Security Audit of Market

Conclusion

In the final contract were not found:

- Backdoors for investor funds withdrawal by anyone.
- Bugs allowing to steal money from the contract.
- Other security problems.

Obvious errors or backdoors were not found in the contract.
The client was acknowledged about all secutiry notes below.

SECURED 100% SECURED

Scope

```
1ca77020962c6c31d51b93d849ef0af8 Архив.zip
(base) vladimirsmelov@Vladimirs-MacBook-Air Архив %
tree contracts
contracts
├── CollectionERC1155.sol
├── CollectionERC721.sol
├── Market.sol
├── interfaces
│   ├── ICollectionERC1155.sol
│   └── ICollectionERC721.sol
├── lib
│   ├── LibConvert.sol
│   ├── LibListing.sol
│   ├── LibSig.sol
│   ├── LibToken.sol
│   └── SafeERC20.sol
└── test
└── MintableToken.sol
```

Methodology

1. Blind audit. Try to understand the structure of the code.
2. Find info in internet.
3. Ask quiestions to developers.
4. Draw the scheme of cross-contracts interactions.
5. Write user-stories, usage cases.
6. Run static analyzers

Find problems with:

- backdoors
- bugs
- math
- potential leaking of funds
- potential locking of the contract
- validate arguments and events
- others

Result

Critical

1. Unguaranteed claiming for users.

At:

- contracts/CrowdsaleVesting.sol:62
  there is a transfer of DDAO tokens, but there is no guarantee for user, that these DDAO token will be really on the balance of the contract. So user may no be sure that he will really receive DDAO tokens.

Recommendation

Add a method to lock relevant amount of DDAO tokens on the contract address.

2. Owner may withdraw all DDAO tokens.

At:

- contracts/CrowdsaleVesting.sol:128

at any point of time owner may withdraw all DDAO balance, so users will not be able to claim any tokens.

Recommendation

Implement some guarantee that users will be able to withdraw their DDAO tokens.

Major

1. Use safeERC20.

At:

- contracts/Market.sol:309
- contracts/Market.sol:313
- contracts/Market.sol:158
- contracts/Market.sol:160
- contracts/Market.sol:227

you ignore success-status from IERC20.

Recommendation

Use safeERC20 methods.

Warning

1. Royalty for ERC1155.

At:

- contracts/lib/LibToken.sol:54

the royalty is processed only for erc721.
It's better to add support for royalty for erc1155 or use exact `require` at

- contracts/lib/LibToken.sol:54

or use specific type check at

- contracts/Market.sol:306

Recommendation

Add royalty for erc1155 or be exact with getRoyalty type-checks.

2. Potential reentry.

At:

- contracts/Market.sol:101
- contracts/Market.sol:136
- contracts/Market.sol:161
- contracts/Market.sol:186

- contracts/Market.sol:204
- contracts/Market.sol:229
- contracts/Market.sol:242
- contracts/Market.sol:249

you change the state after external calls.
It's a bad habbit since it may lead to reentry attack.

Recommendation

Change state before external calls.
Use nonReentrant modifier.

3. Unfair cancelation of started auction.

At:

- contracts/Market.sol:227

it's not fair that auctioneer may cancel auction after some bid was placed.
Bidder already payed gas.

Recommendation

Remove it or add warning comment in the code.

4. Royalty for ERC1155.

At:

- contracts/lib/LibToken.sol:54

the royalty is processed only for erc721.
It's better to add support for royalty for erc1155 or use exact `require` at

- contracts/lib/LibToken.sol:54

or use specific type check at

- contracts/Market.sol:306

Recommendation

Add royalty for erc1155 or be exact with getRoyalty type-checks.

Comment

1. Use openzeppelin imports instead of implementing contracts and libs by yourself.

It seems like several contracts code were copy-pasted from Openzeppelin github.
e.g.

- SafeERC20.sol
- LibConvert.sol
  It's better to use imports from well-known well-tested repository. It decrease the number of files in the repo and decrease the chance of typo mistake. Or if some modification is neccesary, add reference link in a comment to the original.

Recommendation

Use imports or add references.

2. LIBCONVERT IS NOT CRYSTAL CEARL.

The code in

- LibConvert.sol
  is not crystal clear. Can your provide more comments and references about how does it work?

Recommendation

Add more info.

3. FREEZE SOLC VERSION.

Everywhere you use
pragma solidity ^0.8.9;
it better to replace it with exact
pragma solidity 0.8.9;
The main reason for it, is that you want empty external auditor company and everyone wants to mark it as an issue (but basically it's not). Look at the audits of CERTIK and SolidProof, they will mark it as a big red cross :-(

Recommendation

Use
pragma solidity 0.8.9;

4. ADD ZERO-CHECK.

At:

- contracts/Market.sol:64

- contracts/Market.sol:65

you don’t have check for zero-address.
This is kind of security standard.
See **https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation**

(https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation)

Recommendation

Use
require(value != address(0), “zero address”);

5. Pack struct efficiently.

At:

- contracts/lib/LibListing.sol:39

it’s better to place address field the last, to auto-combine it together with two bool, and use much less storage space.
See - **https://medium.com/@novablitz/storing-structs-is-costing-you-gas-774da988895e**

(https://medium.com/@novablitz/storing-structs-is-costing-you-gas-774da988895e)

Recommendation

Re-struct.

6. Rename royalty to royaltyNumerator.

In fact royalty is calculated on 10000 denominator, so it’s better to use royaltyNumerator as a var name to avoid misusing.

Recommendation

Rename.

7. Exclude seller from createListing argument.

At:

- contracts/Market.sol:93

you can exclude this field from the arg struct.

Recommendation

Optimize arg struct.

8. Add limit on listing duration.

At:

- contracts/Market.sol:94

you can add check for listing duration to avoid misusing.

Recommendation

Add
require(listing.end - listing.start > '365 days', "too long");
instead of
require(listing.end > listing.start, "Market: end later than start");

9. ADD LIMIT ON LISTING DURATION.

At:

- contracts/Market.sol:94

you can add check for listing duration to avoid misusing.

Recommendation

Add
require(listing.end - listing.start > '365 days', "too long");
instead of
require(listing.end > listing.start, "Market: end later than start");

10. UNCLEAR 0X1901.

At:

- contracts/Market.sol:120
- contracts/CollectionERC721.sol:96
- contracts/CollectionERC1155.sol:86
  you use 0x1901, but it's not clear what is it.

Recommendation

Add more info.

11. USE `EXTERNAL` MODIFIER.

At:

- contracts/CollectionERC1155.sol:56
- contracts/CollectionERC1155.sol:95
- contracts/CollectionERC721.sol:147
- contracts/CollectionERC721.sol:160

- contracts/CollectionERC1155.sol:130
- contracts/CollectionERC1155.sol:150

you can use `external` modifier to save gas on calls, since in this case EVM will read arguments from calldata directly.

Recommendation

Use `external` modifier, save gas.

12. Define chainId as immutable variable.

At:

- contracts/CollectionERC721.sol:106
- contracts/CollectionERC1155.sol:96

you copy chainId to the stack every time.
It may save some gas to initialize chainId in constructor as an `immutable` variable, then it will be included to the opcode.
But i'm not sure, some tests are required.

Recommendation

Use `immutable` chainId, save gas.

13. Name internal methods starting with `_` .

At:

- contracts/lib/LibToken.sol:61

Recommendation

Rename to `_transfer` .

14. Use safe methods for ERC721.

At:

- contracts/lib/LibToken.sol:64
- contracts/lib/LibToken.sol:88

it's more advanced to use safe method.

Recommendation

Use `IERC721.safeTransferFrom` .

15. Incorrect event names.

At:

- contracts/Market.sol:145

Payment and token were not really claimed.

Recommendation

Use others names.

16. Set token properties before `_mint`.

At:

- contracts/CollectionERC1155.sol:124

you first do mint and then set properties.
However there is a onERC1155Received inside `_mint`, so if receiver will try to process the token properties in some way this will fail.

Recommendation

Set token properties before `_mint`.

17. Exact check for listingType.

At:

- contracts/Market.sol:147

it's better to do exact check, what listingType do you expect.

Recommendation

Use exact check for listingType.

18. Use standard implementation for `isApprovedForAll`.

At:

- contracts/CollectionERC721.sol:167

instead of overriding the method and using more gas you can set approveForAll for the operator=market and use standard method implementation.

Recommendation

Use standard implementation for `isApprovedForAll`.

19. Emit event on new contracts created.

At:

- contracts/Market.sol:242
- contracts/Market.sol:249

it makes sense to emit event, otherwise it will be difficult to analyze the history.

Recommendation

Emit events.

20. THE LASTBID SET IS NOT NEED.

At:

- contracts/Market.sol:102

This is not really need, since you check it here

- contracts/Market.sol:150

Recommendation

Remove the set.

21. INCREASE THE BID FOR THE SAME BIDDER.

At:

- contracts/Market.sol:158

if someone wants to increase his own bid from 1000 to 1200, there is no need to transfer 1000 back to bidder and then 1200 to the contract.
You can just send 200 more tokens to the contract.

Recommendation

Add

```
if (details.lastBidder == msg.sender) {
    currency.transfer(details.lastBidder, amount - details.lastBid);
}
```

or add another `increaseBid` method for such action.

22. EMIT EVENT ON ROYALTY AND PAYMENT PAID.

At:

- contracts/Market.sol:309
- contracts/Market.sol:313

it’s better to emit event for royalty and payment paid for easy data analysis.

Recommendation

Emit event.

Slither static-analyzer log

‘npx hardhat compile --force’ running
Downloading compiler 0.8.9
Compiling 51 files with 0.8.9
Generating typings for: 52 artifacts in dir: typechain for target: ethers-v5
Successfully generated 77 typings!
Compilation finished successfully
Solidity 0.8.9 is not fully supported yet. You can still use Hardhat, but some features, like stack traces, might not work correctly.
Learn more at **https://hardhat.org/reference/solidity-support** (https://hardhat.org/reference/solidity-support)"
[91m
OwnableUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#77) shadows:
- ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
PausableUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#96) shadows:
- ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)
ERC1155Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#458) shadows:
- ERC165Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#35)
- ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#30)

```
ERC1155BurnableUpgradeable.__gap
(node_modules/@openzeppelin/contracts-
upgradeable/token/ERC1155/extensions/ERC1155Burnable
Upgradeable.sol#48) shadows:
- ERC1155Upgradeable.__gap
(node_modules/@openzeppelin/contracts-
upgradeable/token/ERC1155/ERC1155Upgradeable.sol#45
8)
- ERC165Upgradeable.__gap
(node_modules/@openzeppelin/contracts-
upgradeable/utils/introspection/ERC165Upgradeable.sol#3
5)
- ContextUpgradeable.__gap
(node_modules/@openzeppelin/contracts-
upgradeable/utils/ContextUpgradeable.sol#30)
ERC1155PausableUpgradeable.__gap
(node_modules/@openzeppelin/contracts-
upgradeable/token/ERC1155/extensions/ERC1155Pausable
Upgradeable.sol#47) shadows:
- PausableUpgradeable.__gap
(node_modules/@openzeppelin/contracts-
upgradeable/security/PausableUpgradeable.sol#96)
- ERC1155Upgradeable.__gap
(node_modules/@openzeppelin/contracts-
upgradeable/token/ERC1155/ERC1155Upgradeable.sol#45
8)
- ERC165Upgradeable.__gap
(node_modules/@openzeppelin/contracts-
upgradeable/utils/introspection/ERC165Upgradeable.sol#3
5)
- ContextUpgradeable.__gap
(node_modules/@openzeppelin/contracts-
upgradeable/utils/ContextUpgradeable.sol#30)
ERC721Upgradeable.__gap
(node_modules/@openzeppelin/contracts-
upgradeable/token/ERC721/ERC721Upgradeable.sol#418)
shadows:
- ERC165Upgradeable.__gap
(node_modules/@openzeppelin/contracts-
```

upgradeable/utils/introspection/ERC165Upgradeable.sol#3
5)
- ContextUpgradeable.__gap
(node_modules/@openzeppelin/contracts-
upgradeable/utils/ContextUpgradeable.sol#30)
ERC721BurnableUpgradeable.__gap
(node_modules/@openzeppelin/contracts-
upgradeable/token/ERC721/extensions/ERC721BurnableUp
gradeable.sol#34) shadows:
- ERC721Upgradeable.__gap
(node_modules/@openzeppelin/contracts-
upgradeable/token/ERC721/ERC721Upgradeable.sol#418)
- ERC165Upgradeable.__gap
(node_modules/@openzeppelin/contracts-
upgradeable/utils/introspection/ERC165Upgradeable.sol#3
5)
- ContextUpgradeable.__gap
(node_modules/@openzeppelin/contracts-
upgradeable/utils/ContextUpgradeable.sol#30)
ERC721PausableUpgradeable.__gap
(node_modules/@openzeppelin/contracts-
upgradeable/token/ERC721/extensions/ERC721PausableUp
gradeable.sol#42) shadows:
- PausableUpgradeable.__gap
(node_modules/@openzeppelin/contracts-
upgradeable/security/PausableUpgradeable.sol#96)
- ERC721Upgradeable.__gap
(node_modules/@openzeppelin/contracts-
upgradeable/token/ERC721/ERC721Upgradeable.sol#418)
- ERC165Upgradeable.__gap
(node_modules/@openzeppelin/contracts-
upgradeable/utils/introspection/ERC165Upgradeable.sol#3
5)
- ContextUpgradeable.__gap
(node_modules/@openzeppelin/contracts-
upgradeable/utils/ContextUpgradeable.sol#30)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing

[(https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing)](https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing)[0m
[91m
Market.bid(LibListing.Listing,uint256)
(contracts/Market.sol#107-166) ignores return value by
currency.transfer(details.lastBidder,details.lastBid)
(contracts/Market.sol#158)
Market.bid(LibListing.Listing,uint256)
(contracts/Market.sol#107-166) ignores return value by
currency.transferFrom(msg.sender,address(this),amount)
(contracts/Market.sol#160)
Market.cancel(LibListing.Listing)
(contracts/Market.sol#215-232) ignores return value by
currency.transfer(details.lastBidder,details.lastBid)
(contracts/Market.sol#227)
Market._transferPayment(address,address,uint256,LibToken
.Token) (contracts/Market.sol#298-314) ignores return
value by currency.transferFrom(from,creator,royaltyAmount)
(contracts/Market.sol#309)
Market._transferPayment(address,address,uint256,LibToken
.Token) (contracts/Market.sol#298-314) ignores return
value by currency.transferFrom(from,to,amount -
royaltyAmount) (contracts/Market.sol#313)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
[(https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer)](https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer)[0m
[93m
Reentrancy in Market.bid(LibListing.Listing,uint256)
(contracts/Market.sol#107-166):
External calls:
-
_transferPayment(msg.sender,listing.seller,listing.minBid,list
ing.token) (contracts/Market.sol#136-141)
- currency.transferFrom(from,creator,royaltyAmount)
(contracts/Market.sol#309)
- currency.transferFrom(from,to,amount - royaltyAmount)
(contracts/Market.sol#313)
- listing.token.transfer(msg.sender)
(contracts/Market.sol#142)

State variables written after the call(s):
- details.state = LibListing.ListingState.Executed
(contracts/Market.sol#143)
Reentrancy in Market.bid(LibListing.Listing,uint256)
(contracts/Market.sol#107-166):
External calls:
- currency.transfer(details.lastBidder,details.lastBid)
(contracts/Market.sol#158)
- currency.transferFrom(msg.sender,address(this),amount)
(contracts/Market.sol#160)
State variables written after the call(s):
- details.lastBidder = msg.sender
(contracts/Market.sol#161)
- details.lastBid = amount (contracts/Market.sol#162)
Reentrancy in Market.cancel(LibListing.Listing)
(contracts/Market.sol#215-232):
External calls:
- listing.token.transfer(listing.seller)
(contracts/Market.sol#224)
- currency.transfer(details.lastBidder,details.lastBid)
(contracts/Market.sol#227)
State variables written after the call(s):
- details.state = LibListing.ListingState.Cancelled
(contracts/Market.sol#229)
Reentrancy in Market.claimPayment(LibListing.Listing)
(contracts/Market.sol#191-213):
External calls:
-
_transferPayment(address(this),listing.seller,details.lastBid,li
sting.token) (contracts/Market.sol#204-209)
- currency.transferFrom(from,creator,royaltyAmount)
(contracts/Market.sol#309)
- currency.transferFrom(from,to,amount - royaltyAmount)
(contracts/Market.sol#313)
State variables written after the call(s):
- details.paymentClaimed = true (contracts/Market.sol#210)
Reentrancy in Market.claimToken(LibListing.Listing)
(contracts/Market.sol#168-189):
External calls:

- listing.token.transfer(receiver) (contracts/Market.sol#185)
State variables written after the call(s):
- details.tokenClaimed = true (contracts/Market.sol#186)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1 (https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1)[0m
[93m
ERC1155Upgradeable._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).response (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#439) is a local variable never initialized
ERC1155Upgradeable._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).reason (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#421) is a local variable never initialized
ERC1155Upgradeable._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).reason (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#444) is a local variable never initialized
ERC1155Upgradeable._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).response (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#417) is a local variable never initialized
Market._transferPayment(address,address,uint256,LibToken.Token).royaltyAmount (contracts/Market.sol#304) is a local variable never initialized
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables (https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables)[0m
[93m
ERC1155Upgradeable._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes) (node_modules/@openzeppelin/contracts-

upgradeable/token/ERC1155/ERC1155Upgradeable.sol#408-427) ignores return value by IERC1155ReceiverUpgradeable(to).onERC1155Received(operator,from,id,amount,data) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#417-425)
ERC1155Upgradeable._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#429-450) ignores return value by IERC1155ReceiverUpgradeable(to).onERC1155BatchReceived(operator,from,ids,amounts,data) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#438-448)
ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#376-397) ignores return value by IERC721ReceiverUpgradeable(to).onERC721Received(_msgSender(),from,tokenId,_data) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#383-393)
AccessControlEnumerable.grantRole(bytes32,address) (node_modules/@openzeppelin/contracts/access/AccessControlEnumerable.sol#51-54) ignores return value by _roleMembers[role].add(account) (node_modules/@openzeppelin/contracts/access/AccessControlEnumerable.sol#53)
AccessControlEnumerable.revokeRole(bytes32,address) (node_modules/@openzeppelin/contracts/access/AccessControlEnumerable.sol#59-62) ignores return value by *roleMembers[role].remove(account) (node_modules/@openzeppelin/contracts/access/AccessControlEnumerable.sol#61)*

*AccessControlEnumerable.renounceRole(bytes32,address)
(node_modules/@openzeppelin/contracts/access/AccessC
ontrolEnumerable.sol#67-70) ignores return value by
roleMembers[role].remove(account)
(node_modules/@openzeppelin/contracts/access/AccessC
ontrolEnumerable.sol#69)
AccessControlEnumerable.setupRole(bytes32,address)
(node_modules/@openzeppelin/contracts/access/AccessC
ontrolEnumerable.sol#75-78) ignores return value by
roleMembers[role].add(account)
(node_modules/@openzeppelin/contracts/access/AccessC
ontrolEnumerable.sol#77)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return (https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return)[0m
[92m
ERC20PresetMinterPauser.constructor(string,string).name
(node_modules/@openzeppelin/contracts/token/ERC20/pr
esets/ERC20PresetMinterPauser.sol#35) shadows:
- **ERC20.name** (http://ERC20.name)()
(node_modules/@openzeppelin/contracts/token/ERC20/ER
C20.sol#61-63) (function)
- **IERC20Metadata.name** (http://IERC20Metadata.name)()
(node_modules/@openzeppelin/contracts/token/ERC20/ex
tensions/IERC20Metadata.sol#16) (function)
ERC20PresetMinterPauser.constructor(string,string).symb
ol
(node_modules/@openzeppelin/contracts/token/ERC20/pr
esets/ERC20PresetMinterPauser.sol#35) shadows:
- ERC20.symbol()
(node_modules/@openzeppelin/contracts/token/ERC20/ER
C20.sol#69-71) (function)
- IERC20Metadata.symbol()
(node_modules/@openzeppelin/contracts/token/ERC20/ex
tensions/IERC20Metadata.sol#21) (function)
CollectionERC1155.isApprovedForAll(address,address).ow
ner (contracts/CollectionERC1155.sol#149) shadows:
- OwnableUpgradeable.owner()*

*(node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#40-42) (function)
CollectionERC721.initialize(string,string,bool,uint256).name (contracts/CollectionERC721.sol#44) shadows:
- **ERC721Upgradeable.name**()
(node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#85-87) (function)
- **IERC721MetadataUpgradeable.name**()
(node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/extensions/IERC721MetadataUpgradeable.sol#15) (function)
CollectionERC721.initialize(string,string,bool,uint256).symbol (contracts/CollectionERC721.sol#45) shadows:
- ERC721Upgradeable.symbol()
(node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#92-94) (function)
- IERC721MetadataUpgradeable.symbol()
(node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/extensions/IERC721MetadataUpgradeable.sol#20) (function)
CollectionERC721.isApprovedForAll(address,address).owner (contracts/CollectionERC721.sol#160) shadows:
- OwnableUpgradeable.owner()
(node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#40-42) (function)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing[0m
[92m
Market.initialize(IERC20,address,address).implementationERC721 (contracts/Market.sol#64) lacks a zero-check on :
- implementationERC721 = implementationERC721
(contracts/Market.sol#72)*

Market.initialize(IERC20,address,address).implementationERC1155 (contracts/Market.sol#65) lacks a zero-check on :
- implementationERC1155 = implementationERC1155
(contracts/Market.sol#83)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation (https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation)[0m
[92m
Variable
'ERC1155Upgradeable._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).response (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#417)' in
ERC1155Upgradeable._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#408-427) potentially used before declaration: response != IERC1155ReceiverUpgradeable.onERC1155Received.selector (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#418)
Variable
'ERC1155Upgradeable._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).reason (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#421)' in
ERC1155Upgradeable._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#408-427) potentially used before declaration: revert(string)(reason) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#422)
Variable

‘ERC1155Upgradeable._doSafeBatchTransferAcceptanceCh
eck(address,address,address,uint256[],uint256[],bytes).res
ponse (node_modules/@openzeppelin/contracts-
upgradeable/token/ERC1155/ERC1155Upgradeable.sol#43
9)’ in
ERC1155Upgradeable._doSafeBatchTransferAcceptanceCh
eck(address,address,address,uint256[],uint256[],bytes)
(node_modules/@openzeppelin/contracts-
upgradeable/token/ERC1155/ERC1155Upgradeable.sol#42
9-450) potentially used before declaration: response !=
IERC1155ReceiverUpgradeable.onERC1155BatchReceived.s
elector (node_modules/@openzeppelin/contracts-
upgradeable/token/ERC1155/ERC1155Upgradeable.sol#441
)
Variable
‘ERC1155Upgradeable._doSafeBatchTransferAcceptanceCh
eck(address,address,address,uint256[],uint256[],bytes).rea
son (node_modules/@openzeppelin/contracts-
upgradeable/token/ERC1155/ERC1155Upgradeable.sol#44
4)’ in
ERC1155Upgradeable._doSafeBatchTransferAcceptanceCh
eck(address,address,address,uint256[],uint256[],bytes)
(node_modules/@openzeppelin/contracts-
upgradeable/token/ERC1155/ERC1155Upgradeable.sol#42
9-450) potentially used before declaration: revert(string)
(reason) (node_modules/@openzeppelin/contracts-
upgradeable/token/ERC1155/ERC1155Upgradeable.sol#44
5)
Variable
‘ERC721Upgradeable._checkOnERC721Received(address,a
ddress,uint256,bytes).retval
(node_modules/@openzeppelin/contracts-
upgradeable/token/ERC721/ERC721Upgradeable.sol#383)’
in
ERC721Upgradeable._checkOnERC721Received(address,a
ddress,uint256,bytes)
(node_modules/@openzeppelin/contracts-
upgradeable/token/ERC721/ERC721Upgradeable.sol#376-
397) potentially used before declaration: retval ==

IERC721ReceiverUpgradeable.onERC721Received.selector
(node_modules/@openzeppelin/contracts-
upgradeable/token/ERC721/ERC721Upgradeable.sol#384)
Variable
'ERC721Upgradeable._checkOnERC721Received(address,a
ddress,uint256,bytes).reason
(node_modules/@openzeppelin/contracts-
upgradeable/token/ERC721/ERC721Upgradeable.sol#385)'
in
ERC721Upgradeable._checkOnERC721Received(address,a
ddress,uint256,bytes)
(node_modules/@openzeppelin/contracts-
upgradeable/token/ERC721/ERC721Upgradeable.sol#376-
397) potentially used before declaration: reason.length == 0
(node_modules/@openzeppelin/contracts-
upgradeable/token/ERC721/ERC721Upgradeable.sol#386)
Variable
'ERC721Upgradeable._checkOnERC721Received(address,a
ddress,uint256,bytes).reason
(node_modules/@openzeppelin/contracts-
upgradeable/token/ERC721/ERC721Upgradeable.sol#385)'
in
ERC721Upgradeable._checkOnERC721Received(address,a
ddress,uint256,bytes)
(node_modules/@openzeppelin/contracts-
upgradeable/token/ERC721/ERC721Upgradeable.sol#376-
397) potentially used before declaration:
revert(uint256,uint256)(32 + reason,mload(uint256)
(reason)) (node_modules/@openzeppelin/contracts-
upgradeable/token/ERC721/ERC721Upgradeable.sol#390)
Variable 'ECDSA.tryRecover(bytes32,bytes).r
(node_modules/@openzeppelin/contracts/utils/cryptograph
y/ECDSA.sol#59)' in ECDSA.tryRecover(bytes32,bytes)
(node_modules/@openzeppelin/contracts/utils/cryptograph
y/ECDSA.sol#54-83) potentially used before declaration: r =
mload(uint256)(signature + 0x20)
(node_modules/@openzeppelin/contracts/utils/cryptograph
y/ECDSA.sol#76)
Reference:

https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables (https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables)[0m
[92m
Reentrancy in CollectionERC1155._mintNext(address,address,bytes32,uint256) (contracts/CollectionERC1155.sol#116-126):
External calls:
- _mint(to,_lastTypeId,amount,) (contracts/CollectionERC1155.sol#123)
- IERC1155ReceiverUpgradeable(to).onERC1155Received(operator,from,id,amount,data) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#417-425)
State variables written after the call(s):
- _hashOf[*lastTypeId] = typeHash (contracts/CollectionERC1155.sol#124)
- creators[lastTypeId] = creator (contracts/CollectionERC1155.sol#125)
Reentrancy in Market.createListing(LibListing.Listing) (contracts/Market.sol#92-105):
External calls:
- listing.token.deposit() (contracts/Market.sol#97)
State variables written after the call(s):
- details.state = LibListing.ListingState.Created (contracts/Market.sol#101)
- details.lastBid = listing.minBid (contracts/Market.sol#102)
Reentrancy in Market.createPrivateERC1155() (contracts/Market.sol#245-250):
External calls:
- ICollectionERC1155(collection).initialize(true) (contracts/Market.sol#247)
- ICollectionERC1155(collection).transferOwnership(msg.sender) (contracts/Market.sol#248)*

State variables written after the call(s):
- contractTypes[collection] = ContractType.ERC1155
(contracts/Market.sol#249)
Reentrancy in
Market.createPrivateERC721(string,string,uint256)
(contracts/Market.sol#234-243):
External calls:
-
ICollectionERC721(collection).initialize(name,symbol,true,r
oyalty) (contracts/Market.sol#240)
-
ICollectionERC721(collection).transferOwnership(msg.send
er) (contracts/Market.sol#241)
State variables written after the call(s):
- contractTypes[collection] = ContractType.ERC721
(contracts/Market.sol#242)
Reentrancy in Market.initialize(IERC20,address,address)
(contracts/Market.sol#62-88):
External calls:
- currency.safeApprove(address(this),type()(uint256).max)
(contracts/Market.sol#70)
State variables written after the call(s):
- defaultERC721 = implementationERC721.clone()
(contracts/Market.sol#73)
- implementationERC721 = implementationERC721
(contracts/Market.sol#72)
Reentrancy in Market.initialize(IERC20,address,address)
(contracts/Market.sol#62-88):
External calls:
- currency.safeApprove(address(this),type()(uint256).max)
(contracts/Market.sol#70)
- ICollectionERC721(defaultERC721).initialize(Galaxe
NFT,GLXNFT,false,0) (contracts/Market.sol#74-79)
-
ICollectionERC721(defaultERC721).transferOwnership(own
er()) (contracts/Market.sol#80)
State variables written after the call(s):
- contractTypes[defaultERC721] = ContractType.ERC721
(contracts/Market.sol#81)

- defaultERC1155 = implementationERC1155.clone()
(contracts/Market.sol#84)
- implementationERC1155 = implementationERC1155
(contracts/Market.sol#83)
Reentrancy in Market.initialize(IERC20,address,address)
(contracts/Market.sol#62-88):
External calls:
- currency.safeApprove(address(this),type()(uint256).max)
(contracts/Market.sol#70)
- ICollectionERC721(defaultERC721).initialize(Galaxe
NFT,GLXNFT,false,0) (contracts/Market.sol#74-79)
-
ICollectionERC721(defaultERC721).transferOwnership(own
er()) (contracts/Market.sol#80)
- ICollectionERC1155(defaultERC1155).initialize(false)
(contracts/Market.sol#85)
-
ICollectionERC721(defaultERC1155).transferOwnership(ow
ner()) (contracts/Market.sol#86)
State variables written after the call(s):
- contractTypes[defaultERC1155] = ContractType.ERC1155
(contracts/Market.sol#87)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
(https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2)[0m
[92m
Reentrancy in Market.bid(LibListing.Listing,uint256)
(contracts/Market.sol#107-166):
External calls:
-
_transferPayment(msg.sender,listing.seller,listing.minBid,list
ing.token) (contracts/Market.sol#136-141)
- currency.transferFrom(from,creator,royaltyAmount)
(contracts/Market.sol#309)
- currency.transferFrom(from,to,amount - royaltyAmount)
(contracts/Market.sol#313)
- listing.token.transfer(msg.sender)
(contracts/Market.sol#142)

```
Event emitted after the call(s):
- ListingPaymentClaimed(listing)
(contracts/Market.sol#146)
- ListingTokenClaimed(listing,msg.sender)
(contracts/Market.sol#145)
Reentrancy in Market.bid(LibListing.Listing,uint256)
(contracts/Market.sol#107-166):
External calls:
- currency.transfer(details.lastBidder,details.lastBid)
(contracts/Market.sol#158)
- currency.transferFrom(msg.sender,address(this),amount)
(contracts/Market.sol#160)
Event emitted after the call(s):
- ListingBid(listing,msg.sender,amount)
(contracts/Market.sol#164)
Reentrancy in Market.cancel(LibListing.Listing)
(contracts/Market.sol#215-232):
External calls:
- listing.token.transfer(listing.seller)
(contracts/Market.sol#224)
- currency.transfer(details.lastBidder,details.lastBid)
(contracts/Market.sol#227)
Event emitted after the call(s):
- ListingCancelled(listing) (contracts/Market.sol#231)
Reentrancy in Market.claimPayment(LibListing.Listing)
(contracts/Market.sol#191-213):
External calls:
-
_transferPayment(address(this),listing.seller,details.lastBid,li
sting.token) (contracts/Market.sol#204-209)
- currency.transferFrom(from,creator,royaltyAmount)
(contracts/Market.sol#309)
- currency.transferFrom(from,to,amount - royaltyAmount)
(contracts/Market.sol#313)
Event emitted after the call(s):
- ListingPaymentClaimed(listing)
(contracts/Market.sol#212)
Reentrancy in Market.claimToken(LibListing.Listing)
(contracts/Market.sol#168-189):
```

```
External calls:
- listing.token.transfer(receiver) (contracts/Market.sol#185)
Event emitted after the call(s):
- ListingTokenClaimed(listing,msg.sender)
(contracts/Market.sol#188)
Reentrancy in Market.createListing(LibListing.Listing)
(contracts/Market.sol#92-105):
External calls:
- listing.token.deposit() (contracts/Market.sol#97)
Event emitted after the call(s):
- ListingCreated(listing) (contracts/Market.sol#104)
Reference:
```

https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3 (https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3)[0m

```
[92m
Market.createListing(LibListing.Listing)
(contracts/Market.sol#92-105) uses timestamp for
comparisons
Dangerous comparisons:
- require(bool,string)(listing.end >= block.timestamp,Market:
end is in past) (contracts/Market.sol#95)
Market.bid(LibListing.Listing,uint256)
(contracts/Market.sol#107-166) uses timestamp for
comparisons
Dangerous comparisons:
- require(bool,string)(block.timestamp >=
listing.start,Market: listing not started)
(contracts/Market.sol#129-132)
- require(bool,string)(block.timestamp < listing.end,Market:
listing finished) (contracts/Market.sol#133)
Market.claimToken(LibListing.Listing)
(contracts/Market.sol#168-189) uses timestamp for
comparisons
Dangerous comparisons:
- require(bool,string)(block.timestamp >= listing.end,Market:
listing not finished) (contracts/Market.sol#176)
Market.claimPayment(LibListing.Listing)
(contracts/Market.sol#191-213) uses timestamp for
```

```
comparisons
Dangerous comparisons:
- require(bool,string)(block.timestamp >= listing.end,Market:
listing not finished) (contracts/Market.sol#199)
Market.cancel(LibListing.Listing)
(contracts/Market.sol#215-232) uses timestamp for
comparisons
Dangerous comparisons:
- require(bool,string)(block.timestamp < listing.end,Market:
listing finished) (contracts/Market.sol#220)
Reference:
https://github.com/crytic/slither/wiki/Detector-
Documentation#block-timestamp
(https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp)[0m
[92m
ERC721Upgradeable._checkOnERC721Received(address,a
ddress,uint256,bytes)
(node_modules/@openzeppelin/contracts-
upgradeable/token/ERC721/ERC721Upgradeable.sol#376-
397) uses assembly
- INLINE ASM (node_modules/@openzeppelin/contracts-
upgradeable/token/ERC721/ERC721Upgradeable.sol#389-
391)
AddressUpgradeable.isContract(address)
(node_modules/@openzeppelin/contracts-
upgradeable/utils/AddressUpgradeable.sol#26-36) uses
assembly
- INLINE ASM (node_modules/@openzeppelin/contracts-
upgradeable/utils/AddressUpgradeable.sol#32-34)
AddressUpgradeable.verifyCallResult(bool,bytes,string)
(node_modules/@openzeppelin/contracts-
upgradeable/utils/AddressUpgradeable.sol#168-188) uses
assembly
- INLINE ASM (node_modules/@openzeppelin/contracts-
upgradeable/utils/AddressUpgradeable.sol#180-183)
Clones.clone(address)
(node_modules/@openzeppelin/contracts/proxy/Clones.sol
#24-33) uses assembly
- INLINE ASM
```

```
(node_modules/@openzeppelin/contracts/proxy/Clones.sol
#25-31)
Clones.cloneDeterministic(address,bytes32)
(node_modules/@openzeppelin/contracts/proxy/Clones.sol
#42-51) uses assembly
- INLINE ASM
(node_modules/@openzeppelin/contracts/proxy/Clones.sol
#43-49)
Clones.predictDeterministicAddress(address,bytes32,addre
ss)
(node_modules/@openzeppelin/contracts/proxy/Clones.sol
#56-71) uses assembly
- INLINE ASM
(node_modules/@openzeppelin/contracts/proxy/Clones.sol
#61-70)
ECDSA.tryRecover(bytes32,bytes)
(node_modules/@openzeppelin/contracts/utils/cryptograph
y/ECDSA.sol#54-83) uses assembly
- INLINE ASM
(node_modules/@openzeppelin/contracts/utils/cryptograph
y/ECDSA.sol#64-68)
- INLINE ASM
(node_modules/@openzeppelin/contracts/utils/cryptograph
y/ECDSA.sol#75-78)
ECDSA.tryRecover(bytes32,bytes32,bytes32)
(node_modules/@openzeppelin/contracts/utils/cryptograph
y/ECDSA.sol#112-124) uses assembly
- INLINE ASM
(node_modules/@openzeppelin/contracts/utils/cryptograph
y/ECDSA.sol#119-122)
EnumerableSet.values(EnumerableSet.AddressSet)
(node_modules/@openzeppelin/contracts/utils/structs/Enu
merableSet.sol#273-282) uses assembly
- INLINE ASM
(node_modules/@openzeppelin/contracts/utils/structs/Enu
merableSet.sol#277-279)
EnumerableSet.values(EnumerableSet.UintSet)
(node_modules/@openzeppelin/contracts/utils/structs/Enu
merableSet.sol#346-355) uses assembly
```

- INLINE ASM
(node_modules/@openzeppelin/contracts/utils/structs/Enu
merableSet.sol#350-352)
CollectionERC1155.domainHash()
(contracts/CollectionERC1155.sol#95-112) uses assembly
- INLINE ASM (contracts/CollectionERC1155.sol#98-100)
CollectionERC721.domainHash()
(contracts/CollectionERC721.sol#105-122) uses assembly
- INLINE ASM (contracts/CollectionERC721.sol#108-110)
Market.domainHash() (contracts/Market.sol#262-279) uses
assembly
- INLINE ASM (contracts/Market.sol#265-267)
Address.isContract(address)
(contracts/lib/SafeERC20.sol#28-38) uses assembly
- INLINE ASM (contracts/lib/SafeERC20.sol#34-36)
Address.verifyCallResult(bool,bytes,string)
(contracts/lib/SafeERC20.sol#199-219) uses assembly
- INLINE ASM (contracts/lib/SafeERC20.sol#211-214)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
(https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage)[0m
[92m
Different versions of Solidity is used:
- Version used: ['^0.8.0', '^0.8.9']
- ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#3)
- ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#3)
- ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#3)
- ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#3)
- ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/IERC1155ReceiverUpgradeabl
e.sol#3)
- ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/IERC1155Upgradeable.sol#3)
- ^0.8.0 (node_modules/@openzeppelin/contracts-

```
upgradeable/token/ERC1155/extensions/ERC1155Burnable
Upgradeable.sol#3)
- ^0.8.0 (node_modules/@openzeppelin/contracts-
upgradeable/token/ERC1155/extensions/ERC1155Pausable
Upgradeable.sol#3)
- ^0.8.0 (node_modules/@openzeppelin/contracts-
upgradeable/token/ERC1155/extensions/IERC1155Metadata
URIUpgradeable.sol#3)
- ^0.8.0 (node_modules/@openzeppelin/contracts-
upgradeable/token/ERC721/ERC721Upgradeable.sol#3)
- ^0.8.0 (node_modules/@openzeppelin/contracts-
upgradeable/token/ERC721/IERC721ReceiverUpgradeable.s
ol#3)
- ^0.8.0 (node_modules/@openzeppelin/contracts-
upgradeable/token/ERC721/IERC721Upgradeable.sol#3)
- ^0.8.0 (node_modules/@openzeppelin/contracts-
upgradeable/token/ERC721/extensions/ERC721BurnableUp
gradeable.sol#3)
- ^0.8.0 (node_modules/@openzeppelin/contracts-
upgradeable/token/ERC721/extensions/ERC721PausableUp
gradeable.sol#3)
- ^0.8.0 (node_modules/@openzeppelin/contracts-
upgradeable/token/ERC721/extensions/IERC721MetadataU
pgradeable.sol#3)
- ^0.8.0 (node_modules/@openzeppelin/contracts-
upgradeable/utils/AddressUpgradeable.sol#3)
- ^0.8.0 (node_modules/@openzeppelin/contracts-
upgradeable/utils/ContextUpgradeable.sol#3)
- ^0.8.0 (node_modules/@openzeppelin/contracts-
upgradeable/utils/StringsUpgradeable.sol#3)
- ^0.8.0 (node_modules/@openzeppelin/contracts-
upgradeable/utils/introspection/ERC165Upgradeable.sol#3)
- ^0.8.0 (node_modules/@openzeppelin/contracts-
upgradeable/utils/introspection/IERC165Upgradeable.sol#3
)
- ^0.8.0
(node_modules/@openzeppelin/contracts/access/AccessC
ontrol.sol#3)
- ^0.8.0
```

(node_modules/@openzeppelin/contracts/access/AccessC
ontrolEnumerable.sol#3)
- ^0.8.0
(node_modules/@openzeppelin/contracts/access/IAccessC
ontrol.sol#3)
- ^0.8.0
(node_modules/@openzeppelin/contracts/access/IAccessC
ontrolEnumerable.sol#3)
- ^0.8.0
(node_modules/@openzeppelin/contracts/proxy/Clones.sol
#3)
- ^0.8.0
(node_modules/@openzeppelin/contracts/security/Pausable
.sol#3)
- ^0.8.0
(node_modules/@openzeppelin/contracts/token/ERC1155/I
ERC1155.sol#3)
- ^0.8.0
(node_modules/@openzeppelin/contracts/token/ERC20/ER
C20.sol#3)
- ^0.8.0
(node_modules/@openzeppelin/contracts/token/ERC20/IER
C20.sol#3)
- ^0.8.0
(node_modules/@openzeppelin/contracts/token/ERC20/ext
ensions/ERC20Burnable.sol#3)
- ^0.8.0
(node_modules/@openzeppelin/contracts/token/ERC20/ext
ensions/ERC20Pausable.sol#3)
- ^0.8.0
(node_modules/@openzeppelin/contracts/token/ERC20/ext
ensions/IERC20Metadata.sol#3)
- ^0.8.0
(node_modules/@openzeppelin/contracts/token/ERC20/pre
sets/ERC20PresetMinterPauser.sol#3)
- ^0.8.0
(node_modules/@openzeppelin/contracts/token/ERC721/IE
RC721.sol#3)
- ^0.8.0

(node_modules/@openzeppelin/contracts/utils/Context.sol#3)
- ^0.8.0
(node_modules/@openzeppelin/contracts/utils/Strings.sol#3)
- ^0.8.0
(node_modules/@openzeppelin/contracts/utils/cryptography/ECDSA.sol#3)
- ^0.8.0
(node_modules/@openzeppelin/contracts/utils/introspection/ERC165.sol#3)
- ^0.8.0
(node_modules/@openzeppelin/contracts/utils/introspection/IERC165.sol#3)
- ^0.8.0
(node_modules/@openzeppelin/contracts/utils/structs/EnumerableSet.sol#3)
- ^0.8.9 (contracts/CollectionERC1155.sol#2)
- ^0.8.9 (contracts/CollectionERC721.sol#2)
- ^0.8.9 (contracts/Market.sol#2)
- ^0.8.9 (contracts/interfaces/ICollectionERC1155.sol#2)
- ^0.8.9 (contracts/interfaces/ICollectionERC721.sol#2)
- ^0.8.9 (contracts/lib/LibConvert.sol#2)
- ^0.8.9 (contracts/lib/LibListing.sol#2)
- ^0.8.9 (contracts/lib/LibSig.sol#2)
- ^0.8.9 (contracts/lib/LibToken.sol#2)
- ^0.8.0 (contracts/lib/SafeERC20.sol#3)
- ^0.8.9 (contracts/test/MintableToken.sol#2)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
(https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used)
[0m
[92m
Address.functionCall(address,bytes)
(contracts/lib/SafeERC20.sol#87-92) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256)
(contracts/lib/SafeERC20.sol#119-131) is never used and

should be removed
Address.functionStaticCall(address,bytes)
(contracts/lib/SafeERC20.sol#163-174) is never used and
should be removed
Address.functionStaticCall(address,bytes,string)
(contracts/lib/SafeERC20.sol#182-191) is never used and
should be removed
Address.sendValue(address,uint256)
(contracts/lib/SafeERC20.sol#56-67) is never used and
should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint25
6) (contracts/lib/SafeERC20.sol#298-319) is never used
and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint25
6) (contracts/lib/SafeERC20.sol#282-296) is never used
and should be removed
SafeERC20.safeTransfer(IERC20,address,uint256)
(contracts/lib/SafeERC20.sol#234-243) is never used and
should be removed
SafeERC20.safeTransferFrom(IERC20,address,address,uint2
56) (contracts/lib/SafeERC20.sol#245-255) is never used
and should be removed
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code (https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code)[0m
[92m
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#3) allows
old versions
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#3) allows old
versions
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#3) allows
old versions

Pragma version^0.8.0
(node_modules/@openzeppelin/contracts-
upgradeable/token/ERC1155/ERC1155Upgradeable.sol#3)
allows old versions
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts-
upgradeable/token/ERC1155/IERC1155ReceiverUpgradeabl
e.sol#3) allows old versions
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts-
upgradeable/token/ERC1155/IERC1155Upgradeable.sol#3)
allows old versions
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts-
upgradeable/token/ERC1155/extensions/ERC1155Burnable
Upgradeable.sol#3) allows old versions
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts-
upgradeable/token/ERC1155/extensions/ERC1155Pausable
Upgradeable.sol#3) allows old versions
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts-
upgradeable/token/ERC1155/extensions/IERC1155Metadata
URIUpgradeable.sol#3) allows old versions
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts-
upgradeable/token/ERC721/ERC721Upgradeable.sol#3)
allows old versions
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts-
upgradeable/token/ERC721/IERC721ReceiverUpgradeable.s
ol#3) allows old versions
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts-
upgradeable/token/ERC721/IERC721Upgradeable.sol#3)
allows old versions
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts-
upgradeable/token/ERC721/extensions/ERC721BurnableUp

gradeable.sol#3) allows old versions
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/extensions/ERC721PausableUpgradeable.sol#3) allows old versions
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/extensions/IERC721MetadataUpgradeable.sol#3) allows old versions
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#3) allows old versions
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#3) allows old versions
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts-upgradeable/utils/StringsUpgradeable.sol#3) allows old versions
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#3) allows old versions
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/IERC165Upgradeable.sol#3) allows old versions
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts/access/AccessControl.sol#3) allows old versions
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts/access/AccessControlEnumerable.sol#3) allows old versions
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts/access/IAccessControl.sol#3) allows old versions
Pragma version^0.8.0

(node_modules/@openzeppelin/contracts/access/IAccessC
ontrolEnumerable.sol#3) allows old versions
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts/proxy/Clones.sol
#3) allows old versions
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts/security/Pausable
.sol#3) allows old versions
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts/token/ERC1155/I
ERC1155.sol#3) allows old versions
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts/token/ERC20/ER
C20.sol#3) allows old versions
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts/token/ERC20/IER
C20.sol#3) allows old versions
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts/token/ERC20/ext
ensions/ERC20Burnable.sol#3) allows old versions
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts/token/ERC20/ext
ensions/ERC20Pausable.sol#3) allows old versions
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts/token/ERC20/ext
ensions/IERC20Metadata.sol#3) allows old versions
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts/token/ERC20/pre
sets/ERC20PresetMinterPauser.sol#3) allows old versions
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts/token/ERC721/IE
RC721.sol#3) allows old versions
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts/utils/Context.sol#
3) allows old versions
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts/utils/Strings.sol#
3) allows old versions
Pragma version^0.8.0

```
(node_modules/@openzeppelin/contracts/utils/cryptograph
y/ECDSA.sol#3) allows old versions
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts/utils/introspectio
n/ERC165.sol#3) allows old versions
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts/utils/introspectio
n/IERC165.sol#3) allows old versions
Pragma version^0.8.0
(node_modules/@openzeppelin/contracts/utils/structs/Enu
merableSet.sol#3) allows old versions
Pragma version^0.8.9 (contracts/CollectionERC1155.sol#2)
necessitates a version too recent to be trusted. Consider
deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.9 (contracts/CollectionERC721.sol#2)
necessitates a version too recent to be trusted. Consider
deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.9 (contracts/Market.sol#2)
necessitates a version too recent to be trusted. Consider
deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.9
(contracts/interfaces/ICollectionERC1155.sol#2)
necessitates a version too recent to be trusted. Consider
deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.9
(contracts/interfaces/ICollectionERC721.sol#2)
necessitates a version too recent to be trusted. Consider
deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.9 (contracts/lib/LibConvert.sol#2)
necessitates a version too recent to be trusted. Consider
deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.9 (contracts/lib/LibListing.sol#2)
necessitates a version too recent to be trusted. Consider
deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.9 (contracts/lib/LibSig.sol#2)
necessitates a version too recent to be trusted. Consider
deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.9 (contracts/lib/LibToken.sol#2)
necessitates a version too recent to be trusted. Consider
```

```
deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.0 (contracts/lib/SafeERC20.sol#3)
allows old versions
Pragma version^0.8.9 (contracts/test/MintableToken.sol#2)
necessitates a version too recent to be trusted. Consider
deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.9 is not recommended for deployment
Reference:
```

https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity (https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity)[0m

```
[92m
Low level call in
AddressUpgradeable.sendValue(address,uint256)
(node_modules/@openzeppelin/contracts-
upgradeable/utils/AddressUpgradeable.sol#54-59):
- (success) = recipient.call{value: amount}()
(node_modules/@openzeppelin/contracts-
upgradeable/utils/AddressUpgradeable.sol#57)
Low level call in
AddressUpgradeable.functionCallWithValue(address,bytes,
uint256,string) (node_modules/@openzeppelin/contracts-
upgradeable/utils/AddressUpgradeable.sol#122-133):
- (success,returndata) = target.call{value: value}(data)
(node_modules/@openzeppelin/contracts-
upgradeable/utils/AddressUpgradeable.sol#131)
Low level call in
AddressUpgradeable.functionStaticCall(address,bytes,strin
g) (node_modules/@openzeppelin/contracts-
upgradeable/utils/AddressUpgradeable.sol#151-160):
- (success,returndata) = target.staticcall(data)
(node_modules/@openzeppelin/contracts-
upgradeable/utils/AddressUpgradeable.sol#158)
Low level call in Address.sendValue(address,uint256)
(contracts/lib/SafeERC20.sol#56-67):
- (success) = recipient.call{value: amount}()
(contracts/lib/SafeERC20.sol#62)
Low level call in
Address.functionCallWithValue(address,bytes,uint256,strin
```

g) (contracts/lib/SafeERC20.sol#139-155):
- (success,returndata) = target.call{value: value}(data)
(contracts/lib/SafeERC20.sol#151-153)
Low level call in
Address.functionStaticCall(address,bytes,string)
(contracts/lib/SafeERC20.sol#182-191):
- (success,returndata) = target.staticcall(data)
(contracts/lib/SafeERC20.sol#189)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls (https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls)[0m
[92m
CollectionERC1155 (contracts/CollectionERC1155.sol#10-168) should inherit from ICollectionERC1155
(contracts/interfaces/ICollectionERC1155.sol#6-12)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#missing-inheritance (https://github.com/crytic/slither/wiki/Detector-Documentation#missing-inheritance)[0m
[92m
Function OwnableUpgradeable.__Ownable_init()
(node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#28-31) is
not in mixedCase
Function
OwnableUpgradeable.__Ownable_init_unchained()
(node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#33-35) is
not in mixedCase
Variable OwnableUpgradeable.__gap
(node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#77) is not in
mixedCase
Function PausableUpgradeable.__Pausable_init()
(node_modules/@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#33-36) is
not in mixedCase
Function

```
PausableUpgradeable.__Pausable_init_unchained()
(node_modules/@openzeppelin/contracts-
upgradeable/security/PausableUpgradeable.sol#38-40) is
not in mixedCase
Variable PausableUpgradeable.__gap
(node_modules/@openzeppelin/contracts-
upgradeable/security/PausableUpgradeable.sol#96) is not
in mixedCase
Function ERC1155Upgradeable.__ERC1155_init(string)
(node_modules/@openzeppelin/contracts-
upgradeable/token/ERC1155/ERC1155Upgradeable.sol#35-
39) is not in mixedCase
Function
ERC1155Upgradeable.__ERC1155_init_unchained(string)
(node_modules/@openzeppelin/contracts-
upgradeable/token/ERC1155/ERC1155Upgradeable.sol#41-
43) is not in mixedCase
Variable ERC1155Upgradeable.__gap
(node_modules/@openzeppelin/contracts-
upgradeable/token/ERC1155/ERC1155Upgradeable.sol#45
8) is not in mixedCase
Function
ERC1155BurnableUpgradeable.__ERC1155Burnable_init()
(node_modules/@openzeppelin/contracts-
upgradeable/token/ERC1155/extensions/ERC1155Burnable
Upgradeable.sol#15-19) is not in mixedCase
Function
ERC1155BurnableUpgradeable.__ERC1155Burnable_init_un
chained() (node_modules/@openzeppelin/contracts-
upgradeable/token/ERC1155/extensions/ERC1155Burnable
Upgradeable.sol#21-22) is not in mixedCase
Variable ERC1155BurnableUpgradeable.__gap
(node_modules/@openzeppelin/contracts-
upgradeable/token/ERC1155/extensions/ERC1155Burnable
Upgradeable.sol#48) is not in mixedCase
Function
ERC1155PausableUpgradeable.__ERC1155Pausable_init()
(node_modules/@openzeppelin/contracts-
upgradeable/token/ERC1155/extensions/ERC1155Pausable
Upgradeable.sol#19-24) is not in mixedCase
Function
ERC1155PausableUpgradeable.__ERC1155Pausable_init_u
nchained() (node_modules/@openzeppelin/contracts-
upgradeable/token/ERC1155/extensions/ERC1155Pausable
Upgradeable.sol#26-27) is not in mixedCase
Variable ERC1155PausableUpgradeable.__gap
(node_modules/@openzeppelin/contracts-
upgradeable/token/ERC1155/extensions/ERC1155Pausable
Upgradeable.sol#47) is not in mixedCase
Function ERC721Upgradeable.__ERC721_init(string,string)
(node_modules/@openzeppelin/contracts-
upgradeable/token/ERC721/ERC721Upgradeable.sol#44-
48) is not in mixedCase
Function
ERC721Upgradeable.__ERC721_init_unchained(string,strin
g) (node_modules/@openzeppelin/contracts-
upgradeable/token/ERC721/ERC721Upgradeable.sol#50-
53) is not in mixedCase
Parameter
ERC721Upgradeable.safeTransferFrom(address,address,uint
256,bytes)._data
(node_modules/@openzeppelin/contracts-
upgradeable/token/ERC721/ERC721Upgradeable.sol#188)
is not in mixedCase
Variable ERC721Upgradeable.__gap
(node_modules/@openzeppelin/contracts-
upgradeable/token/ERC721/ERC721Upgradeable.sol#418)
is not in mixedCase
Function
ERC721BurnableUpgradeable.__ERC721Burnable_init()
(node_modules/@openzeppelin/contracts-
upgradeable/token/ERC721/extensions/ERC721BurnableUp
gradeable.sol#14-18) is not in mixedCase
Function
ERC721BurnableUpgradeable.__ERC721Burnable_init_unc
hained() (node_modules/@openzeppelin/contracts-
upgradeable/token/ERC721/extensions/ERC721BurnableUp
gradeable.sol#20-21) is not in mixedCase
```

```
Variable ERC721BurnableUpgradeable.__gap
(node_modules/@openzeppelin/contracts-
upgradeable/token/ERC721/extensions/ERC721BurnableUp
gradeable.sol#34) is not in mixedCase
Function
ERC721PausableUpgradeable.__ERC721Pausable_init()
(node_modules/@openzeppelin/contracts-
upgradeable/token/ERC721/extensions/ERC721PausableUp
gradeable.sol#17-22) is not in mixedCase
Function
ERC721PausableUpgradeable.__ERC721Pausable_init_unc
hained() (node_modules/@openzeppelin/contracts-
upgradeable/token/ERC721/extensions/ERC721PausableUp
gradeable.sol#24-25) is not in mixedCase
Variable ERC721PausableUpgradeable.__gap
(node_modules/@openzeppelin/contracts-
upgradeable/token/ERC721/extensions/ERC721PausableUp
gradeable.sol#42) is not in mixedCase
Function ContextUpgradeable.__Context_init()
(node_modules/@openzeppelin/contracts-
upgradeable/utils/ContextUpgradeable.sol#17-19) is not in
mixedCase
Function ContextUpgradeable.__Context_init_unchained()
(node_modules/@openzeppelin/contracts-
upgradeable/utils/ContextUpgradeable.sol#21-22) is not in
mixedCase
Variable ContextUpgradeable.__gap
(node_modules/@openzeppelin/contracts-
upgradeable/utils/ContextUpgradeable.sol#30) is not in
mixedCase
Function ERC165Upgradeable.__ERC165_init()
(node_modules/@openzeppelin/contracts-
upgradeable/utils/introspection/ERC165Upgradeable.sol#2
3-25) is not in mixedCase
Function ERC165Upgradeable.__ERC165_init_unchained()
(node_modules/@openzeppelin/contracts-
upgradeable/utils/introspection/ERC165Upgradeable.sol#2
7-28) is not in mixedCase
Variable ERC165Upgradeable.__gap
```

(node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#35) is not in mixedCase
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions (https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions)[0m
[92m
Clones.clone(address)
(node_modules/@openzeppelin/contracts/proxy/Clones.sol#24-33) uses literals with too many digits:
- mstore(uint256,uint256)
(ptr_clone_asm_0,0x3d602d80600a3d3981f3363d3d373d3d3d363d73000000000000000000000000)
(node_modules/@openzeppelin/contracts/proxy/Clones.sol#27)
Clones.clone(address)
(node_modules/@openzeppelin/contracts/proxy/Clones.sol#24-33) uses literals with too many digits:
- mstore(uint256,uint256)(ptr_clone_asm_0 + 0x28,0x5af43d82803e903d91602b57fd5bf30000000000000000000000000000000000)
(node_modules/@openzeppelin/contracts/proxy/Clones.sol#29)
Clones.cloneDeterministic(address,bytes32)
(node_modules/@openzeppelin/contracts/proxy/Clones.sol#42-51) uses literals with too many digits:
- mstore(uint256,uint256)
(ptr_cloneDeterministic_asm_0,0x3d602d80600a3d3981f3363d3d373d3d3d363d73000000000000000000000000)
(node_modules/@openzeppelin/contracts/proxy/Clones.sol#45)
Clones.cloneDeterministic(address,bytes32)
(node_modules/@openzeppelin/contracts/proxy/Clones.sol#42-51) uses literals with too many digits:
- mstore(uint256,uint256)(ptr_cloneDeterministic_asm_0 + 0x28,0x5af43d82803e903d91602b57fd5bf3000000000

0000000000000000000000000)
(node_modules/@openzeppelin/contracts/proxy/Clones.sol
#47)
Clones.predictDeterministicAddress(address,bytes32,addre
ss)
(node_modules/@openzeppelin/contracts/proxy/Clones.sol
#56-71) uses literals with too many digits:
- mstore(uint256,uint256)
(ptr_predictDeterministicAddress_asm_0,0x3d602d80600
a3d3981f3363d3d373d3d3d363d7300000000000000000
000000000)
(node_modules/@openzeppelin/contracts/proxy/Clones.sol
#63)
Clones.predictDeterministicAddress(address,bytes32,addre
ss)
(node_modules/@openzeppelin/contracts/proxy/Clones.sol
#56-71) uses literals with too many digits:
- mstore(uint256,uint256)
(ptr_predictDeterministicAddress_asm_0 +
0x28,0x5af43d82803e903d91602b57fd5bf3ff00000000
0000000000000000000000000)
(node_modules/@openzeppelin/contracts/proxy/Clones.sol
#65)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
(https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits)[0m
[92m
OwnableUpgradeable.__gap
(node_modules/@openzeppelin/contracts-
upgradeable/access/OwnableUpgradeable.sol#77) is never
used in CollectionERC1155
(contracts/CollectionERC1155.sol#10-168)
OwnableUpgradeable.__gap
(node_modules/@openzeppelin/contracts-
upgradeable/access/OwnableUpgradeable.sol#77) is never
used in CollectionERC721
(contracts/CollectionERC721.sol#10-176)
OwnableUpgradeable.__gap

(node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#77) is never used in Market (contracts/Market.sol#11-315)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable (https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable)[0m
[92m
renounceOwnership() should be declared external:
- OwnableUpgradeable.renounceOwnership() (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#59-61)
transferOwnership(address) should be declared external:
- OwnableUpgradeable.transferOwnership(address) (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#67-70)
uri(uint256) should be declared external:
- CollectionERC1155.uri(uint256) (contracts/CollectionERC1155.sol#130-132)
- ERC1155Upgradeable.uri(uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#65-67)
balanceOfBatch(address[],uint256[]) should be declared external:
- ERC1155Upgradeable.balanceOfBatch(address[],uint256[]) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#88-104)
setApprovalForAll(address,bool) should be declared external:
- ERC1155Upgradeable.setApprovalForAll(address,bool) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#109-114)
safeTransferFrom(address,address,uint256,uint256,bytes) should be declared external:
-

ERC1155Upgradeable.safeTransferFrom(address,address,uint256,uint256,bytes) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#126-138)
safeBatchTransferFrom(address,address,uint256[],uint256[],bytes) should be declared external:
-
ERC1155Upgradeable.safeBatchTransferFrom(address,address,uint256[],uint256[],bytes) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol#143-155)
burn(address,uint256,uint256) should be declared external:
-
ERC1155BurnableUpgradeable.burn(address,uint256,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/extensions/ERC1155BurnableUpgradeable.sol#23-34)
burnBatch(address,uint256[],uint256[]) should be declared external:
-
ERC1155BurnableUpgradeable.burnBatch(address,uint256[],uint256[]) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC1155/extensions/ERC1155BurnableUpgradeable.sol#36-47)
balanceOf(address) should be declared external:
- ERC721Upgradeable.balanceOf(address) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#68-71)
name() should be declared external:
- **ERC721Upgradeable.name** (http://ERC721Upgradeable.name)() (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#85-87)
symbol() should be declared external:
- ERC721Upgradeable.symbol() (node_modules/@openzeppelin/contracts-

```
upgradeable/token/ERC721/ERC721Upgradeable.sol#92-
94)
tokenURI(uint256) should be declared external:
- CollectionERC721.tokenURI(uint256)
(contracts/CollectionERC721.sol#147-158)
- ERC721Upgradeable.tokenURI(uint256)
(node_modules/@openzeppelin/contracts-
upgradeable/token/ERC721/ERC721Upgradeable.sol#99-
104)
approve(address,uint256) should be declared external:
- ERC721Upgradeable.approve(address,uint256)
(node_modules/@openzeppelin/contracts-
upgradeable/token/ERC721/ERC721Upgradeable.sol#118-
128)
setApprovalForAll(address,bool) should be declared
external:
- ERC721Upgradeable.setApprovalForAll(address,bool)
(node_modules/@openzeppelin/contracts-
upgradeable/token/ERC721/ERC721Upgradeable.sol#142-
147)
transferFrom(address,address,uint256) should be declared
external:
-
ERC721Upgradeable.transferFrom(address,address,uint256
) (node_modules/@openzeppelin/contracts-
upgradeable/token/ERC721/ERC721Upgradeable.sol#159-
168)
safeTransferFrom(address,address,uint256) should be
declared external:
-
ERC721Upgradeable.safeTransferFrom(address,address,uint
256) (node_modules/@openzeppelin/contracts-
upgradeable/token/ERC721/ERC721Upgradeable.sol#173-
179)
burn(uint256) should be declared external:
- ERC721BurnableUpgradeable.burn(uint256)
(node_modules/@openzeppelin/contracts-
upgradeable/token/ERC721/extensions/ERC721BurnableUp
gradeable.sol#29-33)
```

getRoleMember(bytes32,uint256) should be declared external:
-
AccessControlEnumerable.getRoleMember(bytes32,uint256)
(node_modules/@openzeppelin/contracts/access/AccessControlEnumerable.sol#36-38)
getRoleMemberCount(bytes32) should be declared external:
-
AccessControlEnumerable.getRoleMemberCount(bytes32)
(node_modules/@openzeppelin/contracts/access/AccessControlEnumerable.sol#44-46)
name() should be declared external:
- **ERC20.name** (http://ERC20.name)()
(node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#61-63)
symbol() should be declared external:
- ERC20.symbol()
(node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#69-71)
decimals() should be declared external:
- ERC20.decimals()
(node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#86-88)
totalSupply() should be declared external:
- ERC20.totalSupply()
(node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#93-95)
balanceOf(address) should be declared external:
- ERC20.balanceOf(address)
(node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#100-102)
transfer(address,uint256) should be declared external:
- ERC20.transfer(address,uint256)
(node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#112-115)
approve(address,uint256) should be declared external:
- ERC20.approve(address,uint256)

```
(node_modules/@openzeppelin/contracts/token/ERC20/ER
C20.sol#131-134)
transferFrom(address,address,uint256) should be declared
external:
- ERC20.transferFrom(address,address,uint256)
(node_modules/@openzeppelin/contracts/token/ERC20/ER
C20.sol#149-163)
increaseAllowance(address,uint256) should be declared
external:
- ERC20.increaseAllowance(address,uint256)
(node_modules/@openzeppelin/contracts/token/ERC20/ER
C20.sol#177-180)
decreaseAllowance(address,uint256) should be declared
external:
- ERC20.decreaseAllowance(address,uint256)
(node_modules/@openzeppelin/contracts/token/ERC20/ER
C20.sol#196-204)
burn(uint256) should be declared external:
- ERC20Burnable.burn(uint256)
(node_modules/@openzeppelin/contracts/token/ERC20/ext
ensions/ERC20Burnable.sol#19-21)
burnFrom(address,uint256) should be declared external:
- ERC20Burnable.burnFrom(address,uint256)
(node_modules/@openzeppelin/contracts/token/ERC20/ext
ensions/ERC20Burnable.sol#34-41)
mint(address,uint256) should be declared external:
- ERC20PresetMinterPauser.mint(address,uint256)
(node_modules/@openzeppelin/contracts/token/ERC20/pre
sets/ERC20PresetMinterPauser.sol#51-54)
pause() should be declared external:
- ERC20PresetMinterPauser.pause()
(node_modules/@openzeppelin/contracts/token/ERC20/pre
sets/ERC20PresetMinterPauser.sol#65-68)
unpause() should be declared external:
- ERC20PresetMinterPauser.unpause()
(node_modules/@openzeppelin/contracts/token/ERC20/pre
sets/ERC20PresetMinterPauser.sol#79-82)
mint(bytes32,uint256) should be declared external:
- CollectionERC1155.mint(bytes32,uint256)
```

(contracts/CollectionERC1155.sol#56-64)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external (https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external)[0m
. analyzed (52 contracts with 77 detectors), 224 result(s) found