Smart Contract

Security Audit V1

Wyndblast Marketplace & MiniGame Audit

18/5/2022

OnlyUp Capital

Table of contents

Introduction

OnlyUp was contracted by the Wyndblast team to perform the Security audit of the Wyndblast Protocol smart contracts code. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on May 18th, 2022.

The purpose of this audit was to address the following:

- Ensure that all claimed functions exist and function correctly.
- Identify any security vulnerabilities that may be present in the smart contract.

Project Background

WyndBlast is a play and earn multiplayer co-operative game built on the Avalanche blockchain. This audit project consists of game and marketplace smart contracts.

Audit scope

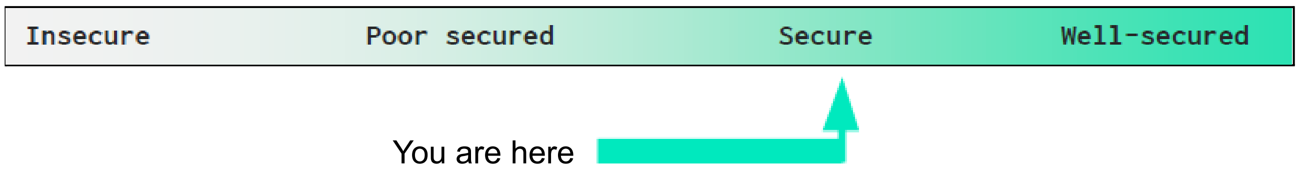
Name	**Code Review and Security Analysis Report for Wyndblast Protocol Smart Contracts**
Platform	**Avalanche / Solidity**
File 1	WBGame.sol
File 1 MD5 Hash	B4797CB4DDD640A73E62788D73B0EBB8
Updated File 1 MD5 Hash	1523F577436762D8C012AEDDC3F542A6
File 2	Marketplace.sol
File 2 MD5 Hash	D314A92483287FA721B0D33B1DFF86EB
Updated File 2 MD5 Hash	3D8FEE90C0D9AF0AE2262B91070FF794
Updated File 2 MD5 Hash	52A495046B9044380F7CB6C8152F2B8C
Audit Date	May 14th, 2022
Revise Audit Date	May 18th, 2022

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1 WBGame.sol ● Breeding Cost: 200 ● Owner can set rewards for an individual wallet, attach tokens to the contract, dispatch tokens from the contract ● Users can breed, buyTickets, move token holdings, claim rewards	**YES, This is valid.**
File 2 Marketplace.sol ● Bid Threshold: 50 ● Auction for NFT tokens, Bid, buy, sell for auction ● Owner can set the bid threshold, the job executor, token address for payment, cancel the auction, add/remove fee collectors, publication fee	**YES, This is valid.**

Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are **"Secured"**. This token contract does contain owner control, which does not make it fully decentralized.



We used various tools like Slither, Solhint and Remix IDE. At the same time this finding is based on critical analysis of the manual audit.
All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the Audit overview section. General overview is presented in AS-IS section and all identified issues can be found in the Audit overview section.

We found 0 critical, 1 high, 3 medium and 1 low and some very low level issues.

All these issues have been fixed / acknowledged in the revised code.

Investors Advice: Technical audit of the smart contract does not guarantee the ethical nature of the project. Any owner controlled functions should be executed by the owner with responsibility. All investors/users are advised to do their due diligence before investing in the project.

Technical Quick Stats

Main Category	Subcategory	Result
Contract Programming	Solidity version not specified	Passed
	Solidity version too old	Passed
	Integer overflow/underflow	Passed
	Function input parameters lack of check	Passed
	Function input parameters check bypass	Passed
	Function access control lacks management	Passed
	Critical operation lacks event log	Passed
	Human/contract checks bypass	Passed
	Random number generation/use vulnerability	N/A
	Fallback function misuse	Passed
	Race condition	Passed
	Logical vulnerability	Passed
	Features claimed	Passed
	Other programming issues	Passed
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Passed
	Unused code	Passed
Gas Optimization	“Out of Gas” Issue	Passed
	High consumption ‘for/while’ loop	Passed
	High consumption ‘storage’ storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage not set	Passed
	“Short Address” Attack	Passed
	“Double Spend” Attack	Passed

Overall Audit Result: PASSED

Code Quality

This audit scope has 2 smart contract files. Smart contracts contain Libraries, Smart contracts, inherits and Interfaces. This is a compact and well written smart contract.

The libraries in the Wyndblast Protocol are part of its logical algorithm. A library is a different type of smart contract that contains reusable code. Once deployed on the blockchain (only once), it is assigned a specific address and its properties / methods can be reused many times by other contracts in the Wyndblast Protocol.

The Wyndblast team has not provided unit test scripts, which would have helped to determine the integrity of the code in an automated way.

Code parts are **well** commented on smart contracts.

Documentation

We were given a Wyndblast Protocol smart contract code in the form of a github weblink. The hash of that code is mentioned above in the table.

As mentioned above, code parts are **well** commented. So it is easy to quickly understand the programming flow as well as complex code logic. Comments are very helpful in understanding the overall architecture of the protocol.

Another source of information was its official website https://wyndblast.com which provided rich information about the project architecture.

Use of Dependencies

As per our observation, the libraries are used in this smart contracts infrastructure that are based on well known industry standard open source projects.

Apart from libraries, its functions are used in external smart contract calls.

AS-IS overview

WBGame.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	onERC721Received	write	Passed	No Issue
3	__Ownable_init	internal	access only Initializing	No Issue
4	__Ownable_init_unchained	internal	access only Initializing	No Issue
5	owner	read	Passed	No Issue
6	_transferOwnership	internal	Passed	No Issue
7	onlyOwner	modifier	Passed	No Issue
8	renounceOwnership	write	access only Owner	No Issue
9	transferOwnership	write	access only Owner	No Issue
10	__ReentrancyGuard_init	internal	access only Initializing	No Issue
11	__ReentrancyGuard_init_unchained	internal	access only Initializing	No Issue
12	nonReentrant	modifier	Passed	No Issue
13	initialize	write	Passed	No Issue
14	buyTicket	write	Passed	No Issue
15	_submit	internal	Passed	No Issue
16	batchSubmit	write	Passed	No Issue
17	_dispatch	internal	Passed	No Issue
18	batchDispatch	write	Passed	No Issue
19	_removeElement	internal	Passed	No Issue
20	_remove	internal	Passed	No Issue
21	_save	internal	Passed	No Issue
22	idsOf	read	Passed	No Issue
23	setReward	write	access only Owner	No Issue
24	batchSetReward	write	access only Owner	No Issue
25	claimReward	write	Passed	No Issue
26	safeDispatch	write	access only Owner	No Issue
27	viewTotalRewards	external	access only Owner	No Issue
28	breed	write	Passed	No Issue
29	breedCountOf	read	Passed	No Issue
30	move	write	Passed	No Issue
31	batchMove	write	Passed	No Issue

Marketplace.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	onERC721Received	write	Passed	No Issue
3	__Ownable_init	internal	access only Initializing	No Issue
4	__Ownable_init_unchained	internal	access only Initializing	No Issue
5	owner	read	Passed	No Issue
6	_transferOwnership	internal	Passed	No Issue
7	onlyOwner	modifier	Passed	No Issue
8	renounceOwnership	write	access only Owner	No Issue
9	transferOwnership	write	access only Owner	No Issue
10	__Pausable_init	internal	access only Initializing	No Issue
11	__Pausable_init_unchained	internal	access only Initializing	No Issue
12	paused	read	Passed	No Issue
13	whenNotPaused	modifier	Passed	No Issue
14	whenPaused	modifier	Passed	No Issue
15	_pause	internal	Passed	No Issue
16	_unpause	internal	Passed	No Issue
17	initialize	write	Passed	No Issue
18	onlyExecutor	modifier	Passed	No Issue
19	sellerOf	read	Passed	No Issue
20	auction	write	Passed	No Issue
21	sell	write	Passed	No Issue
22	cancel	write	Passed	No Issue
23	buy	write	Passed	No Issue
24	bid	write	Passed	No Issue
25	swap	write	Passed	No Issue
26	approveSwap	write	Passed	No Issue
27	rejectSwap	write	Passed	No Issue
28	cancelSwap	write	Passed	No Issue
29	getAuctionExpiry	read	Passed	No Issue
30	getItems	read	Passed	No Issue
31	getItem	read	Passed	No Issue
32	getBids	read	Passed	No Issue
33	getBid	read	Passed	No Issue
34	getSwaps	read	Passed	No Issue
35	getSwap	read	Passed	No Issue
36	getRoyaltyInfo	external	Passed	No Issue
37	checkRoyalties	external	Passed	No Issue
38	setTokenAddress	write	access only Owner	No Issue
39	pause	write	access only Owner	No Issue
40	unpause	write	access only Owner	No Issue
41	getCollections	read	Passed	No Issue
42	createCollection	write	access only Owner	No Issue
43	removeCollection	write	access only Owner	No Issue
44	updateCollection	write	access only Owner	No Issue

45	getFeeCollectors	read	access only Owner	No Issue
46	addFeeCollector	write	access only Owner	No Issue
47	removeFeeCollector	write	access only Owner	No Issue
48	emergencyTransferTo	write	access only Owner	No Issue
49	emergencyCancel	write	access only Owner	No Issue
50	setJobExecutor	write	access only Owner	No Issue
51	setBidThreshold	write	access only Owner	No Issue
52	setPublicationFee	write	access only Owner	No Issue
53	setPublicationFeeWallet	write	access only Owner	No Issue
54	getPublicationFeeWallet	write	access only Owner	No Issue
55	executeJob	write	access only Owner	No Issue
56	_putHoldAmount	internal	Passed	No Issue
57	_releaseHoldAmount	internal	Passed	No Issue
58	_isRoyaltiesSupport	read	Passed	No Issue
59	_getRoyaltyInfo	read	Passed	No Issue
60	_createItem	internal	Passed	No Issue
61	_createItem	internal	Passed	No Issue
62	_isActiveCollection	internal	Passed	No Issue
63	_executePayment	internal	access only Owner	No Issue

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to token loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial
Medium	Medium-level vulnerabilities are important to fix; however, they can’t lead to tokens lose
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can’t have significant impact on execution
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations and info statements can’t affect smart contract execution and can be ignored.

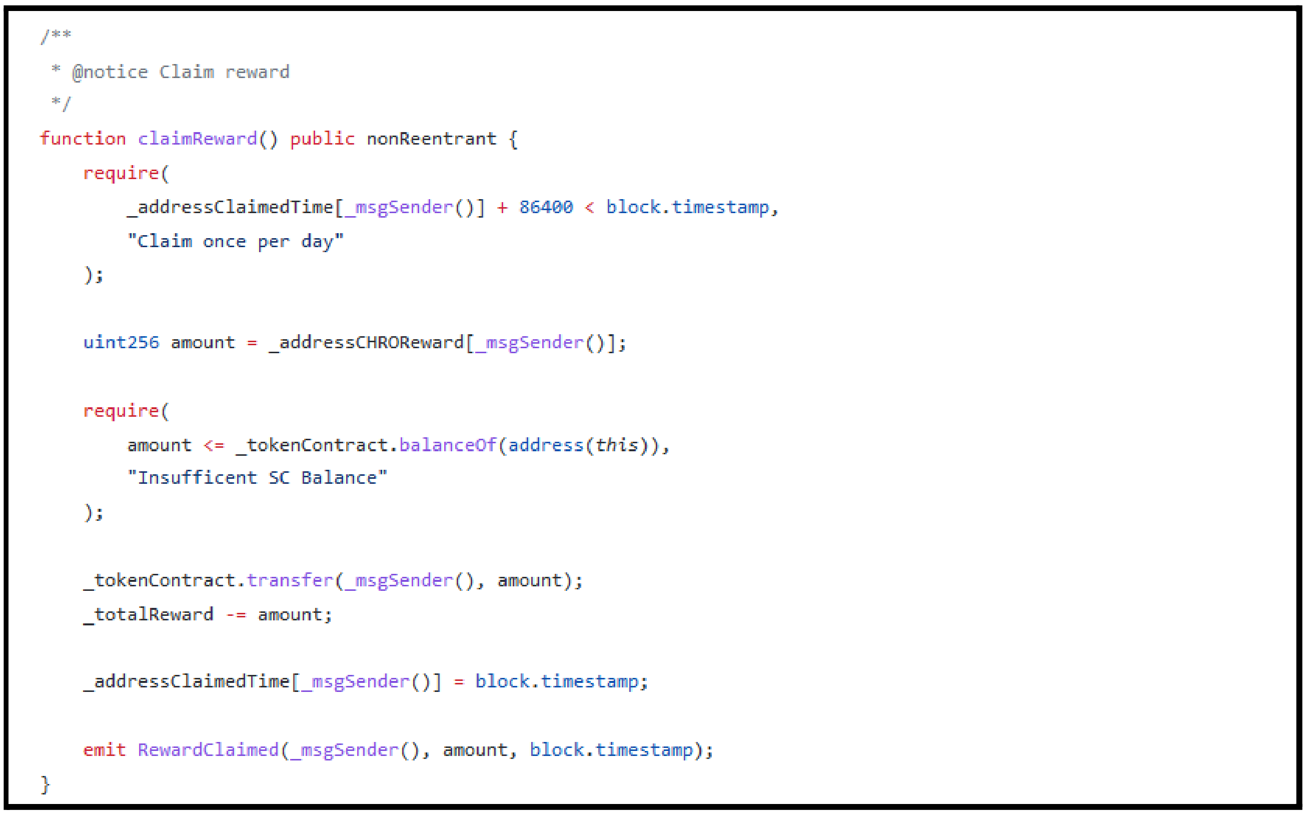
Audit Findings

Critical Severity

No Critical severity vulnerabilities were found.

High Severity

(1) Users can claim rewards everyday: **WBGame.sol**



```
/**
 * @notice Claim reward
 */
function claimReward() public nonReentrant {
    require(
        _addressClaimedTime[_msgSender()] + 86400 < block.timestamp,
        "Claim once per day"
    );

    uint256 amount = _addressCHROReward[_msgSender()];

    require(
        amount <= _tokenContract.balanceOf(address(this)),
        "Insufficent SC Balance"
    );

    _tokenContract.transfer(_msgSender(), amount);
    _totalReward -= amount;

    _addressClaimedTime[_msgSender()] = block.timestamp;

    emit RewardClaimed(_msgSender(), amount, block.timestamp);
}
```

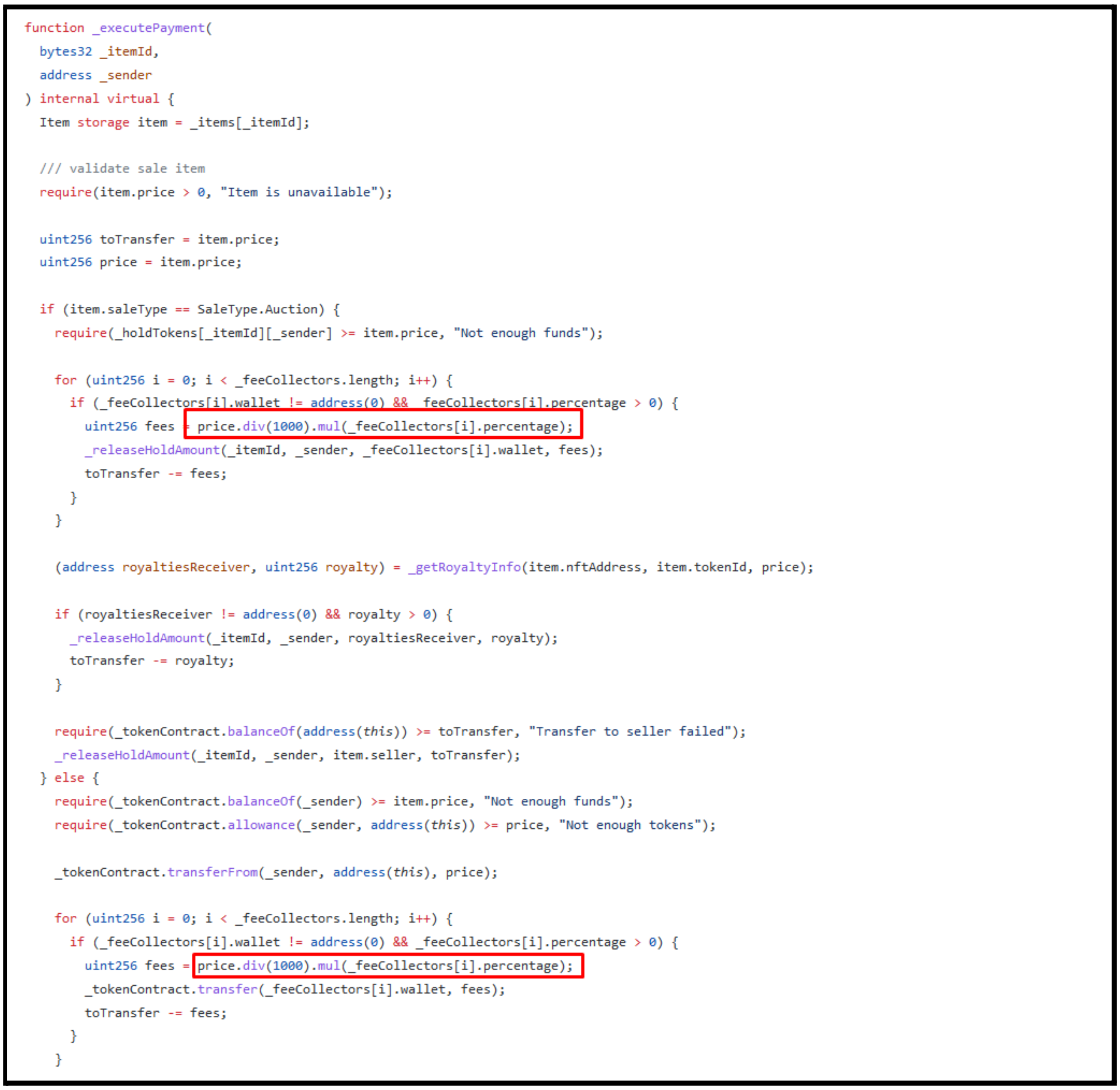
In the claimRewards function, users can claim their rewards everyday. But that reward amount has not been decreased from what the owner has assigned to that user.

Resolution: We suggest correcting the logic for claimRewards to avoid funds draining from the contract. If this is a part of the plan then check for the _totalReward, it does not allow the user to claim if _totalReward reached to 0.

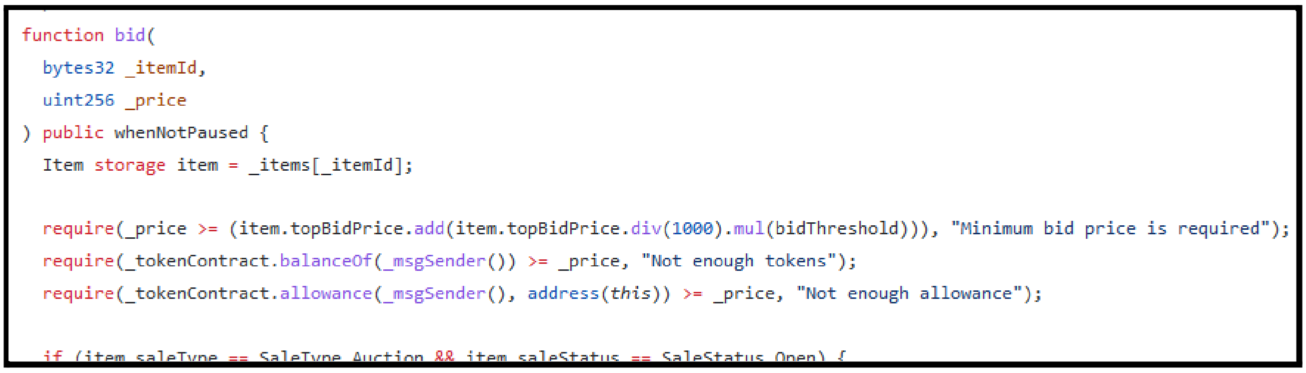
Status: Fixed

Medium

(1) Division before multiplication: **Marketplace.sol**



```
function _executePayment(
  bytes32 _itemId,
  address _sender
) internal virtual {
  Item storage item = _items[_itemId];

  /// validate sale item
  require(item.price > 0, "Item is unavailable");

  uint256 toTransfer = item.price;
  uint256 price = item.price;

  if (item.saleType == SaleType.Auction) {
    require(_holdTokens[_itemId][_sender] >= item.price, "Not enough funds");

    for (uint256 i = 0; i < _feeCollectors.length; i++) {
      if (_feeCollectors[i].wallet != address(0) && _feeCollectors[i].percentage > 0) {
        uint256 fees = price.div(1000).mul(_feeCollectors[i].percentage);
        _releaseHoldAmount(_itemId, _sender, _feeCollectors[i].wallet, fees);
        toTransfer -= fees;
      }
    }

    (address royaltiesReceiver, uint256 royalty) = _getRoyaltyInfo(item.nftAddress, item.tokenId, price);

    if (royaltiesReceiver != address(0) && royalty > 0) {
      _releaseHoldAmount(_itemId, _sender, royaltiesReceiver, royalty);
      toTransfer -= royalty;
    }

    require(_tokenContract.balanceOf(address(this)) >= toTransfer, "Transfer to seller failed");
    _releaseHoldAmount(_itemId, _sender, item.seller, toTransfer);
  } else {
    require(_tokenContract.balanceOf(_sender) >= item.price, "Not enough funds");
    require(_tokenContract.allowance(_sender, address(this)) >= price, "Not enough tokens");

    _tokenContract.transferFrom(_sender, address(this), price);

    for (uint256 i = 0; i < _feeCollectors.length; i++) {
      if (_feeCollectors[i].wallet != address(0) && _feeCollectors[i].percentage > 0) {
        uint256 fees = price.div(1000).mul(_feeCollectors[i].percentage);
        _tokenContract.transfer(_feeCollectors[i].wallet, fees);
        toTransfer -= fees;
      }
    }
```

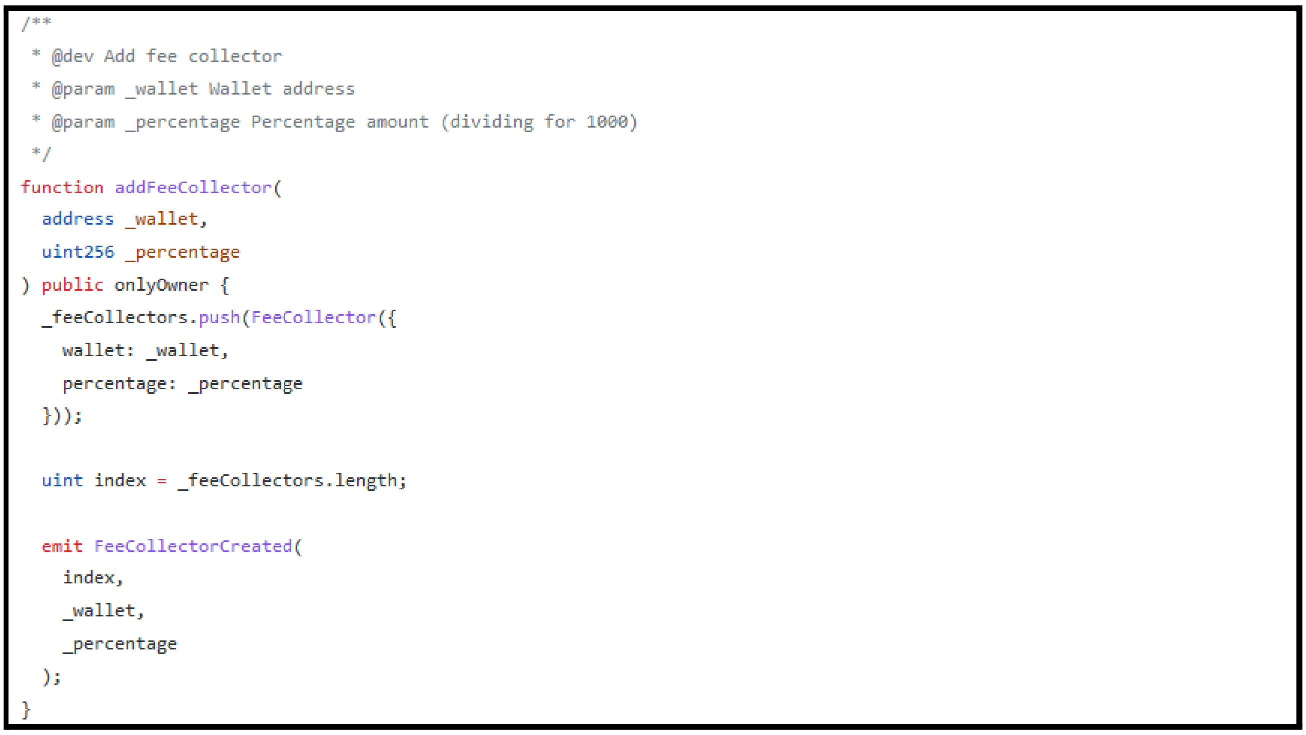


```
function bid(
  bytes32 _itemId,
  uint256 _price
) public whenNotPaused {
  Item storage item = _items[_itemId];

  require(_price >= (item.topBidPrice.add(item.topBidPrice.div(1000).mul(bidThreshold))), "Minimum bid price is required");
  require(_tokenContract.balanceOf(_msgSender()) >= _price, "Not enough tokens");
  require(_tokenContract.allowance(_msgSender(), address(this)) >= _price, "Not enough allowance");

  if (item.saleType == SaleType.Auction && item.saleStatus == SaleStatus.Open) {
```

Solidity being resource constrained language, dividing any amount and then multiplying will cause discrepancies in the outcome. Therefore always multiply the amount first and then divide it

Resolution: Consider ordering multiplication before division.
Status: **Fixed**

(2) Fee validation: **Marketplace.sol**



```
/**
 * @dev Add fee collector
 * @param _wallet Wallet address
 * @param _percentage Percentage amount (dividing for 1000)
 */
function addFeeCollector(
  address _wallet,
  uint256 _percentage
) public onlyOwner {
  _feeCollectors.push(FeeCollector({
    wallet: _wallet,
    percentage: _percentage
  }));

  uint index = _feeCollectors.length;

  emit FeeCollectorCreated(
    index,
    _wallet,
    _percentage
  );
}
```

The owner can set the fee percentage to 100%. so the seller cannot get any amount for his NFT.

Resolution: We suggest using some maximum limit for fees.
Status: **Fixed**

(3) Owner should not be allowed to bid/buy his own auction/sell: **Marketplace.sol**
Auction owner can place a bid for his own auction and can buy his own items.

Resolution: We suggest not allowing the auction owner to place a bid for his own auction or buying his own items.
Status: **Fixed**

Low

(1) Bid can be placed with 0 price: **Marketplace.sol**

Users can place a bid with 0 price.

Resolution: We suggest checking for price while bidding.

Status: Acknowledged

Very Low / Informational / Best practices:

(1) SafeMath Library: **Marketplace.sol**

SafeMath Library is used in this contract code, but the compiler version is greater than or equal to 0.8.0, Then it will not be required to use it, solidity automatically handles overflow / underflow.

Resolution: Remove the SafeMath library and use normal math operators, It will improve code size, and less gas consumption.

Status: Acknowledged

(2) Unused event: **Marketplace.sol**

The SwapApproved() event is defined but not used in code.

Resolution: We suggest removing unused events.

Status: **Fixed**

(3) Compile time error: **Marketplace.sol**

1 - ParserError: Only state variables or file-level variables can have a docstring.

2 - DocstringParsingError: Documentation tag @notice not valid for non-public state variables.

Resolution: Remove single slash before this comment - release nft and transfer it to the seller. There are three slashes added in the comment.

Status: **Fixed**

(4) Unused variables: **WBGame.sol**

There are many variables defined but not used anywhere.

Variables are: _nftAddress, _caller, _treasury, _trainingCost, _forgingCost

Resolution: Remove unused variables from the code.

Status: Fixed

Centralization

This smart contract has some functions which can be executed by the Admin (Owner) only. If the admin wallet private key would be compromised, then it would create trouble. Following are Admin functions:

- setReward: WBGame owner can set address mapping to Rewards.
- batchSetReward: WBGame owner can set address mapping to Rewards.
- safeDispatch: WBGame owner can set the Force Dispatch token from this contract.
- viewTotalRewards: WBGame owner can view total rewards.
- setTokenAddress: Marketplace owner can set ERC20 contract address.
- pause: Marketplace owners can trigger a stopped state.
- unpause: Marketplace owners can return to their normal state.
- createCollection: Marketplace owners can create collections.
- removeCollection: Marketplace owners can Remove collection.
- updateCollection: Marketplace owners can update collection
- getFeeCollectors: Marketplace owners can get fee collectors.
- addFeeCollector: Marketplace owners can add fee collectors.
- removeFeeCollector: Marketplace owners can remove fee collectors.
- emergencyTransferTo: Marketplace owners can transfer NFT to the user for emergency purposes.
- emergencyCancel: Marketplace can emergency cancel sale item by admin
- setJobExecutor: Marketplace owners can set job executors.
- setBidThreshold: Marketplace owners can set bid threshold.
- setPublicationFee: Marketplace owners can set publication fee.
- setPublicationFeeWallet: Marketplace owners can set the address of the publication fee.
- getPublicationFeeWallet: Marketplace owners can get the address of the publication fee.
- executeJob: Marketplace owners can execute all expired auctions.

To make the smart contract 100% decentralized, we suggest renouncing ownership in the smart contract once its function is completed.

Conclusion

We were given a contract code in the form of Github weblink. And we have used all possible tests based on given objects as files.We have observed some major issues and those issues have been fixed. So, **the smart contract is good to go to production.**

Since possible test cases can be unlimited for such smart contracts protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan everything.

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. Smart Contract's high-level description of functionality was presented in the As-is overview section of the report.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

Security state of the reviewed contract, based on standard audit procedure scope, is **"Secured".**

Our Methodology

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our security audits are to improve the quality of systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

Manual Code Review:
In manually reviewing all of the code, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

Vulnerability Analysis:
Our audit techniques included manual code analysis, user interface interaction, and whitebox penetration testing. We look at the project's web site to get a high level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. While we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation.

Documenting Results:

We follow a conservative, transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even though we have not yet verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this we analyze the feasibility of an attack in a live system.

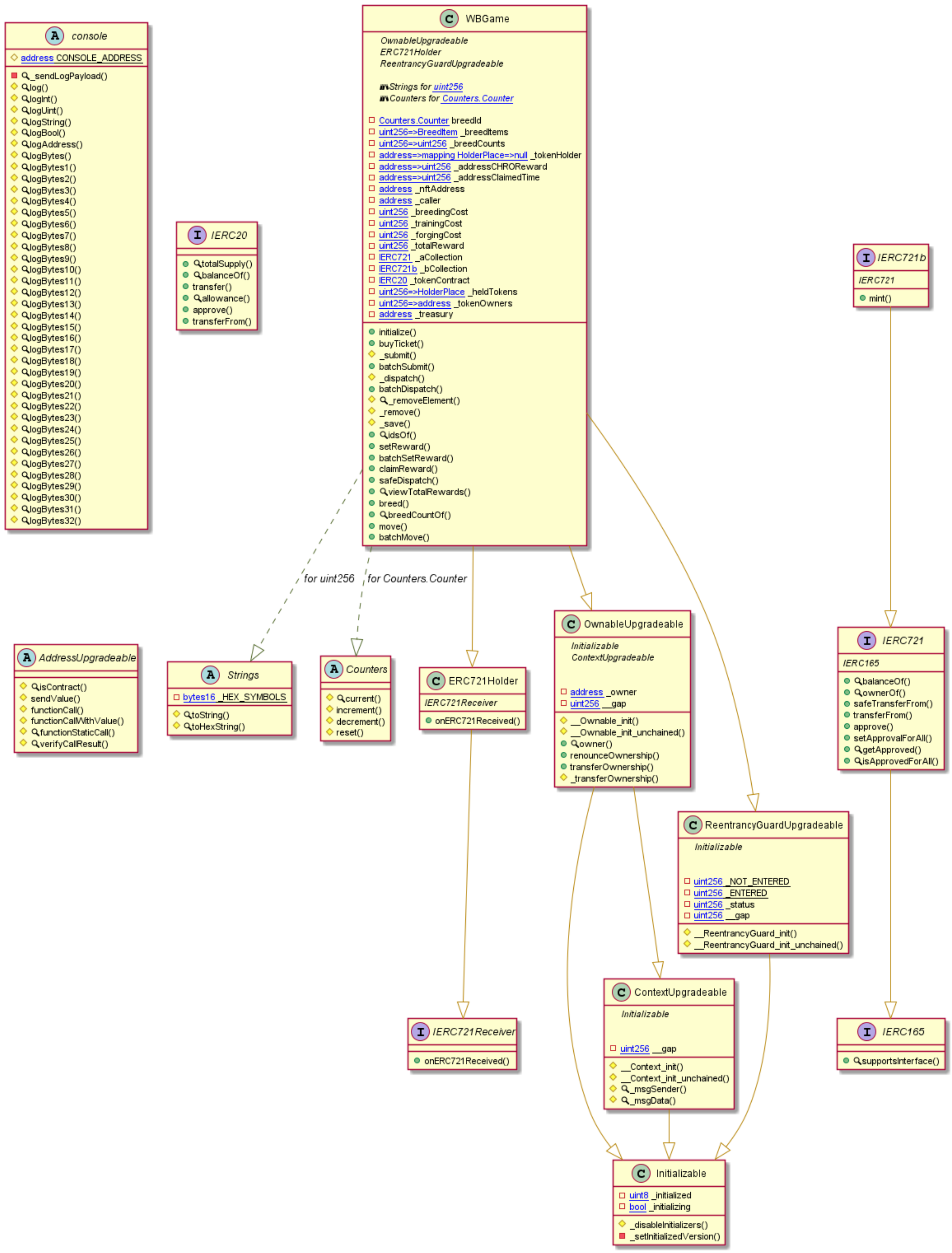
Suggested Solutions:

We search for immediate mitigations that live deployments can take, and finally we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinized by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our report, and before the details are made public.

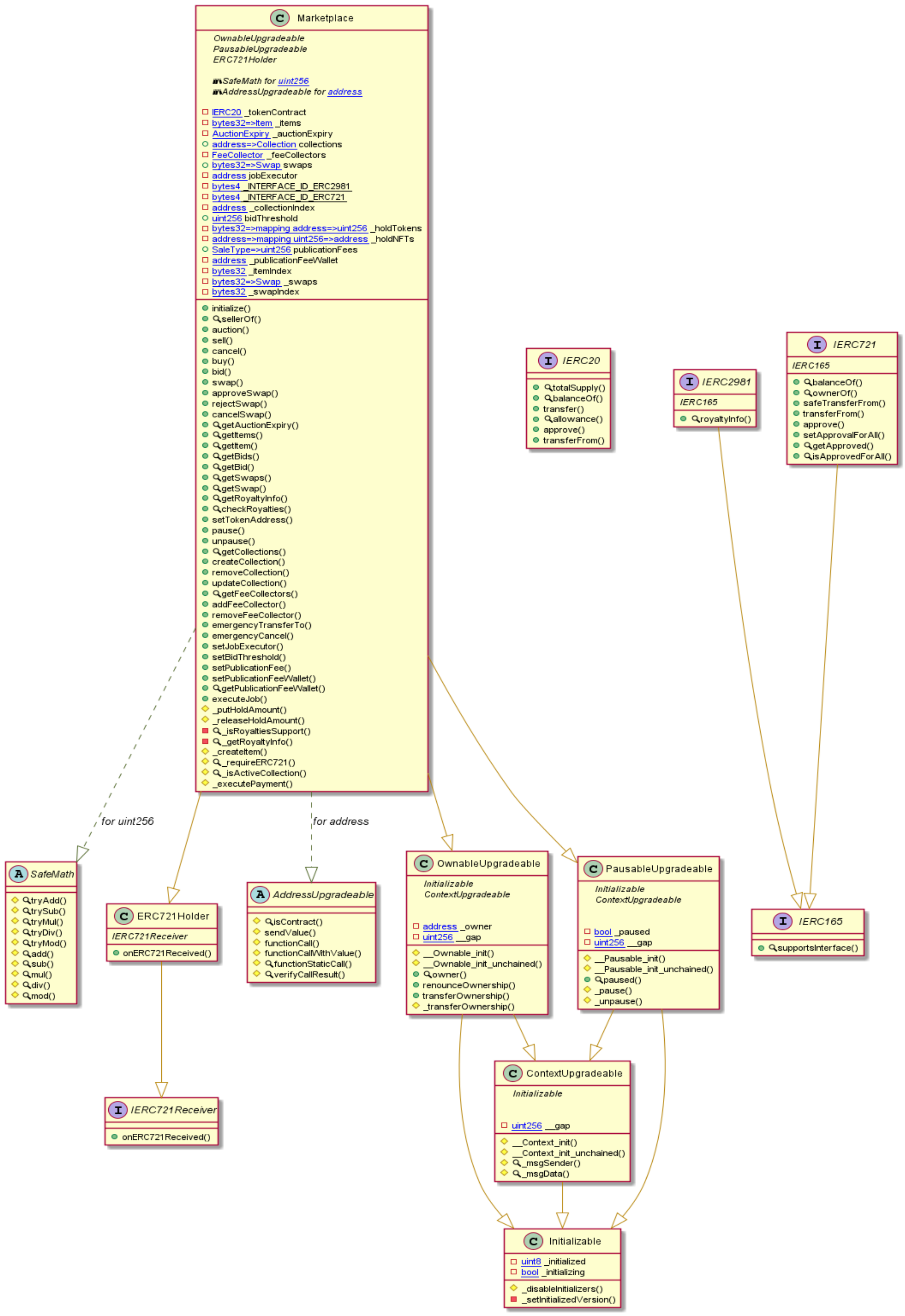
Appendix

Code Flow Diagram - Wyndblast Protocol

WBGame Diagram

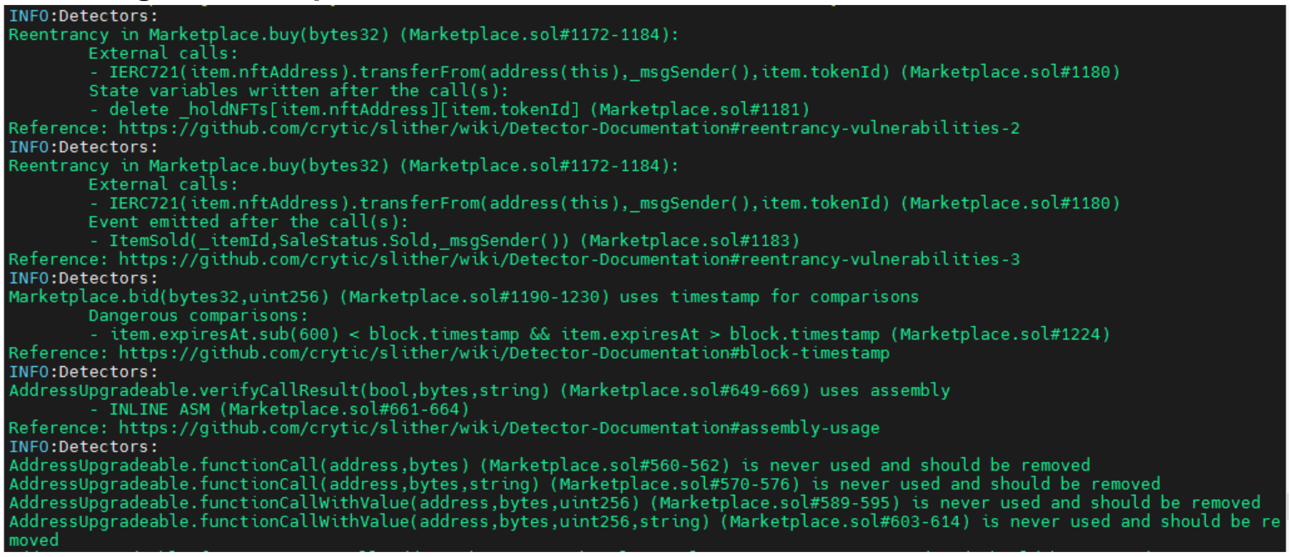


Marketplace Diagram

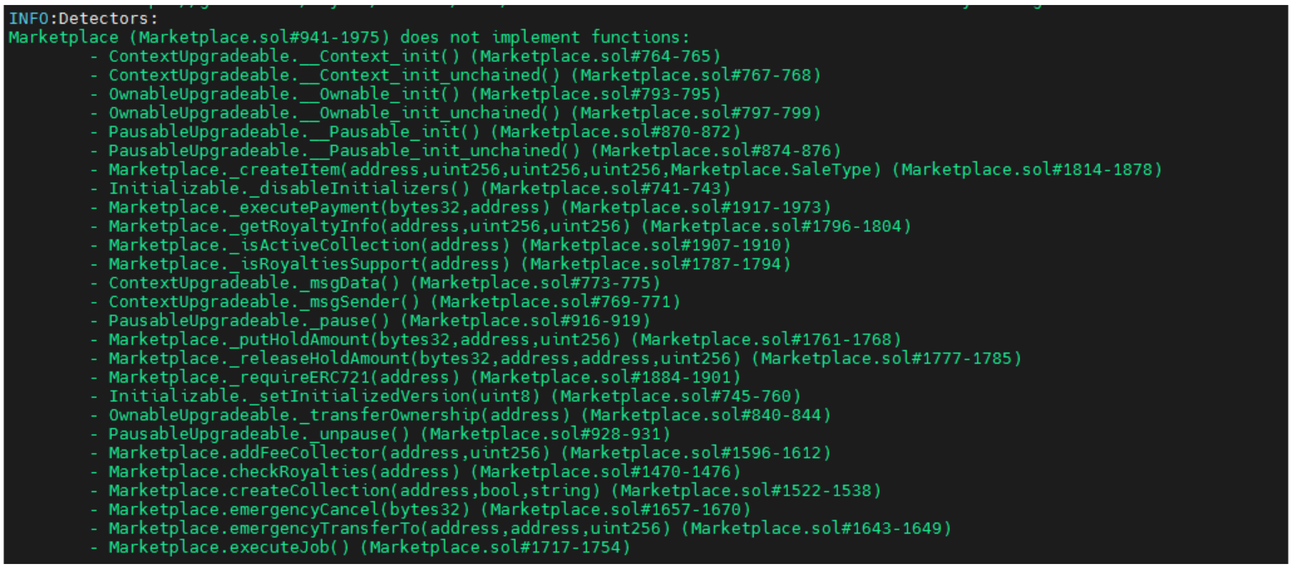


Slither Results Log

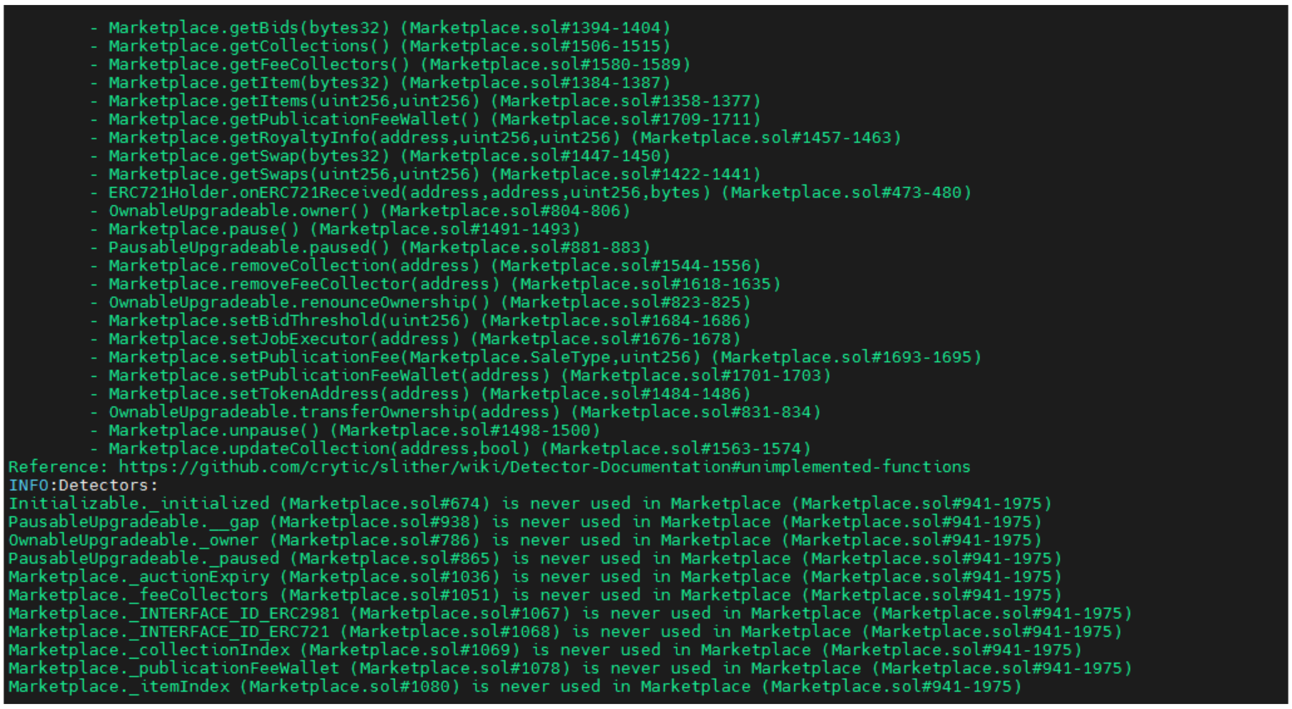
Slither log >> Marketplace.sol



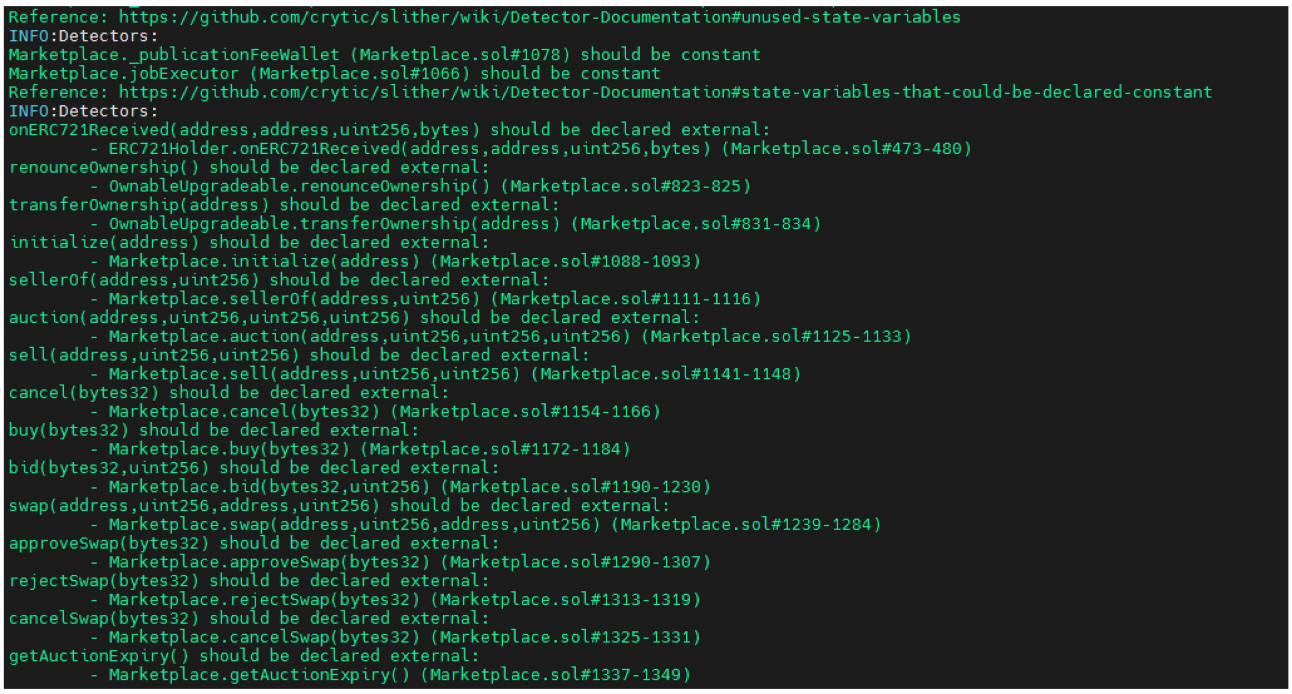
```
INFO:Detectors:
Reentrancy in Marketplace.buy(bytes32) (Marketplace.sol#1172-1184):
        External calls:
        - IERC721(item.nftAddress).transferFrom(address(this),_msgSender(),item.tokenId) (Marketplace.sol#1180)
        State variables written after the call(s):
        - delete _holdNFTs[item.nftAddress][item.tokenId] (Marketplace.sol#1181)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in Marketplace.buy(bytes32) (Marketplace.sol#1172-1184):
        External calls:
        - IERC721(item.nftAddress).transferFrom(address(this),_msgSender(),item.tokenId) (Marketplace.sol#1180)
        Event emitted after the call(s):
        - ItemSold(_itemId,SaleStatus.Sold,_msgSender()) (Marketplace.sol#1183)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Marketplace.bid(bytes32,uint256) (Marketplace.sol#1190-1230) uses timestamp for comparisons
        Dangerous comparisons:
        - item.expiresAt.sub(600) < block.timestamp && item.expiresAt > block.timestamp (Marketplace.sol#1224)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
AddressUpgradeable.verifyCallResult(bool,bytes,string) (Marketplace.sol#649-669) uses assembly
        - INLINE ASM (Marketplace.sol#661-664)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
AddressUpgradeable.functionCall(address,bytes) (Marketplace.sol#560-562) is never used and should be removed
AddressUpgradeable.functionCall(address,bytes,string) (Marketplace.sol#570-576) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (Marketplace.sol#589-595) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (Marketplace.sol#603-614) is never used and should be re
moved
```



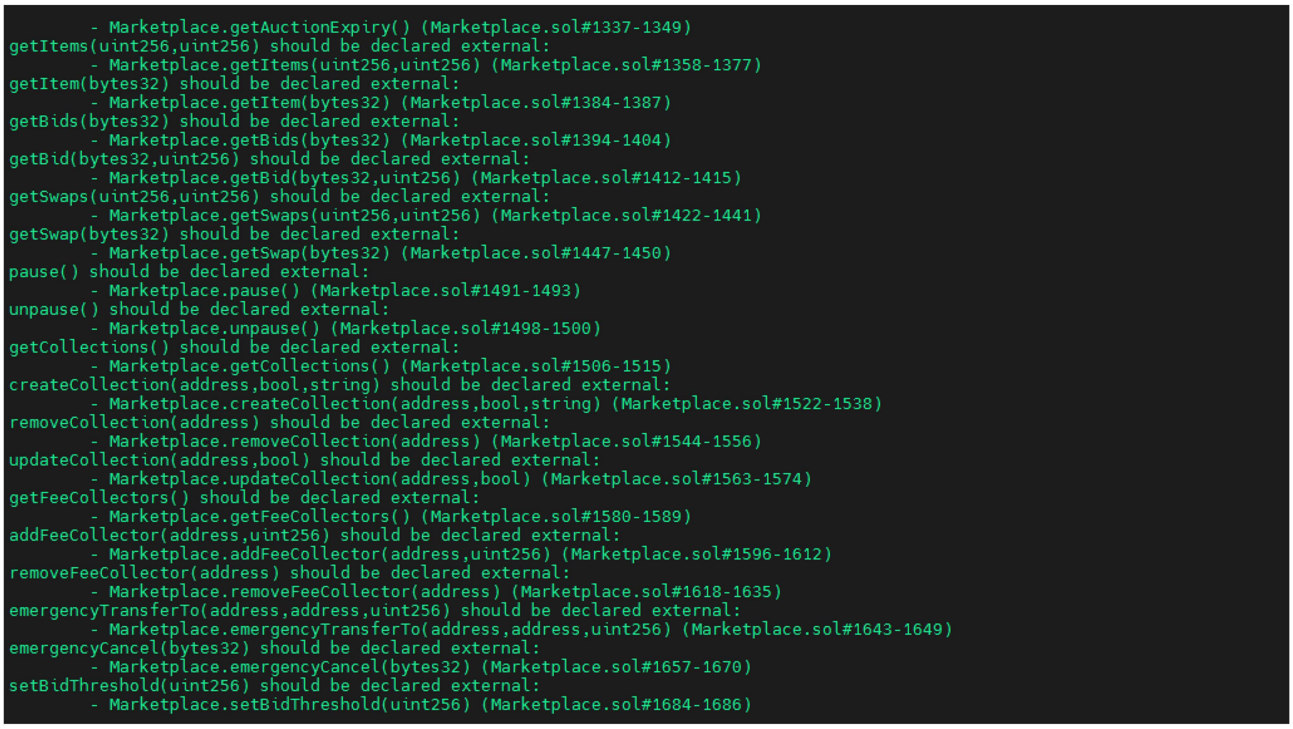
```
INFO:Detectors:
Marketplace (Marketplace.sol#941-1975) does not implement functions:
        - ContextUpgradeable.__Context_init() (Marketplace.sol#764-765)
        - ContextUpgradeable.__Context_init_unchained() (Marketplace.sol#767-768)
        - OwnableUpgradeable.__Ownable_init() (Marketplace.sol#793-795)
        - OwnableUpgradeable.__Ownable_init_unchained() (Marketplace.sol#797-799)
        - PausableUpgradeable.__Pausable_init() (Marketplace.sol#870-872)
        - PausableUpgradeable.__Pausable_init_unchained() (Marketplace.sol#874-876)
        - Marketplace._createItem(address,uint256,uint256,uint256,Marketplace.SaleType) (Marketplace.sol#1814-1878)
        - Initializable._disableInitializers() (Marketplace.sol#741-743)
        - Marketplace._executePayment(bytes32,address) (Marketplace.sol#1917-1973)
        - Marketplace._getRoyaltyInfo(address,uint256,uint256) (Marketplace.sol#1796-1804)
        - Marketplace._isActiveCollection(address) (Marketplace.sol#1907-1910)
        - Marketplace._isRoyaltiesSupport(address) (Marketplace.sol#1787-1794)
        - ContextUpgradeable._msgData() (Marketplace.sol#773-775)
        - ContextUpgradeable._msgSender() (Marketplace.sol#769-771)
        - PausableUpgradeable._pause() (Marketplace.sol#916-919)
        - Marketplace._putHoldAmount(bytes32,address,uint256) (Marketplace.sol#1761-1768)
        - Marketplace._releaseHoldAmount(bytes32,address,address,uint256) (Marketplace.sol#1777-1785)
        - Marketplace._requireERC721(address) (Marketplace.sol#1884-1901)
        - Initializable._setInitializedVersion(uint8) (Marketplace.sol#745-760)
        - OwnableUpgradeable._transferOwnership(address) (Marketplace.sol#840-844)
        - PausableUpgradeable._unpause() (Marketplace.sol#928-931)
        - Marketplace.addFeeCollector(address,uint256) (Marketplace.sol#1596-1612)
        - Marketplace.checkRoyalties(address) (Marketplace.sol#1470-1476)
        - Marketplace.createCollection(address,bool,string) (Marketplace.sol#1522-1538)
        - Marketplace.emergencyCancel(bytes32) (Marketplace.sol#1657-1670)
        - Marketplace.emergencyTransferTo(address,address,uint256) (Marketplace.sol#1643-1649)
        - Marketplace.executeJob() (Marketplace.sol#1717-1754)
```



```
        - Marketplace.getBids(bytes32) (Marketplace.sol#1394-1404)
        - Marketplace.getCollections() (Marketplace.sol#1506-1515)
        - Marketplace.getFeeCollectors() (Marketplace.sol#1580-1589)
        - Marketplace.getItem(bytes32) (Marketplace.sol#1384-1387)
        - Marketplace.getItems(uint256,uint256) (Marketplace.sol#1358-1377)
        - Marketplace.getPublicationFeeWallet() (Marketplace.sol#1709-1711)
        - Marketplace.getRoyaltyInfo(address,uint256,uint256) (Marketplace.sol#1457-1463)
        - Marketplace.getSwap(bytes32) (Marketplace.sol#1447-1450)
        - Marketplace.getSwaps(uint256,uint256) (Marketplace.sol#1422-1441)
        - ERC721Holder.onERC721Received(address,address,uint256,bytes) (Marketplace.sol#473-480)
        - OwnableUpgradeable.owner() (Marketplace.sol#804-806)
        - Marketplace.pause() (Marketplace.sol#1491-1493)
        - PausableUpgradeable.paused() (Marketplace.sol#881-883)
        - Marketplace.removeCollection(address) (Marketplace.sol#1544-1556)
        - Marketplace.removeFeeCollector(address) (Marketplace.sol#1618-1635)
        - OwnableUpgradeable.renounceOwnership() (Marketplace.sol#823-825)
        - Marketplace.setBidThreshold(uint256) (Marketplace.sol#1684-1686)
        - Marketplace.setJobExecutor(address) (Marketplace.sol#1676-1678)
        - Marketplace.setPublicationFee(Marketplace.SaleType,uint256) (Marketplace.sol#1693-1695)
        - Marketplace.setPublicationFeeWallet(address) (Marketplace.sol#1701-1703)
        - Marketplace.setTokenAddress(address) (Marketplace.sol#1484-1486)
        - OwnableUpgradeable.transferOwnership(address) (Marketplace.sol#831-834)
        - Marketplace.unpause() (Marketplace.sol#1498-1500)
        - Marketplace.updateCollection(address,bool) (Marketplace.sol#1563-1574)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions
INFO:Detectors:
Initializable._initialized (Marketplace.sol#674) is never used in Marketplace (Marketplace.sol#941-1975)
PausableUpgradeable.__gap (Marketplace.sol#938) is never used in Marketplace (Marketplace.sol#941-1975)
OwnableUpgradeable._owner (Marketplace.sol#786) is never used in Marketplace (Marketplace.sol#941-1975)
PausableUpgradeable._paused (Marketplace.sol#865) is never used in Marketplace (Marketplace.sol#941-1975)
Marketplace._auctionExpiry (Marketplace.sol#1036) is never used in Marketplace (Marketplace.sol#941-1975)
Marketplace._feeCollectors (Marketplace.sol#1051) is never used in Marketplace (Marketplace.sol#941-1975)
Marketplace._INTERFACE_ID_ERC2981 (Marketplace.sol#1067) is never used in Marketplace (Marketplace.sol#941-1975)
Marketplace._INTERFACE_ID_ERC721 (Marketplace.sol#1068) is never used in Marketplace (Marketplace.sol#941-1975)
Marketplace._collectionIndex (Marketplace.sol#1069) is never used in Marketplace (Marketplace.sol#941-1975)
Marketplace._publicationFeeWallet (Marketplace.sol#1078) is never used in Marketplace (Marketplace.sol#941-1975)
Marketplace._itemIndex (Marketplace.sol#1080) is never used in Marketplace (Marketplace.sol#941-1975)
```

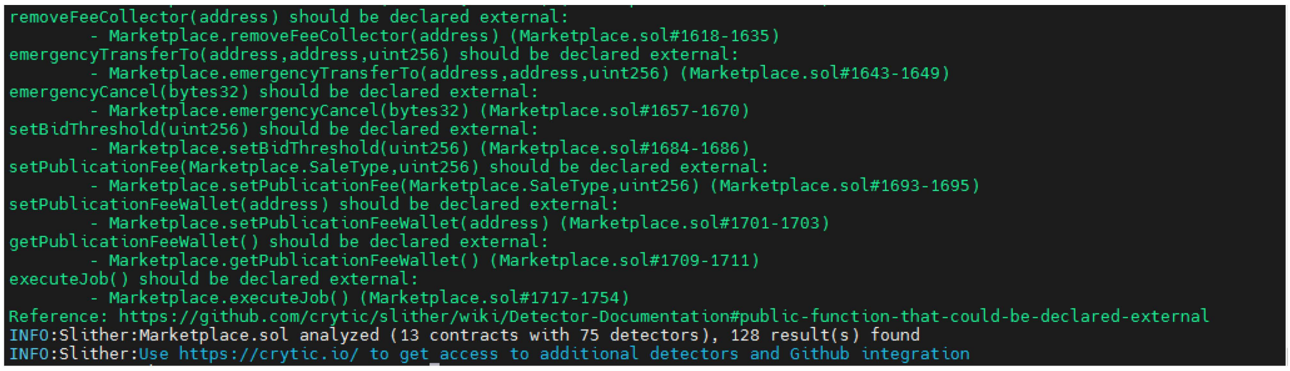


```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
Marketplace._publicationFeeWallet (Marketplace.sol#1078) should be constant
Marketplace.jobExecutor (Marketplace.sol#1066) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
onERC721Received(address,address,uint256,bytes) should be declared external:
        - ERC721Holder.onERC721Received(address,address,uint256,bytes) (Marketplace.sol#473-480)
renounceOwnership() should be declared external:
        - OwnableUpgradeable.renounceOwnership() (Marketplace.sol#823-825)
transferOwnership(address) should be declared external:
        - OwnableUpgradeable.transferOwnership(address) (Marketplace.sol#831-834)
initialize(address) should be declared external:
        - Marketplace.initialize(address) (Marketplace.sol#1088-1093)
sellerOf(address,uint256) should be declared external:
        - Marketplace.sellerOf(address,uint256) (Marketplace.sol#1111-1116)
auction(address,uint256,uint256,uint256) should be declared external:
        - Marketplace.auction(address,uint256,uint256,uint256) (Marketplace.sol#1125-1133)
sell(address,uint256,uint256) should be declared external:
        - Marketplace.sell(address,uint256,uint256) (Marketplace.sol#1141-1148)
cancel(bytes32) should be declared external:
        - Marketplace.cancel(bytes32) (Marketplace.sol#1154-1166)
buy(bytes32) should be declared external:
        - Marketplace.buy(bytes32) (Marketplace.sol#1172-1184)
bid(bytes32,uint256) should be declared external:
        - Marketplace.bid(bytes32,uint256) (Marketplace.sol#1190-1230)
swap(address,uint256,address,uint256) should be declared external:
        - Marketplace.swap(address,uint256,address,uint256) (Marketplace.sol#1239-1284)
approveSwap(bytes32) should be declared external:
        - Marketplace.approveSwap(bytes32) (Marketplace.sol#1290-1307)
rejectSwap(bytes32) should be declared external:
        - Marketplace.rejectSwap(bytes32) (Marketplace.sol#1313-1319)
cancelSwap(bytes32) should be declared external:
        - Marketplace.cancelSwap(bytes32) (Marketplace.sol#1325-1331)
getAuctionExpiry() should be declared external:
        - Marketplace.getAuctionExpiry() (Marketplace.sol#1337-1349)
```



```
        - Marketplace.getAuctionExpiry() (Marketplace.sol#1337-1349)
```

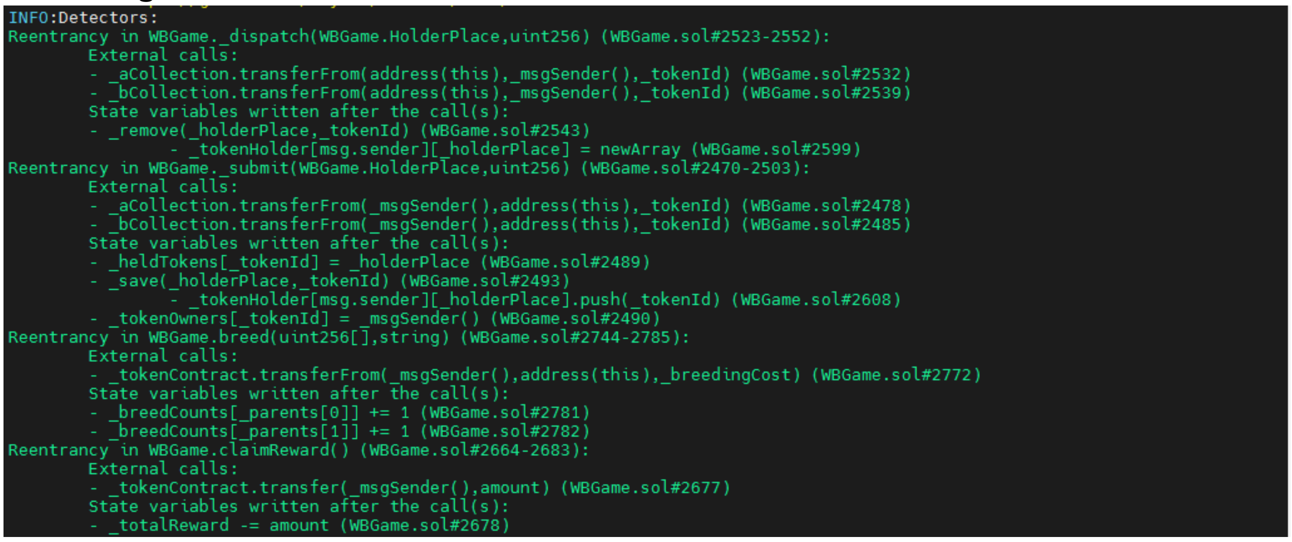
```
getItems(uint256,uint256) should be declared external:
        - Marketplace.getItems(uint256,uint256) (Marketplace.sol#1358-1377)
getItem(bytes32) should be declared external:
        - Marketplace.getItem(bytes32) (Marketplace.sol#1384-1387)
getBids(bytes32) should be declared external:
        - Marketplace.getBids(bytes32) (Marketplace.sol#1394-1404)
getBid(bytes32,uint256) should be declared external:
        - Marketplace.getBid(bytes32,uint256) (Marketplace.sol#1412-1415)
getSwaps(uint256,uint256) should be declared external:
        - Marketplace.getSwaps(uint256,uint256) (Marketplace.sol#1422-1441)
getSwap(bytes32) should be declared external:
        - Marketplace.getSwap(bytes32) (Marketplace.sol#1447-1450)
pause() should be declared external:
        - Marketplace.pause() (Marketplace.sol#1491-1493)
unpause() should be declared external:
        - Marketplace.unpause() (Marketplace.sol#1498-1500)
getCollections() should be declared external:
        - Marketplace.getCollections() (Marketplace.sol#1506-1515)
createCollection(address,bool,string) should be declared external:
        - Marketplace.createCollection(address,bool,string) (Marketplace.sol#1522-1538)
removeCollection(address) should be declared external:
        - Marketplace.removeCollection(address) (Marketplace.sol#1544-1556)
updateCollection(address,bool) should be declared external:
        - Marketplace.updateCollection(address,bool) (Marketplace.sol#1563-1574)
getFeeCollectors() should be declared external:
        - Marketplace.getFeeCollectors() (Marketplace.sol#1580-1589)
addFeeCollector(address,uint256) should be declared external:
        - Marketplace.addFeeCollector(address,uint256) (Marketplace.sol#1596-1612)
removeFeeCollector(address) should be declared external:
        - Marketplace.removeFeeCollector(address) (Marketplace.sol#1618-1635)
emergencyTransferTo(address,address,uint256) should be declared external:
        - Marketplace.emergencyTransferTo(address,address,uint256) (Marketplace.sol#1643-1649)
emergencyCancel(bytes32) should be declared external:
        - Marketplace.emergencyCancel(bytes32) (Marketplace.sol#1657-1670)
setBidThreshold(uint256) should be declared external:
        - Marketplace.setBidThreshold(uint256) (Marketplace.sol#1684-1686)
```



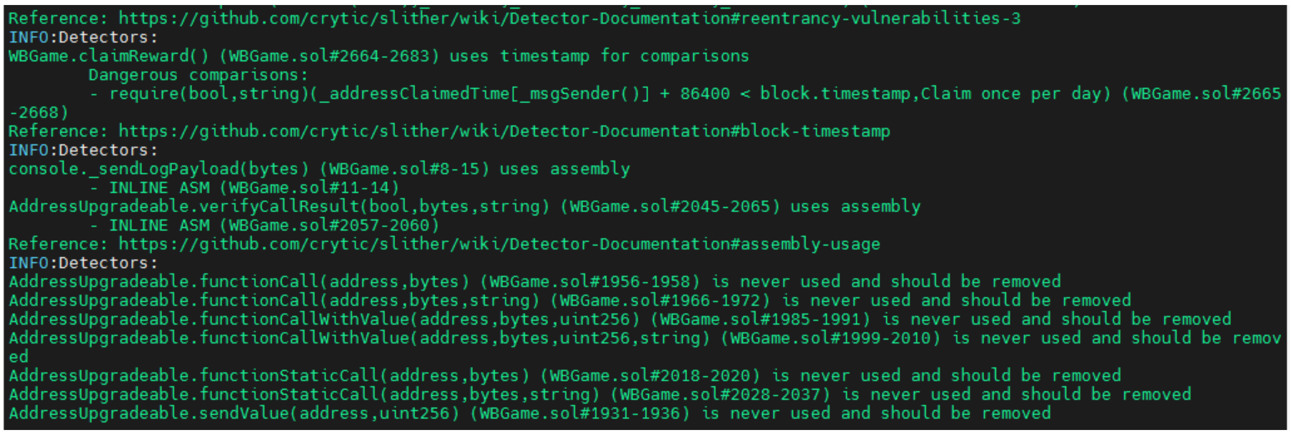
```
removeFeeCollector(address) should be declared external:
        - Marketplace.removeFeeCollector(address) (Marketplace.sol#1618-1635)
emergencyTransferTo(address,address,uint256) should be declared external:
        - Marketplace.emergencyTransferTo(address,address,uint256) (Marketplace.sol#1643-1649)
emergencyCancel(bytes32) should be declared external:
        - Marketplace.emergencyCancel(bytes32) (Marketplace.sol#1657-1670)
setBidThreshold(uint256) should be declared external:
        - Marketplace.setBidThreshold(uint256) (Marketplace.sol#1684-1686)
```

```
setPublicationFee(Marketplace.SaleType,uint256) should be declared external:
        - Marketplace.setPublicationFee(Marketplace.SaleType,uint256) (Marketplace.sol#1693-1695)
setPublicationFeeWallet(address) should be declared external:
        - Marketplace.setPublicationFeeWallet(address) (Marketplace.sol#1701-1703)
getPublicationFeeWallet() should be declared external:
        - Marketplace.getPublicationFeeWallet() (Marketplace.sol#1709-1711)
executeJob() should be declared external:
        - Marketplace.executeJob() (Marketplace.sol#1717-1754)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:Marketplace.sol analyzed (13 contracts with 75 detectors), 128 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

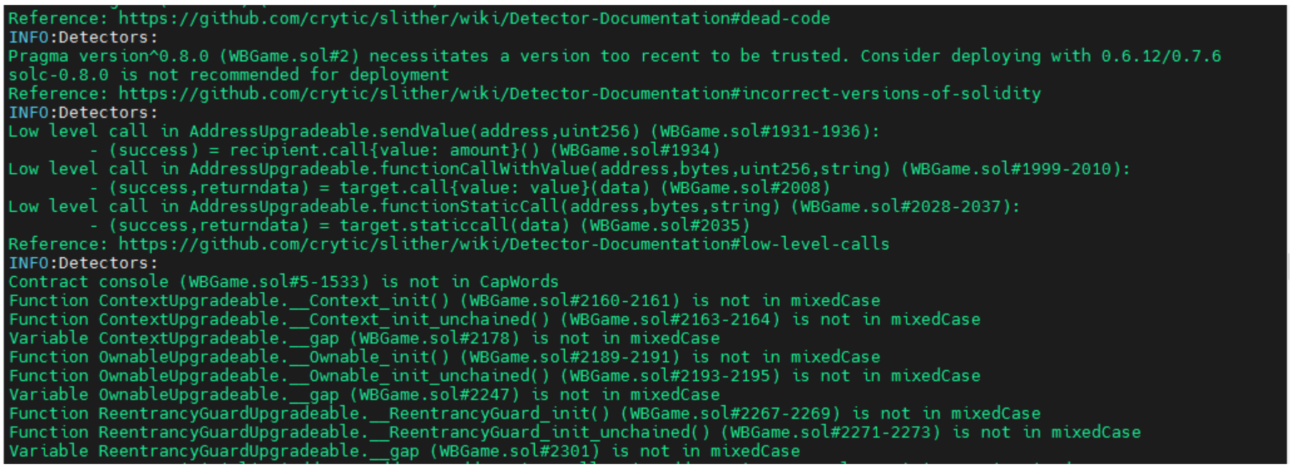
Slither log >> WBGame.sol



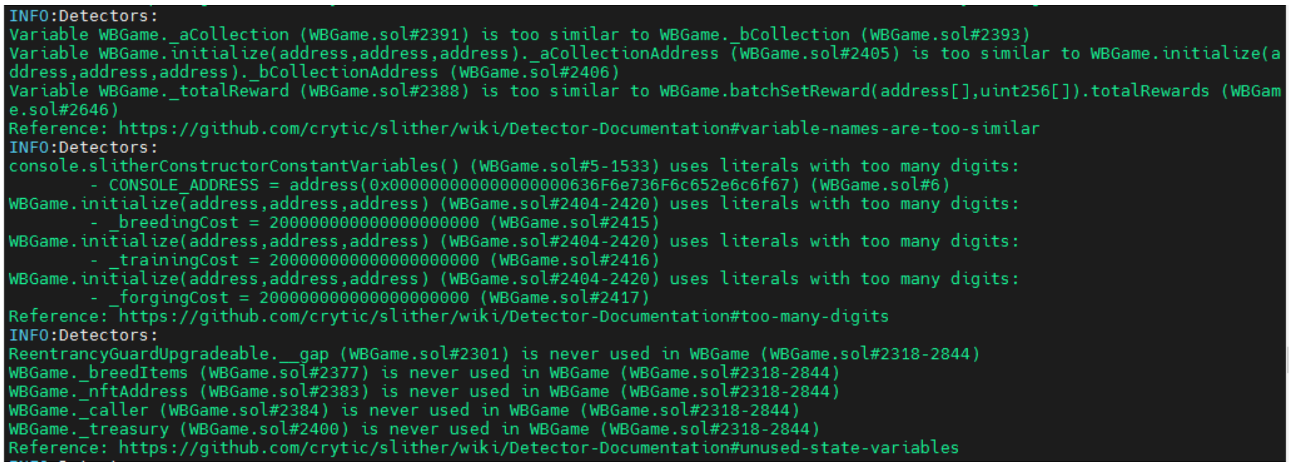
```
INFO:Detectors:
Reentrancy in WBGame._dispatch(WBGame.HolderPlace,uint256) (WBGame.sol#2523-2552):
        External calls:
        - _aCollection.transferFrom(address(this),_msgSender(),_tokenId) (WBGame.sol#2532)
        - _bCollection.transferFrom(address(this),_msgSender(),_tokenId) (WBGame.sol#2539)
        State variables written after the call(s):
        - _remove(_holderPlace,_tokenId) (WBGame.sol#2543)
                - _tokenHolder[msg.sender][_holderPlace] = newArray (WBGame.sol#2599)
Reentrancy in WBGame._submit(WBGame.HolderPlace,uint256) (WBGame.sol#2470-2503):
        External calls:
        - _aCollection.transferFrom(_msgSender(),address(this),_tokenId) (WBGame.sol#2478)
        - _bCollection.transferFrom(_msgSender(),address(this),_tokenId) (WBGame.sol#2485)
        State variables written after the call(s):
        - _heldTokens[_tokenId] = _holderPlace (WBGame.sol#2489)
        - _save(_holderPlace,_tokenId) (WBGame.sol#2493)
                - _tokenHolder[msg.sender][_holderPlace].push(_tokenId) (WBGame.sol#2608)
        - _tokenOwners[_tokenId] = _msgSender() (WBGame.sol#2490)
Reentrancy in WBGame.breed(uint256[],string) (WBGame.sol#2744-2785):
        External calls:
        - _tokenContract.transferFrom(_msgSender(),address(this),_breedingCost) (WBGame.sol#2772)
        State variables written after the call(s):
        - _breedCounts[_parents[0]] += 1 (WBGame.sol#2781)
        - _breedCounts[_parents[1]] += 1 (WBGame.sol#2782)
Reentrancy in WBGame.claimReward() (WBGame.sol#2664-2683):
        External calls:
        - _tokenContract.transfer(_msgSender(),amount) (WBGame.sol#2677)
        State variables written after the call(s):
        - _totalReward -= amount (WBGame.sol#2678)
```



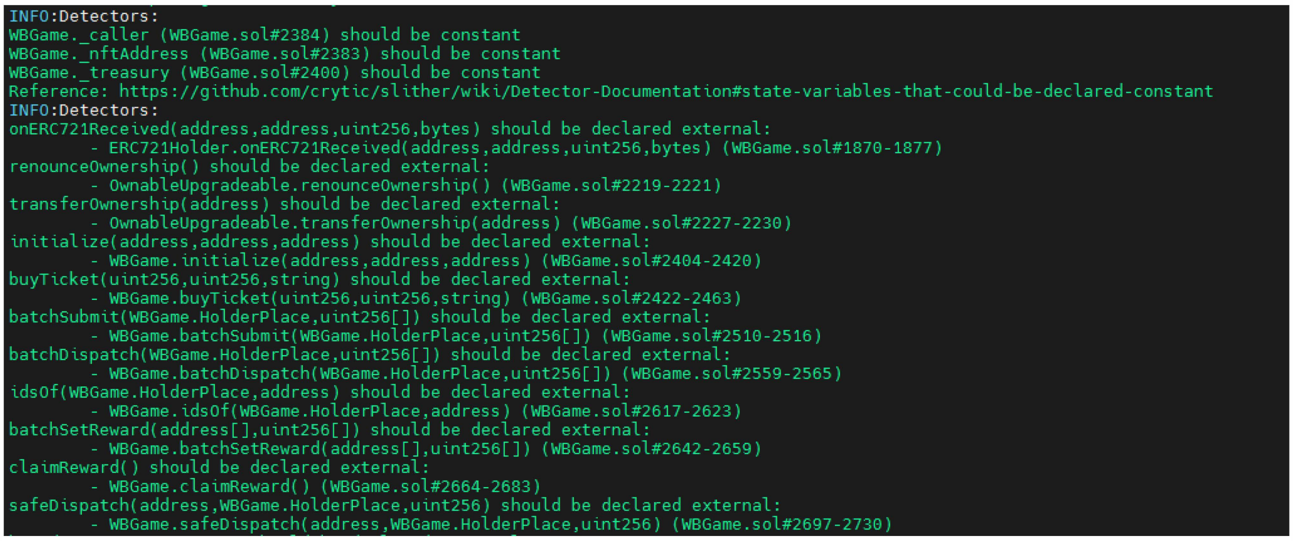
```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
WBGame.claimReward() (WBGame.sol#2664-2683) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(_addressClaimedTime[_msgSender()] + 86400 < block.timestamp,Claim once per day) (WBGame.sol#2665
-2668)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
console._sendLogPayload(bytes) (WBGame.sol#8-15) uses assembly
        - INLINE ASM (WBGame.sol#11-14)
AddressUpgradeable.verifyCallResult(bool,bytes,string) (WBGame.sol#2045-2065) uses assembly
        - INLINE ASM (WBGame.sol#2057-2060)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
AddressUpgradeable.functionCall(address,bytes) (WBGame.sol#1956-1958) is never used and should be removed
AddressUpgradeable.functionCall(address,bytes,string) (WBGame.sol#1966-1972) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (WBGame.sol#1985-1991) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (WBGame.sol#1999-2010) is never used and should be remov
ed
AddressUpgradeable.functionStaticCall(address,bytes) (WBGame.sol#2018-2020) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes,string) (WBGame.sol#2028-2037) is never used and should be removed
AddressUpgradeable.sendValue(address,uint256) (WBGame.sol#1931-1936) is never used and should be removed
```



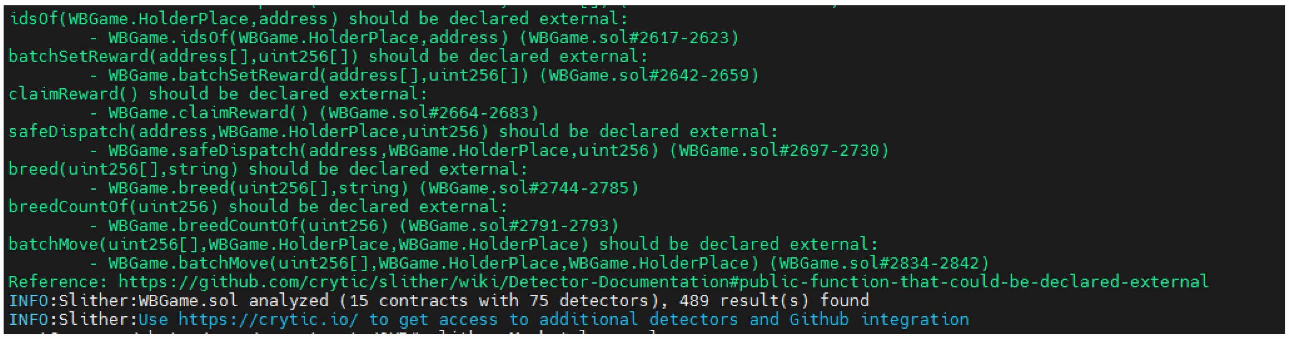
```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (WBGame.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in AddressUpgradeable.sendValue(address,uint256) (WBGame.sol#1931-1936):
        - (success) = recipient.call{value: amount}() (WBGame.sol#1934)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (WBGame.sol#1999-2010):
        - (success,returndata) = target.call{value: value}(data) (WBGame.sol#2008)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (WBGame.sol#2028-2037):
        - (success,returndata) = target.staticcall(data) (WBGame.sol#2035)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Contract console (WBGame.sol#5-1533) is not in CapWords
Function ContextUpgradeable.__Context_init() (WBGame.sol#2160-2161) is not in mixedCase
Function ContextUpgradeable.__Context_init_unchained() (WBGame.sol#2163-2164) is not in mixedCase
Variable ContextUpgradeable.__gap (WBGame.sol#2178) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init() (WBGame.sol#2189-2191) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init_unchained() (WBGame.sol#2193-2195) is not in mixedCase
Variable OwnableUpgradeable.__gap (WBGame.sol#2247) is not in mixedCase
Function ReentrancyGuardUpgradeable.__ReentrancyGuard_init() (WBGame.sol#2267-2269) is not in mixedCase
Function ReentrancyGuardUpgradeable.__ReentrancyGuard_init_unchained() (WBGame.sol#2271-2273) is not in mixedCase
Variable ReentrancyGuardUpgradeable.__gap (WBGame.sol#2301) is not in mixedCase
```



```
INFO:Detectors:
Variable WBGame._aCollection (WBGame.sol#2391) is too similar to WBGame._bCollection (WBGame.sol#2393)
Variable WBGame.initialize(address,address,address)._aCollectionAddress (WBGame.sol#2405) is too similar to WBGame.initialize(a
ddress,address,address)._bCollectionAddress (WBGame.sol#2406)
Variable WBGame._totalReward (WBGame.sol#2388) is too similar to WBGame.batchSetReward(address[],uint256[]).totalRewards (WBGam
e.sol#2646)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
console.slitherConstructorConstantVariables() (WBGame.sol#5-1533) uses literals with too many digits:
        - CONSOLE_ADDRESS = address(0x000000000000000000636F6e736F6c652e6c6f67) (WBGame.sol#6)
WBGame.initialize(address,address,address) (WBGame.sol#2404-2420) uses literals with too many digits:
        - _breedingCost = 20000000000000000000 (WBGame.sol#2415)
WBGame.initialize(address,address,address) (WBGame.sol#2404-2420) uses literals with too many digits:
        - _trainingCost = 20000000000000000000 (WBGame.sol#2416)
WBGame.initialize(address,address,address) (WBGame.sol#2404-2420) uses literals with too many digits:
        - _forgingCost = 20000000000000000000 (WBGame.sol#2417)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
ReentrancyGuardUpgradeable.__gap (WBGame.sol#2301) is never used in WBGame (WBGame.sol#2318-2844)
WBGame._breedItems (WBGame.sol#2377) is never used in WBGame (WBGame.sol#2318-2844)
WBGame._nftAddress (WBGame.sol#2383) is never used in WBGame (WBGame.sol#2318-2844)
WBGame._caller (WBGame.sol#2384) is never used in WBGame (WBGame.sol#2318-2844)
WBGame._treasury (WBGame.sol#2400) is never used in WBGame (WBGame.sol#2318-2844)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
```



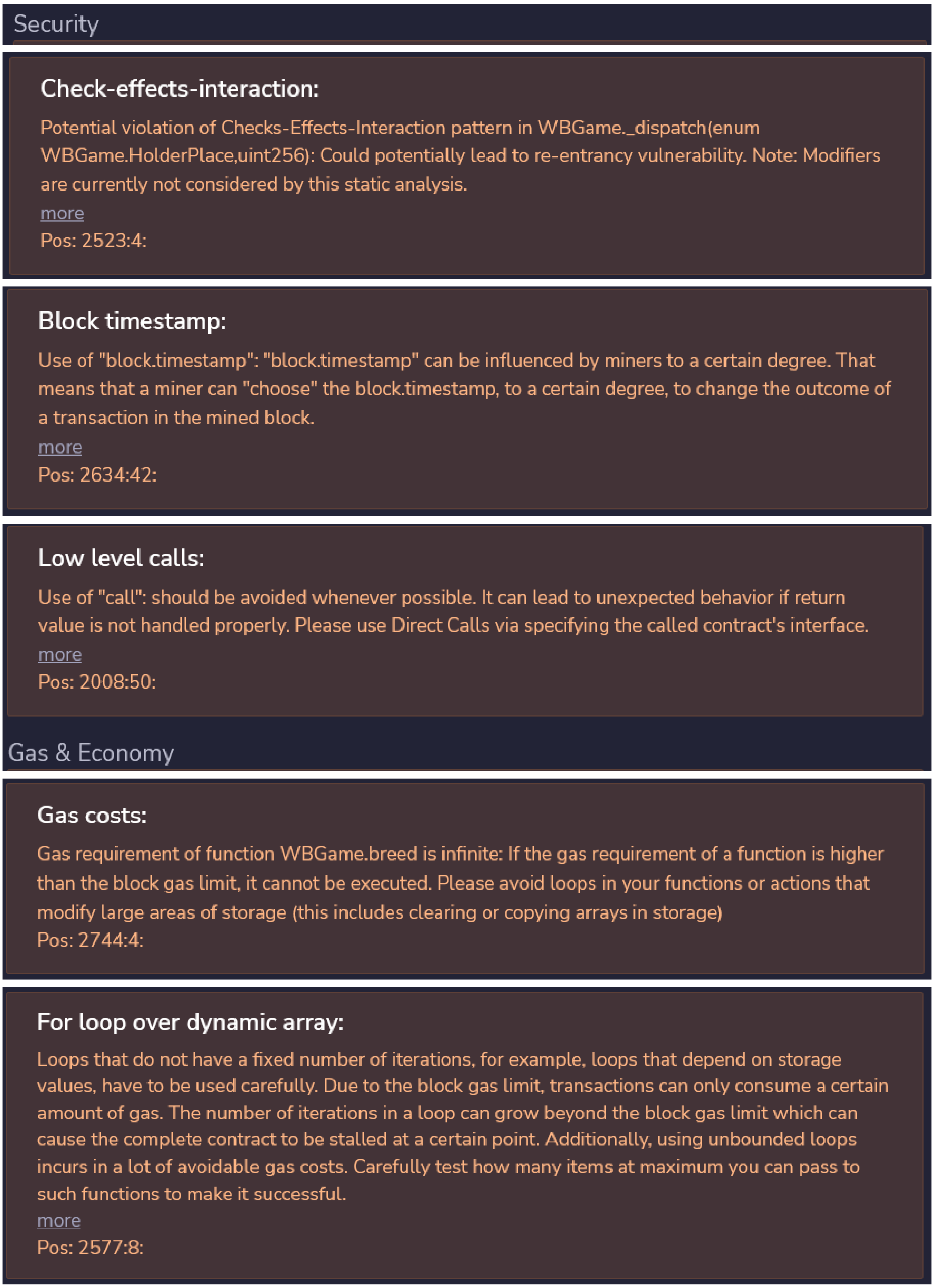
```
INFO:Detectors:
WBGame._caller (WBGame.sol#2384) should be constant
WBGame._nftAddress (WBGame.sol#2383) should be constant
WBGame._treasury (WBGame.sol#2400) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
onERC721Received(address,address,uint256,bytes) should be declared external:
        - ERC721Holder.onERC721Received(address,address,uint256,bytes) (WBGame.sol#1870-1877)
renounceOwnership() should be declared external:
        - OwnableUpgradeable.renounceOwnership() (WBGame.sol#2219-2221)
transferOwnership(address) should be declared external:
        - OwnableUpgradeable.transferOwnership(address) (WBGame.sol#2227-2230)
initialize(address,address,address) should be declared external:
        - WBGame.initialize(address,address,address) (WBGame.sol#2404-2420)
buyTicket(uint256,uint256,string) should be declared external:
        - WBGame.buyTicket(uint256,uint256,string) (WBGame.sol#2422-2463)
batchSubmit(WBGame.HolderPlace,uint256[]) should be declared external:
        - WBGame.batchSubmit(WBGame.HolderPlace,uint256[]) (WBGame.sol#2510-2516)
batchDispatch(WBGame.HolderPlace,uint256[]) should be declared external:
        - WBGame.batchDispatch(WBGame.HolderPlace,uint256[]) (WBGame.sol#2559-2565)
idsOf(WBGame.HolderPlace,address) should be declared external:
        - WBGame.idsOf(WBGame.HolderPlace,address) (WBGame.sol#2617-2623)
batchSetReward(address[],uint256[]) should be declared external:
        - WBGame.batchSetReward(address[],uint256[]) (WBGame.sol#2642-2659)
claimReward() should be declared external:
        - WBGame.claimReward() (WBGame.sol#2664-2683)
safeDispatch(address,WBGame.HolderPlace,uint256) should be declared external:
        - WBGame.safeDispatch(address,WBGame.HolderPlace,uint256) (WBGame.sol#2697-2730)
```



```
idsOf(WBGame.HolderPlace,address) should be declared external:
        - WBGame.idsOf(WBGame.HolderPlace,address) (WBGame.sol#2617-2623)
batchSetReward(address[],uint256[]) should be declared external:
        - WBGame.batchSetReward(address[],uint256[]) (WBGame.sol#2642-2659)
claimReward() should be declared external:
        - WBGame.claimReward() (WBGame.sol#2664-2683)
safeDispatch(address,WBGame.HolderPlace,uint256) should be declared external:
        - WBGame.safeDispatch(address,WBGame.HolderPlace,uint256) (WBGame.sol#2697-2730)
breed(uint256[],string) should be declared external:
        - WBGame.breed(uint256[],string) (WBGame.sol#2744-2785)
breedCountOf(uint256) should be declared external:
        - WBGame.breedCountOf(uint256) (WBGame.sol#2791-2793)
batchMove(uint256[],WBGame.HolderPlace,WBGame.HolderPlace) should be declared external:
        - WBGame.batchMove(uint256[],WBGame.HolderPlace,WBGame.HolderPlace) (WBGame.sol#2834-2842)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:WBGame.sol analyzed (15 contracts with 75 detectors), 489 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Solidity Static Analysis

WBGame.sol



Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in WBGame._dispatch(enum WBGame.HolderPlace,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 2523:4:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 2634:42:

Low level calls:

Use of "call": should be avoided whenever possible. It can lead to unexpected behavior if return value is not handled properly. Please use Direct Calls via specifying the called contract's interface.
more
Pos: 2008:50:

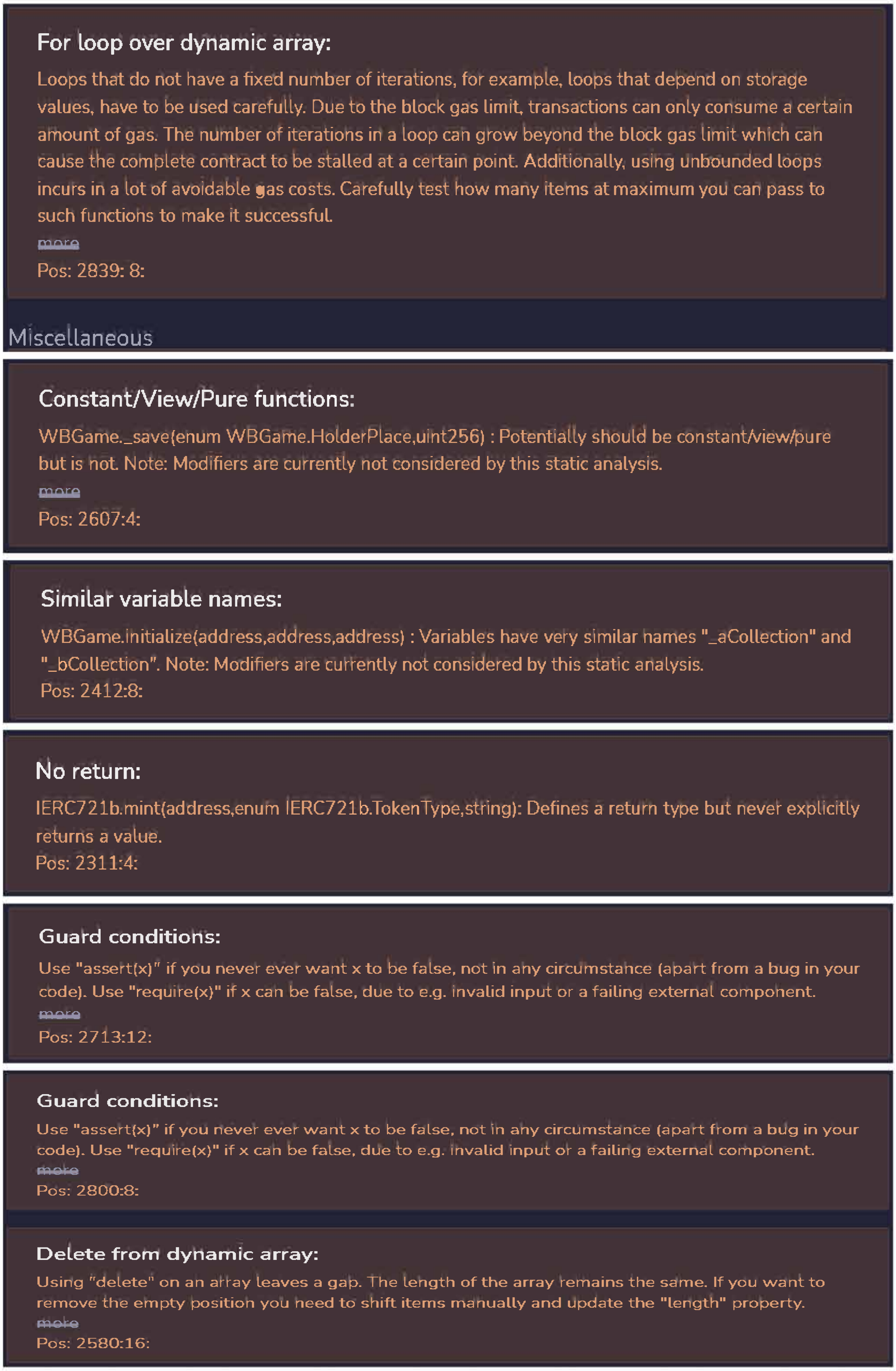
Gas & Economy

Gas costs:

Gas requirement of function WBGame.breed is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 2744:4:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.
more
Pos: 2577:8:



For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.
more
Pos: 2839: 8:

Miscellaneous

Constant/View/Pure functions:

WBGame._save(enum WBGame.HolderPlace,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 2607:4:

Similar variable names:

WBGame.initialize(address,address,address) : Variables have very similar names "_aCollection" and "_bCollection". Note: Modifiers are currently not considered by this static analysis.
Pos: 2412:8:

No return:

IERC721b.mint(address,enum IERC721b.TokenType,string): Defines a return type but never explicitly returns a value.
Pos: 2311:4:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 2713:12:

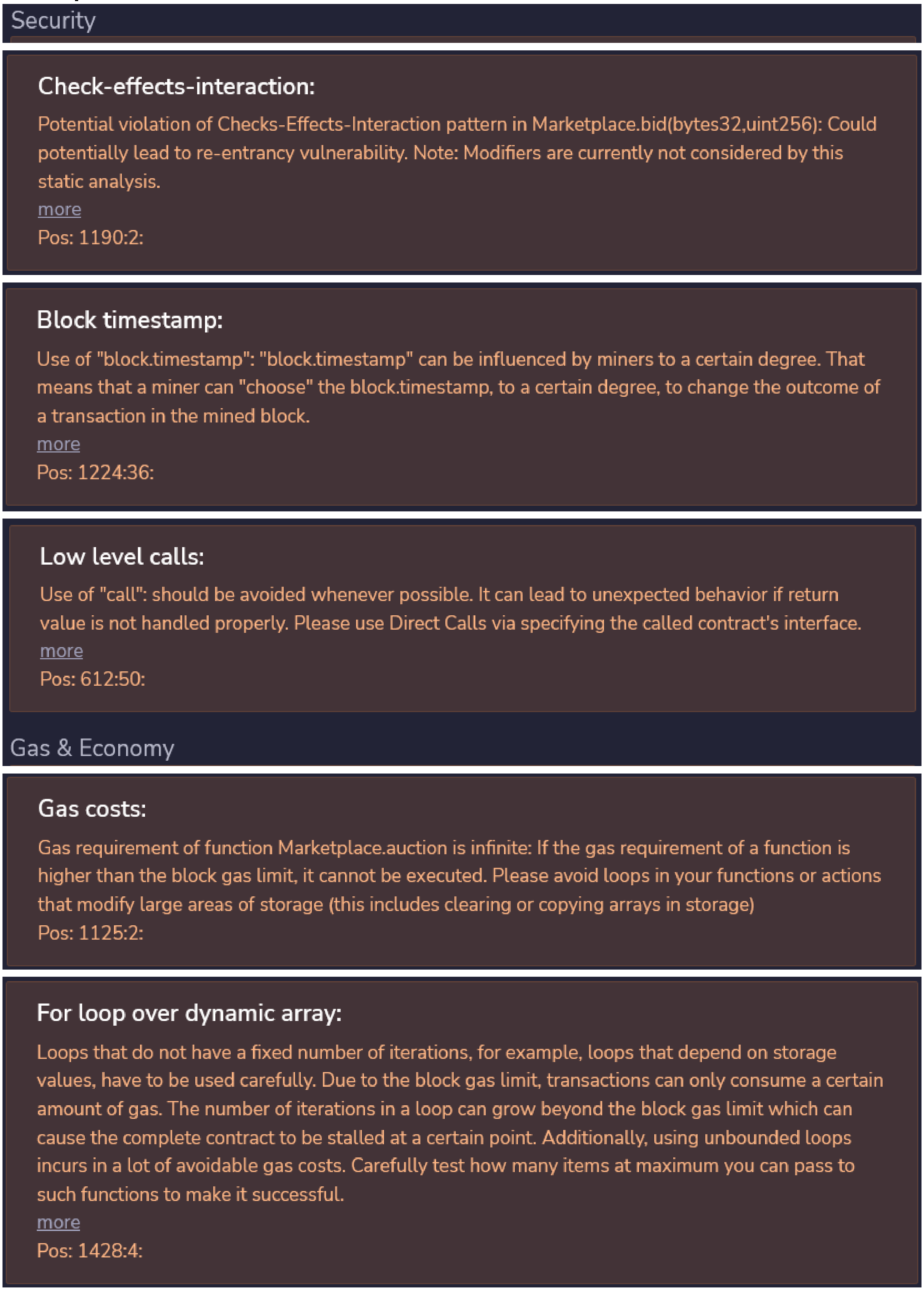
Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 2800:8:

Delete from dynamic array:

Using "delete" on an array leaves a gap. The length of the array remains the same. If you want to remove the empty position you need to shift items manually and update the "length" property.
more
Pos: 2580:16:

Marketplace.sol



Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Marketplace.bid(bytes32,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 1190:2:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.

more

Pos: 1224:36:

Low level calls:

Use of "call": should be avoided whenever possible. It can lead to unexpected behavior if return value is not handled properly. Please use Direct Calls via specifying the called contract's interface.

more

Pos: 612:50:

Gas & Economy

Gas costs:

Gas requirement of function Marketplace.auction is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

Pos: 1125:2:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.

more

Pos: 1428:4:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.
more
Pos: 1932:6:

Miscellaneous

Constant/View/Pure functions:

Marketplace.getFeeCollectors() : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1580:2:

Similar variable names:

Marketplace.cancel(bytes32) : Variables have very similar names "_items" and "item". Note: Modifiers are currently not considered by this static analysis.
Pos: 1163:4:

Similar variable names:

Marketplace.executeJob() : Variables have very similar names "_items" and "item". Note: Modifiers are currently not considered by this static analysis.
Pos: 1727:25:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 1196:4:

Delete from dynamic array:

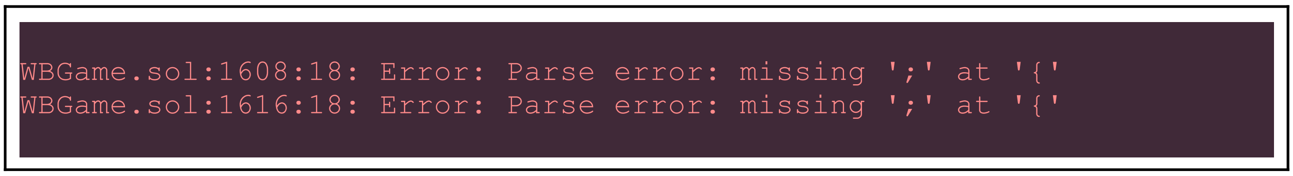
Using "delete" on an array leaves a gap. The length of the array remains the same. If you want to remove the empty position you need to shift items manually and update the "length" property.
more
Pos: 1733:12:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 186:19:

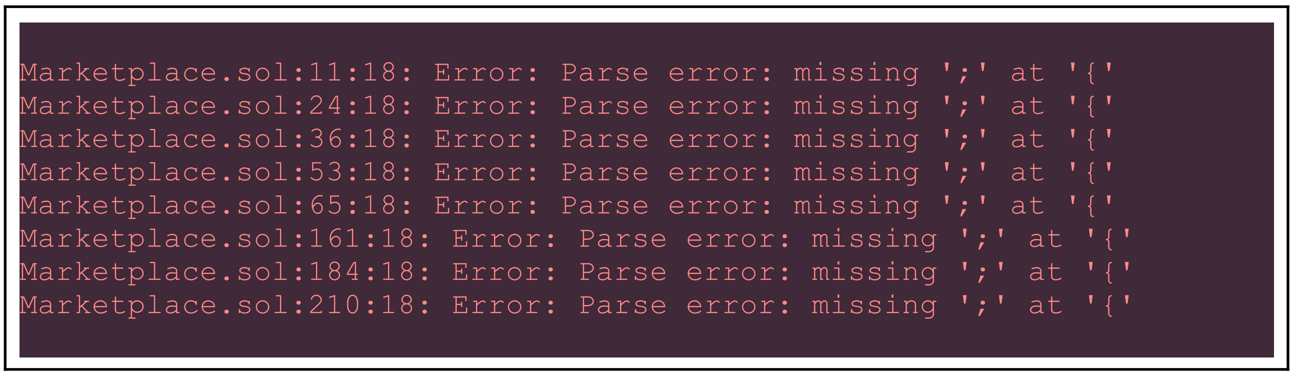
Solhint Linter

WBGame.sol



```
WBGame.sol:1608:18: Error: Parse error: missing ';' at '{'
WBGame.sol:1616:18: Error: Parse error: missing ';' at '{'
```

Marketplace.sol



```
Marketplace.sol:11:18: Error: Parse error: missing ';' at '{'
Marketplace.sol:24:18: Error: Parse error: missing ';' at '{'
Marketplace.sol:36:18: Error: Parse error: missing ';' at '{'
Marketplace.sol:53:18: Error: Parse error: missing ';' at '{'
Marketplace.sol:65:18: Error: Parse error: missing ';' at '{'
Marketplace.sol:161:18: Error: Parse error: missing ';' at '{'
Marketplace.sol:184:18: Error: Parse error: missing ';' at '{'
Marketplace.sol:210:18: Error: Parse error: missing ';' at '{'
```

Overall Software analysis result:

These software reported many false positive results and some are informational issues. So, those issues can be safely ignored.

Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice as of the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against the team on the basis of what it says or doesn't say, or how team produced it, and it is important for you to conduct your own independent investigations before making any decisions. The team will go into more detail on this in the disclaimer below – please make sure to read it in full.

By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and OnlyUp Capital and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (OnlyUp Capital) owe no duty of care towards you or any other person, nor does OnlyUp Capital make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and OnlyUp Capital hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, OnlyUp Capital hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against OnlyUp Capital, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report. The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.