/utils/Initializable.sol#129-131) is never used and shou
ld be removed
StorageSlotUpgradeable.getBytes32Slot(bytes32) (../../openzeppelin-contract
s-upgradeable/contracts/utils/StorageSlotUpgradeable.sol#70-74) is never us
ed and should be removed
StorageSlotUpgradeable.getUint256Slot(bytes32) (../../openzeppelin-contract
s-upgradeable/contracts/utils/StorageSlotUpgradeable.sol#79-83) is never us
ed and should be removed
UUPSUpgradeable.__UUPSUpgradeable_init_unchained() (../../openzeppelin-cont
racts-upgradeable/contracts/proxy/utils/UUPSUpgradeable.sol#26-27) is never
 used and should be removed
UUPSUpgradeable._authorizeUpgrade(address) (../../openzeppelin-contracts-up
gradeable/contracts/proxy/utils/UUPSUpgradeable.sol#100) is never used and
should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#de
ad-code

Pragma version^0.8.6 (../interfaces/IGNARDescriptor.sol#2) necessitates a v
ersion too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.6 (../interfaces/IGNARSeeder.sol#2) necessitates a versi
on too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.6 (../interfaces/ISkateContractV2.sol#18) necessitates a
 version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.6 (../interfaces/ISkateContractV2AuctionHouse.sol#18) ne
cessitates a version too recent to be trusted. Consider deploying with 0.6.
12/0.7.6
Pragma version^0.8.6 (../interfaces/IWETH.sol#3) necessitates a version too
```

```
 recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (../../openzeppelin-contracts-upgradeable/contracts/ac
cess/OwnableUpgradeable.sol#4) necessitates a version too recent to be trus
ted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (../../openzeppelin-contracts-upgradeable/contracts/in
terfaces/draft-IERC1822Upgradeable.sol#4) necessitates a version too recent
 to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.2 (../../openzeppelin-contracts-upgradeable/contracts/pr
oxy/ERC1967/ERC1967UpgradeUpgradeable.sol#4) necessitates a version too rec
ent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (../../openzeppelin-contracts-upgradeable/contracts/pr
oxy/beacon/IBeaconUpgradeable.sol#4) necessitates a version too recent to b
e trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (../../openzeppelin-contracts-upgradeable/contracts/pr
oxy/utils/Initializable.sol#4) necessitates a version too recent to be trus
ted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (../../openzeppelin-contracts-upgradeable/contracts/pr
oxy/utils/UUPSUpgradeable.sol#4) necessitates a version too recent to be tr
usted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (../../openzeppelin-contracts-upgradeable/contracts/se
curity/PausableUpgradeable.sol#4) necessitates a version too recent to be t
rusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (../../openzeppelin-contracts-upgradeable/contracts/se
curity/ReentrancyGuardUpgradeable.sol#4) necessitates a version too recent
to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.1 (../../openzeppelin-contracts-upgradeable/contracts/ut
ils/AddressUpgradeable.sol#4) necessitates a version too recent to be trust
ed. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (../../openzeppelin-contracts-upgradeable/contracts/ut
ils/ContextUpgradeable.sol#4) necessitates a version too recent to be trust
ed. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (../../openzeppelin-contracts-upgradeable/contracts/ut
ils/StorageSlotUpgradeable.sol#4) necessitates a version too recent to be t
rusted. Consider deploying with 0.6.12/0.7.6
```

```
Pragma version^0.8.0 (../../openzeppelin-contracts/contracts/token/ERC20/IE
RC20.sol#3) necessitates a version too recent to be trusted. Consider deplo
ying with 0.6.12/0.7.6
Pragma version^0.8.0 (../../openzeppelin-contracts/contracts/token/ERC721/I
ERC721.sol#3) necessitates a version too recent to be trusted. Consider dep
loying with 0.6.12/0.7.6
Pragma version^0.8.0 (../../openzeppelin-contracts/contracts/utils/introspe
ction/IERC165.sol#3) necessitates a version too recent to be trusted. Consi
der deploying with 0.6.12/0.7.6
Pragma version^0.8.6 (SkateContractV2AuctionHouse.sol#13) necessitates a ve
rsion too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.6 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#in
correct-versions-of-solidity

Low level call in ERC1967UpgradeUpgradeable._functionDelegateCall(address,b
ytes) (../../openzeppelin-contracts-upgradeable/contracts/proxy/ERC1967/ERC
1967UpgradeUpgradeable.sol#198-204):
       - (success,returndata) = target.delegatecall(data) (../../openzeppe
lin-contracts-upgradeable/contracts/proxy/ERC1967/ERC1967UpgradeUpgradeable
.sol#202)
Low level call in AddressUpgradeable.sendValue(address,uint256) (../../open
zeppelin-contracts-upgradeable/contracts/utils/AddressUpgradeable.sol#60-65
):
       - (success) = recipient.call{value: amount}() (../../openzeppelin-c
ontracts-upgradeable/contracts/utils/AddressUpgradeable.sol#63)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,ui
nt256,string) (../../openzeppelin-contracts-upgradeable/contracts/utils/Add
ressUpgradeable.sol#128-139):
       - (success,returndata) = target.call{value: value}(data) (../../ope
nzeppelin-contracts-upgradeable/contracts/utils/AddressUpgradeable.sol#137)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,strin
g) (../../openzeppelin-contracts-upgradeable/contracts/utils/AddressUpgrade
able.sol#157-166):
        - (success,returndata) = target.staticcall(data) (../../openzeppeli
n-contracts-upgradeable/contracts/utils/AddressUpgradeable.sol#164)
Low level call in SkateContractV2AuctionHouse._safeTransferETH(address,uint
256) (SkateContractV2AuctionHouse.sol#319-325):
        - (success) = to.call{gas: 30_000,value: value}(new bytes(0)) (Skat
eContractV2AuctionHouse.sol#323)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#lo
w-level-calls

Function OwnableUpgradeable.__Ownable_init() (../../openzeppelin-contracts-
upgradeable/contracts/access/OwnableUpgradeable.sol#29-31) is not in mixedC
ase
Function OwnableUpgradeable.__Ownable_init_unchained() (../../openzeppelin-
contracts-upgradeable/contracts/access/OwnableUpgradeable.sol#33-35) is not
 in mixedCase
Variable OwnableUpgradeable.__gap (../../openzeppelin-contracts-upgradeable
/contracts/access/OwnableUpgradeable.sol#87) is not in mixedCase
Function ERC1967UpgradeUpgradeable.__ERC1967Upgrade_init() (../../openzeppe
lin-contracts-upgradeable/contracts/proxy/ERC1967/ERC1967UpgradeUpgradeable
.sol#21-22) is not in mixedCase
Function ERC1967UpgradeUpgradeable.__ERC1967Upgrade_init_unchained() (../..
/openzeppelin-contracts-upgradeable/contracts/proxy/ERC1967/ERC1967UpgradeU
pgradeable.sol#24-25) is not in mixedCase
Variable ERC1967UpgradeUpgradeable.__gap (../../openzeppelin-contracts-upgr
adeable/contracts/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#211) is not i
n mixedCase
Function UUPSUpgradeable.__UUPSUpgradeable_init() (../../openzeppelin-contr
acts-upgradeable/contracts/proxy/utils/UUPSUpgradeable.sol#23-24) is not in
 mixedCase
Function UUPSUpgradeable.__UUPSUpgradeable_init_unchained() (../../openzepp
elin-contracts-upgradeable/contracts/proxy/utils/UUPSUpgradeable.sol#26-27)
 is not in mixedCase
Variable UUPSUpgradeable.__gap (../../openzeppelin-contracts-upgradeable/co
ntracts/proxy/utils/UUPSUpgradeable.sol#107) is not in mixedCase
```

```
Variable UUPSUpgradeable.__self (../../openzeppelin-contracts-upgradeable/c
ontracts/proxy/utils/UUPSUpgradeable.sol#29) is not in mixedCase
Function PausableUpgradeable.__Pausable_init() (../../openzeppelin-contract
s-upgradeable/contracts/security/PausableUpgradeable.sol#34-36) is not in m
ixedCase
Function PausableUpgradeable.__Pausable_init_unchained() (../../openzeppeli
n-contracts-upgradeable/contracts/security/PausableUpgradeable.sol#38-40) i
s not in mixedCase
Variable PausableUpgradeable.__gap (../../openzeppelin-contracts-upgradeabl
e/contracts/security/PausableUpgradeable.sol#102) is not in mixedCase
Function ReentrancyGuardUpgradeable.__ReentrancyGuard_init() (../../openzep
pelin-contracts-upgradeable/contracts/security/ReentrancyGuardUpgradeable.s
ol#40-42) is not in mixedCase
Function ReentrancyGuardUpgradeable.__ReentrancyGuard_init_unchained() (../
../openzeppelin-contracts-upgradeable/contracts/security/ReentrancyGuardUpg
radeable.sol#44-46) is not in mixedCase
Variable ReentrancyGuardUpgradeable.__gap (../../openzeppelin-contracts-upg
radeable/contracts/security/ReentrancyGuardUpgradeable.sol#74) is not in mi
xedCase
Function ContextUpgradeable.__Context_init() (../../openzeppelin-contracts-
upgradeable/contracts/utils/ContextUpgradeable.sol#18-19) is not in mixedCa
se
Function ContextUpgradeable.__Context_init_unchained() (../../openzeppelin-
contracts-upgradeable/contracts/utils/ContextUpgradeable.sol#21-22) is not
in mixedCase
Variable ContextUpgradeable.__gap (../../openzeppelin-contracts-upgradeable
/contracts/utils/ContextUpgradeable.sol#36) is not in mixedCase
Parameter SkateContractV2AuctionHouse.initialize(address,address,ISkateCont
ractV2,address,uint256,uint8)._skate (SkateContractV2AuctionHouse.sol#63) i
s not in mixedCase
Parameter SkateContractV2AuctionHouse.initialize(address,address,ISkateCont
ractV2,address,uint256,uint8)._dao (SkateContractV2AuctionHouse.sol#64) is
not in mixedCase
Parameter SkateContractV2AuctionHouse.initialize(address,address,ISkateCont
ractV2,address,uint256,uint8)._gnars (SkateContractV2AuctionHouse.sol#65) i
s not in mixedCase
Parameter SkateContractV2AuctionHouse.initialize(address,address,ISkateCont
ractV2,address,uint256,uint8)._weth (SkateContractV2AuctionHouse.sol#66) is
 not in mixedCase
Parameter SkateContractV2AuctionHouse.initialize(address,address,ISkateCont
ractV2,address,uint256,uint8)._reservePrice (SkateContractV2AuctionHouse.so
l#67) is not in mixedCase
Parameter SkateContractV2AuctionHouse.initialize(address,address,ISkateCont
ractV2,address,uint256,uint8)._minBidIncrementPercentage (SkateContractV2Au
ctionHouse.sol#68) is not in mixedCase
Parameter SkateContractV2AuctionHouse.setReservePrice(uint256)._reservePric
e (SkateContractV2AuctionHouse.sol#176) is not in mixedCase
Parameter SkateContractV2AuctionHouse.setMinBidIncrementPercentage(uint8)._
minBidIncrementPercentage (SkateContractV2AuctionHouse.sol#190) is not in m
ixedCase
Parameter SkateContractV2AuctionHouse.setAuctionPeriodBlocks(uint16)._aucti
onPeriodBlocks (SkateContractV2AuctionHouse.sol#202) is not in mixedCase
Parameter SkateContractV2AuctionHouse.setSkateDaoAddresses(address,address)
._skate (SkateContractV2AuctionHouse.sol#216) is not in mixedCase
Parameter SkateContractV2AuctionHouse.setSkateDaoAddresses(address,address)
._dao (SkateContractV2AuctionHouse.sol#216) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#co
nformance-to-solidity-naming-conventions

UUPSUpgradeable.__gap (../../openzeppelin-contracts-upgradeable/contracts/p
roxy/utils/UUPSUpgradeable.sol#107) is never used in SkateContractV2Auction
House (SkateContractV2AuctionHouse.sol#24-326)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#un
used-state-variable

renounceOwnership() should be declared external:
	- OwnableUpgradeable.renounceOwnership() (../../openzeppelin-contra
cts-upgradeable/contracts/access/OwnableUpgradeable.sol#59-61)
```

```
transferOwnership(address) should be declared external:
	- OwnableUpgradeable.transferOwnership(address) (../../openzeppelin
-contracts-upgradeable/contracts/access/OwnableUpgradeable.sol#67-70)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pu
blic-function-that-could-be-declared-external
SkateContractV2AuctionHouse.sol analyzed (20 contracts with 75 detectors),
119 result(s) found
```

Conclusion:

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

6 Conclusion

In this audit, we examined the design and implementation of Gnars V2 contract and discovered several issues of varying severity. Bitlabs.dev team addressed 6 issues raised in the initial report and implemented the necessary fixes, while classifying the rest as a risk with low-probability of occurrence. Shellboxes' auditors advised Bitlabs.dev Team to maintain a high level of vigilance and to keep those findings in mind in order to avoid any future complications.

7 Disclaimer

Shellboxes reports should not be construed as "endorsements" or "disapprovals" of particular teams or projects. These reports do not reflect the economics or value of any "product" or "asset" produced by any team or project that engages Shellboxes to do a security evaluation, nor should they be regarded as such. Shellboxes Reports do not provide any warranty or guarantee regarding the absolute bug-free nature of the examined technology, nor do they provide any indication of the technology's proprietors, business model, business or legal compliance. Shellboxes Reports should not be used in any way to decide whether to invest in or take part in a certain project. These reports don't offer any kind of investing advice and shouldn't be used that way. Shellboxes Reports are the result of a thorough auditing process designed to assist our clients in improving the quality of their code while lowering the significant risk posed by blockchain technology. According to Shellboxes, each business and person is in charge of their own due diligence and ongoing security. Shellboxes does not guarantee the security or functionality of the technology we agree to research; instead, our purpose is to assist in limiting the attack vectors and the high degree of variation associated with using new and evolving technologies.

SHELLBOXES

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