ENTERSOFT

Ekta

Smart Contract Audit (Final)

DATE: 10th March 2022 | **PREPARED BY:** Entersoft Pte Ltd of 1B Trengganu Street, Singapore 058455

Contents

Revision History & Version Control

Version	Date	Author(s)	Description
1.0	16/02/2022	Abhishek Sharma	Initial Draft of Final Report
1.0	17/02/2022	Jake Lemke	Released Initial report
1.0	10/03/2022	Jake Lemke	Released Final Report

Entersoft was commissioned to perform a source code review on their solidity smart contract.
The review was conducted between January 27th to February 7th, 2021. The report is organized into the following sections.

- Executive Summary: A high-level overview of the security audit findings.
- Technical analysis: Our detailed analysis of the Smart Contract code

The information in this report should be used to understand overall code quality, security, correctness, and meaning that code will work as described in the smart contract.

1.0 Disclaimer

This is a limited audit report on our findings based on our analysis, in accordance with good industry practice as at the date of this report, in relation to: (i) smart contract best coding practices and issues in the framework and algorithms based on white paper, code, the details of which are set out in this report, (Smart Contract audit). 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 us based on what it says or does not say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the disclaimer below – please make sure to read it in full. DISCLAIMER: 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 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 Entersoft Australia and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers, and other representatives) (Entersoft) owe no duty of care towards you or any other person, nor does Entersoft 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 Entersoft 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, Entersoft hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against Entersoft, 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 Smart contract is purely based on the smart contract code shared with us alone.

2.0 Overview

2.1 Project Overview

During the period of **9th of February - 16th of February**, Entersoft performed smart contract security audits for **Ekta**.

2.2 Scope

The scope of this audit was to analyze and document the smart contract codebase for quality, security, and correctness.

OUT-OF-SCOPE: External contracts, External Oracles, other smart contracts in the repository or imported smart contracts.

2.3 Project Summary

Project Name	Ekta
Codebase	https://github.com/
Verified	Yes
Audited	Yes
Vulnerabilities / Issues	As per report. Section 2.6

2.4 Audit Summary

Delivery Date	10th Of March
Method of Audit	Remix IDE, Truffle, Truffle Team, Solhint, Mythril, Slither, Solidity statistic analysis, Theo.
Consultants Engaged	2

2.5 Security Level References

Every issue in this report was assigned a severity level from the following classification table:

IMPACT	HIGH	**CRITICAL**	**HIGH**	**MEDIUM**	
	MEDIUM	**HIGH**	**MEDIUM**	**LOW**	
	LOW	**MEDIUM**	**LOW**	**LOW**	**INFORMATIONAL**
		HIGH	MEDIUM	LOW	

LIKELIHOOD

2.6 Vulnerability Summary

Total Critical	0
Total High	0
Total Medium	0
Total Low	0
Total Informational	0

2.7 Audit Results Overview

Audit Item	Audit Subclass	Audit Result
Overflow	-	Passed
Race Conditions	-	Passed
Permissions	Permission Vulnerability Audit Excessive Auditing Authority	Passed
Safety Design	Zeppelin Safe Math	Passed
DDOS Attack	Call Function Security	Passed
Gas Optimization	-	Passed
Design Logic	-	Passed
Know Attacks	-	Passed
Overall Audit Result	-	Passed

3.0 Executive Summary

3.1 Findings

EKTA-001	Function should be called by owner only	Medium	**RESOLVED**
EKTA-002	Re-entrancy possibility in exchange.Sol	High	**RESOLVED**
EKTA-003	Events emit after external call	High	**RESOLVED**
EKTA-004	Variable Shadowing	Low	**RESOLVED**
EKTA-005	Unused return in Exchange.Sol	High	**RESOLVED**
EKTA-006	Unused return in Exchange.sol	High	**RESOLVED**
EKTA-007	Unprotected upgradable contract	High	**RESOLVED**
EKTA-008	Unused return	High	**RESOLVED**
EKTA-009	Local Variable Shadow	High	**RESOLVED**
EKTA-010	Re-entrancy	High	**RESOLVED**
EKTA-011	Multiple Re-entrancy issues have been observed	High	**RESOLVED**
EKTA-012	Dead Code suspected	High	**RESOLVED**
EKTA-013	Unused State Variable	High	**RESOLVED**

3.2 Comments

Overall, the smart contracts are very well written, and they adhere to best security practices and industry guidelines.

4.0 Vulnerabilities

4.1 Function should be called by owner only

Severity	Confidence	Status
Medium	High	Resolved

Description:

Modifier onlyOwner should be used to update baseURI

```
   function updateBaseURI(string memory _baseUri) external {
       baseURI = _baseUri;
   }
```

Remediation:

Use modifier onlyOwner

4.2 Re-entrancy possibility in exchange.sol

Severity	Confidence	Status
High	High	Resolved

Description:

State variable is updated after external call

```
   (success) = address (order.seller).call[value: order .amount}() (Exchange/exchange.
sol#165)
IERC721Upgradeable (order. tokenAddress). transferFrom(order. seller ,msg. sender, order.
tokenId) (Exchange/exchange.sol#169-173)
External calls sending eth:
- (success) = address (order seller).call(value: order.amount]() (Exchange/exchange. sol#165)
State variables written after the call(s):
OrderStatus[hashkey] = COMPLETED ORDER _CLASS (Exchange/exchange.sol#176)
```

Remediation:

Use Safeguard for re-entrancy and do not update variable after external calls

4.3 Events emit after external call

Severity	Confidence	Status
High	High	Resolved

Description:

Event has been emitted after external call is also a type of Re-entrancy possibility, in a case like cross chain bridge or exchange, event has been considered to update state on other chain.

```
   (success) = address (order.seller).call[value: order .amount}() (Exchange/exchange.
sol#165)
IERC721Upgradeable (order. tokenAddress). transferFrom(order. seller ,msg. sender, order.
tokenId) (Exchange/exchange.sol#169-173)
External calls sending eth:
- (success) = address (order seller).call(value: order.amount]() (Exchange/exchange. sol#165)
State variables written after the call(s):
OrderStatus[hashkey] = COMPLETED ORDER _CLASS (Exchange/exchange.sol#176)
```

Remediation:

Do not emit event after external call

4.4 Variable Shadowing

Severity	Confidence	Status
Low	High	Resolved

Description:

```
ERC721BurnableUpgradeable..

(@openzeppelin/contracts-
upgradeable/token/ERC721/extensions/ERC721BurnableUpgradeable.sol#35)shadows:
ERC721Upgradeable. gap (@openzeppelin/contracts-
upgradeable/token/ERC721/ERC721Upgradeable.sol#431)
ERC165Upgradeable.

(@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#36)
ContextUpgradeable. gap (@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#31)
ERC721URIStorageUpgradeable.

ERC721Upgradeable,
(@openzeppelin/contracts-
upgradeable/token/ERC721/extensions/ERC721URIStorageUpgradeable.sol#76)shadows:

(@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#431)
ERC165Upgradeable.
(@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#36)
- ContextUpgradeable.
gap (@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#31)
```

```
AccessControlUpgradeable.
gap (@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#232)shadows:

ERC165Upgradeable.

(@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#36)
- ContextUpgradeable._

 (@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#31)
ERC20Upgradeable.  (@openzeppelin/contracts-
upgradeable/token/ERC20/ERC20Upgradeable.sol#362)shadows:
ContextUpgradeable._gap(@openzeppelin/contracts-upgradeable/utils/ContextUpgrade
able.sol#31)
UUPSUpgradeable.

(@openzeppelin/contracts-upgradeable/proxy/utils/UPSUpgradeable.sol#81)shadows:
- ERC1967UpgradeUpgradeable.
(@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable,sol#215)

ERC721Upgradeable.
(@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#431) shadows:
ERC165Upgradeable.
gap (@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable,sol#36)
ContextUpqradeable.
gap (@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#31)
PausableUpgradeable.
gap (@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#97)shadows:
ContextUpgradeable.
gap (@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#31)
OwnableUpgradeable._
gap (@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#82)shadows:
- ContextUpgradeable.
gap (@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#31)
```

Remediation:
Do not emit event after external call

4.5 Unused return in Exchange.sol

Severity	Confidence	Status
High	High	Resolved

Description:

The return value of an external call is not stored in a local or state variable.

```
IWETH(WEKTA).transferFrom(bidOrder.buyer, msg.sender, bidOrder.bidAmount);
```

Remediation:

Ensure that all the return values of the function calls are used.

4.6 Unused return in Exchange.sol

Severity	Confidence	Status
High	High	Resolved

Description:

The return value of an external call is not stored in a local or state variable.

```
IERC721Upgradeable(bidOrder.tokenAddress).transferFrom(msg.sender, bidOrder.buyer,
bidOrder.tokenId);
```

Remediation:

Ensure that all the return values of the function calls are used.

4.7 Unprotected upgradable contract

Severity	Confidence	Status
High	High	Resolved

Description:

Detects logic contract that can be destroyed

```
              EKTANFT(TokenFactory/tokens/ERC721.so(#13-71) is an upgradable contract that
does not protect its initiliaze functions: EKTANFT.initialize(string, string)
(TokenFactory/tokens/ERC721.50
initiliaze functions: Exchange.initialize() (Exchange/exchange.sol#89-98). Anyone can delete
the contract with: UPSUpgradeable.upgradeTo(address) (@openzeppelin/contracts-upgradeable/p
roxy/utils/UUPSUpgradeable.sol#52-
55)UUPSUpgradeable.upgradeToAndCall(address,bytes)(@openzeppelin/contracts-
upgradeable/proxy/utils/UUPSUpgradeable.sol#65-68)
```

Remediation:

Add a constructor to ensure initialize cannot be called on the logic contract.

4.8 Unused return

Severity	Confidence	Status
High	High	Resolved

Description:

Detects logic contract that can be destroyed.

```
ERC1967Upgrade._upgradeToAndCal1 (address, bytes, bool)
(@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#64-73) ignores return value by
Address. functionDelegateCall (newImplementati
on, data) (dopenzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#71)

ERC1967Upgrade._upgradeToAndCal1Secure(address, bytes, bool)
(@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#80-108) ignores return value by
Address. functionDelegateCall (newImple

mentation, data) (dopenzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#90)
ERC1967Upgrade._upgradeToAndCal1Secure(address, bytes, bool)
(@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#80-108) ignores return value by
Address. functionDelegateCall (newImple
mentation,abi.encodeWithsignature(upgradeTo(address),oldImplementation))(Gopenzeppelin/contra
cts/proxy/ERC1967/ERC1967Upgrade.sol#98-101)

F NE SeLLOA, tEnen7 etenca,. Ta An, a noDenteese: PinTesntracts/070317ERPi)57 EROi5 FtAbPSa.:
352F1957 /EnCi96TUpgrade. sol7 83-193) ignores return value by Address, functionbelegateCall
(ibeacon

ERC721Upgradeable._checkOnERC721Received (address, address, wint256, bytes)
(@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#389-410)ignores
return value by IERC721Re
ceiverUpgradeable(to). onERC721Received (_msgSender (), from, tokenId, data)
(dopenzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#396-406)
```

Remediation:

Ensure that all the return values of the function calls are used.

4.9 Local Variable Shadow

Severity	Confidence	Status
High	High	Resolved

Description:

Detection of shadowing using local variables.

```
EktaNft.initialise ERC721upgradable.Name ()
EktaNft.initialise ERC721upgradable.symbol ()
```

Remediation:

Rename the local variables that shadow another component.

4.10 Re-entrancy

Severity	Confidence	Status
High	High	Resolved

Description:

State variable is updated after external call

```
Reentrancy in EKTANFT.safeMint (address, string) (TokenFactory/tokens/ERC721.s01#38-43):
External calls:
_safeMint (to, tokenId) (TokenFactory/tokens/ERC721.s0l#41)
- IERC721ReceiverUpgradeable(to)-onERC721Received(_msgSender (), from, tokenId, _data)
(@openzeppelin/contracts-upgradable/token/ERC721/ERC721Upgradeable.sol#396-406)
State vartables written after the call(s):
_setTokenURI(tokenId,uri)(TokenFactory/tokens/ERC721.s01#42)
tokenURIs[tokenId]=
tokenURI(@openzeppelin/contracts-
upgradeable/token/ERC721/extensions/ERC721URIStorageUpgradeable.sol#56)
```

Remediation:

Use Safeguard for Re-entrancy and do not update variables after external calls.

4.11 Multiple Re-entrancy issues have been observed

Severity	Confidence	Status
High	High	Resolved

Description:

State variable is updated after external call

```
Reentrancy in ERC1967UpgradeUpgradeable._upgradeToAndCallSecure (address,bytes,bool)
(@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#87-115):
External calls:
functionDelegateCall(newImplementation,data)(popenzeppelin/contracts-
upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#97)
(success,returndata) = target.delegatecall(data) (@openzeppelin/contracts-
upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#212)

functionDelegateCall(newImplementation,abi.encodeWithsignature(upgradeTo(address),oldImplemen
tation))(popenzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradea
ble.sol#105-108)
(success,returndata) = target.delegatecall (data) (@openzeppelin/contracts-
upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#212)
Event emitted after the call(s):
- Upgraded

(newImplementation)(@openzeppelin/contracts-
upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#63)
_upgradeTo(newImplementation)(@openzeppelin/contracts-
upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#113)
Reentrancy in ERC1967Upgrade. _upgradeToAndCallSecure(address, bytes,bool)

(@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade. sol#80-108) :
External calls:
- Address. functionDelegateCall (newImplementation,data)
(@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#90)
Address.functionDelegateCall(newImplementation,abi.encodeWithSignature(upgradeTo(address),old
Implementation))(@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#98-101)
Event emitted after the call(s):
- Upgraded (newImplementation)(@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#56)

upgradeTo(newImplementation)(@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#106)

Reentrancy in Exchange.completeBidding (Exchange.BidOrder,bytes) (Exchange/exchange.sol#190-
255) :
External calls:
-
INETH(WEKTA).transferFrom(bidOrder.buyer,msg.sender,bidorder.bidAnount)(Exchange/exchange.sol
#233-237)
IERC721Upgradeable(bidorder. tokenAddress). transferFron(msg.sender ,bidorder.buyer,bidorder.
tokenId) (Exchange/exchange.sol#240-244)
Event emitted after the call(s):
Buy(bidOrder.seller,bidorder.buyer,bidorder.tokenAddress,bidorder.tokenId)(Exchange/exchange.
sol#249-254)
```

```
Reentrancy in Exchange.completeFixedSale (Exchange.Order) (Exchange/exchange.sol#140-179) :
External calls:
(success) = address (order.seller).call{value: order.amount]() (Exchange/exchange. sol#165)
IERC721Upgradeable(order.tokenAddress).transferFron(order.seller,msq.sender,order.tokenId)(Ex
change/exchange.sol#169-173)

External calls sending eth:

- (success) = address (order.seller).callivalue: order.amount]() (Exchange/exchange.sol#165)
Event emitted after the call(s):
- Buy(order.seller,msg.sender,order. tokenAddress, order. tokenId)

(Exchange/exchange.sol#178)
Reentrancy in ERC721Factory.createToken (string, string, string,wint256)
(TokenFactory/factories/ERC721TokenFactory.sol#50-56) :
External calls:

token.updateBaseURI(_baseUri)(TokenFactory/factories/ERC721TokenFactory.sol#53)
token.transferOwnership(nsq.sender)(TokenFactory/factories/ERC721TokenFactory.sol#54)

Event emitted after the call(s):
Create721 (address (token)) (TokenFactory/factories/ERC721TokenFactory.sol#55)
```

Remediation:

Use Safeguard for Re-entrancy and do not update variable after external calls.

4.12 Dead Code suspected

Severity	Confidence	Status
High	High	Resolved

Description:

Functions that are not sued.

```
   /**
    * @dev deploying contract with create2
    *
    * Requirements:
    * - @param bytecode.
    * - @param salt.
    */
   function deploy(bytes memory bytecode, uint salt) internal returns(address){
       address tokenAddress;
       assembly {
           tokenAddress := create2(0, add(bytecode, 0x20), mload(bytecode), salt)
       }
       return tokenAddress;
   }
```

Remediation:

Remove unused functions.

4.13 Unused State Variable

Severity	Confidence	Status
High	High	Resolved

Description:

State variable is not used in a contract call

```
  ERC721BurnableUpgradeable.
_gap (@openzeppelin/contracts-
upgradeable/token/ERC721/extensions/ERC721BurnableUpgradeable.sol#35)is never used in EKTANFT
(TokenFactory/tokens/ERC721.50l#13
-71)
AccessControlupgradeable.
gap (@openzeppelin/contracts-
upgradeable/access/AccessControlUpgradeable.sol#232)isneverusedinExchange(Exchange/exchange.s
ol#22-381)
ERC20Upgradeable..
_gap (@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#362)is never used
in ERC20Upgradeable (@openzeppelin/contracts-upgradeable/token/ERC20/ERC20
Upgradable. soL#36-363)
```

Remediation:

Remove unused state variables.

5.0 Ekta Functional Tests

The following is the list of functions tested and checked for vulnerabilities during the audit:

Function Name()	Technical Result	Logical Result	Overall Result
safeWithdrawBnb	Pass	Pass	Pass
approve	Pass	Pass	Pass
allowance	Pass	Pass	Pass
transfer	Pass	Pass	Pass
transferFrom	Pass	Pass	Pass

6.0 Automated Testing

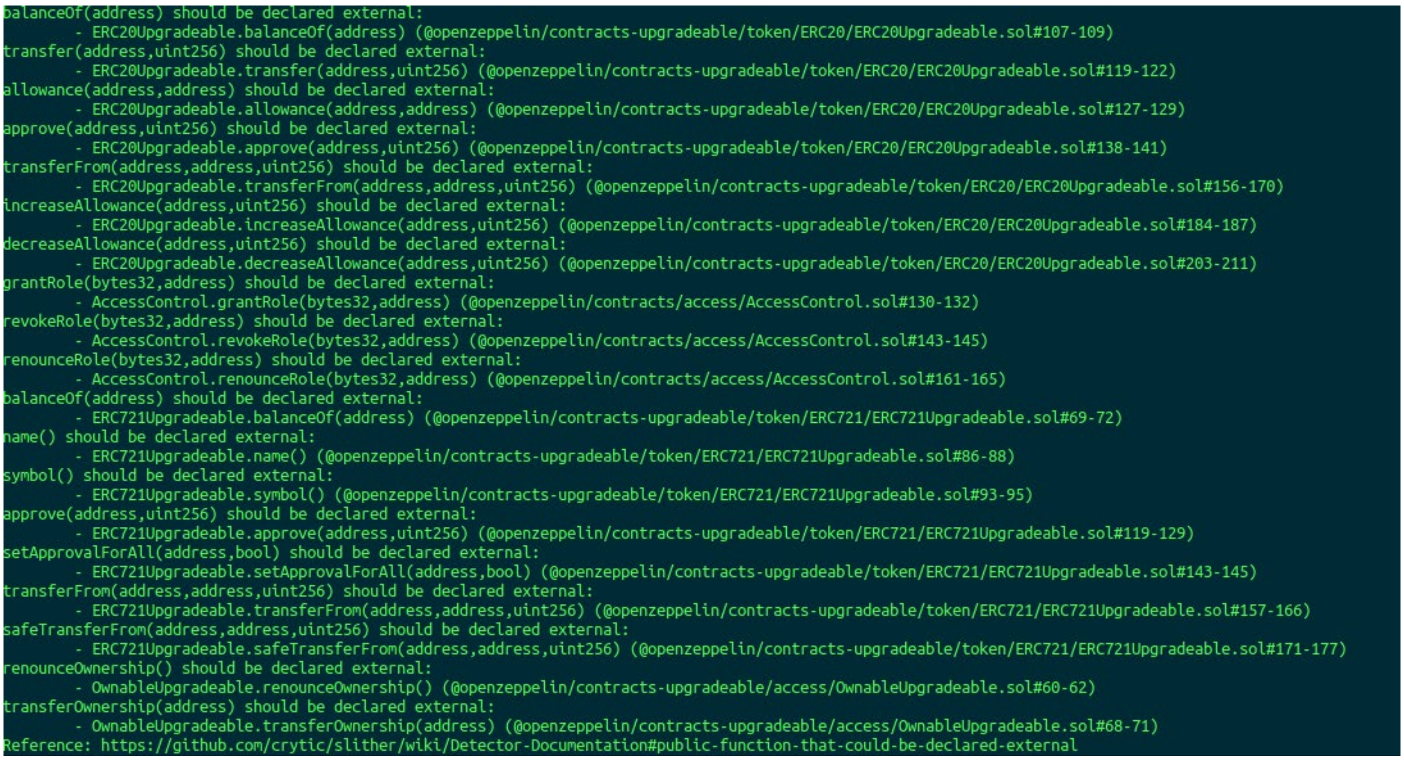
Automated testing is carried out with the following tools:

- Slither
- Mythril
- Echidna
- Manticore
- Surya
- Solhint

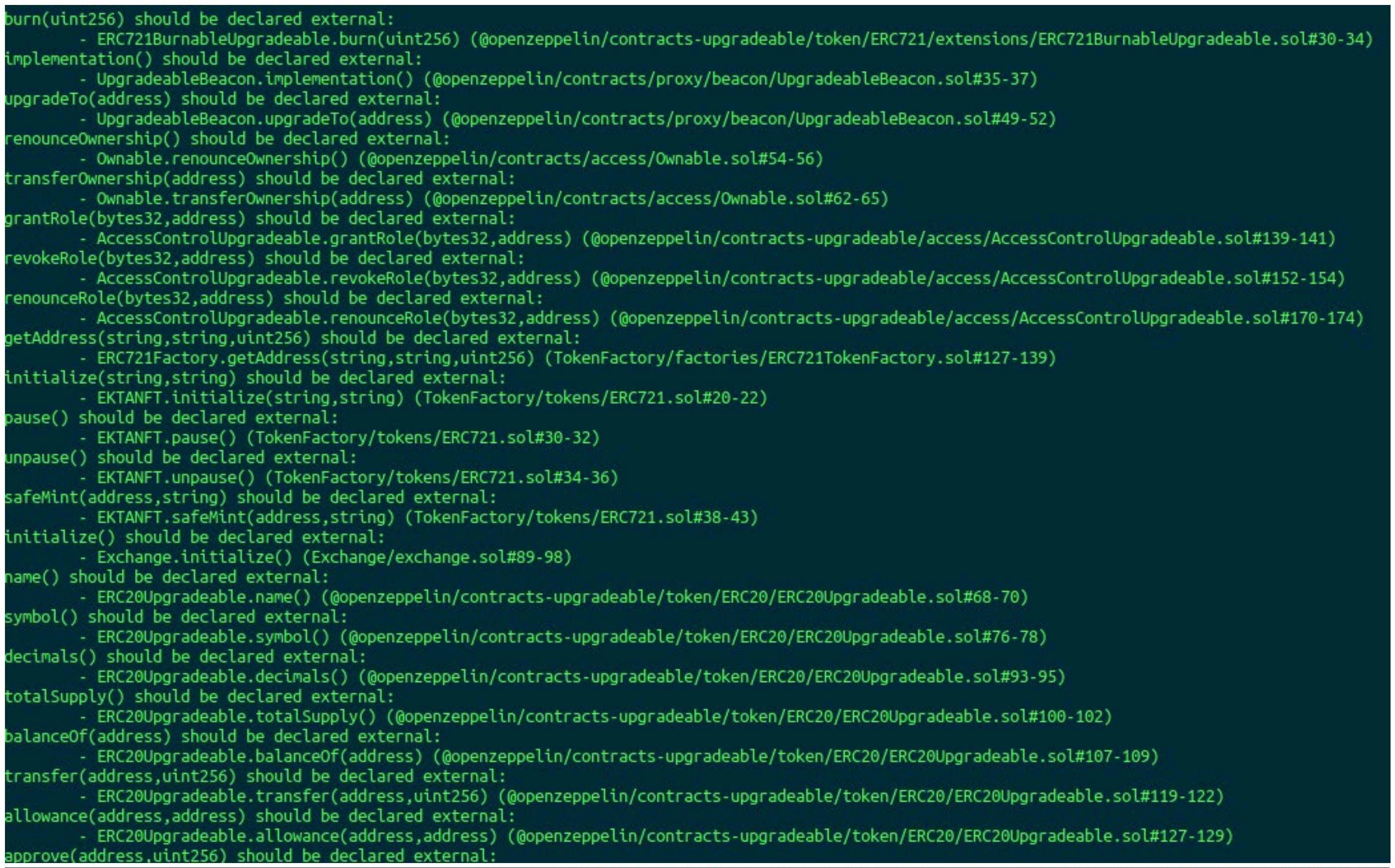
6.1 Slither

Slither is an open-source Solidity static analysis framework. This tool provides rich information about Ethereum smart contracts while ensuring all critical properties. It runs a suite of vulnerability detectors, prints visual information about contract details, and provides an API to easily write custom analyses.

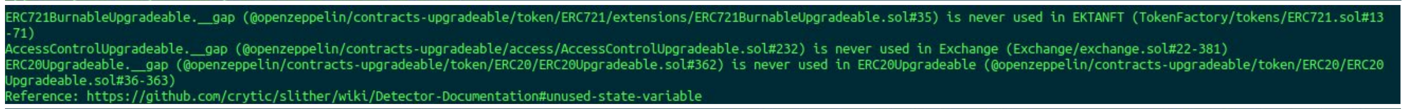
No major issues were found. Some false positive errors were reported by the tool. All the other issues have been categorized above according to their level of severity.



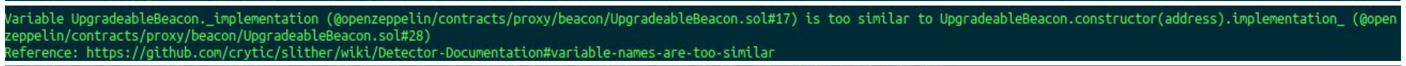
```
balanceOf(address) should be declared external:
        - ERC20Upgradeable.balanceOf(address) (@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#107-109)
transfer(address,uint256) should be declared external:
        - ERC20Upgradeable.transfer(address,uint256) (@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#119-122)
allowance(address,address) should be declared external:
        - ERC20Upgradeable.allowance(address,address) (@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#127-129)
approve(address,uint256) should be declared external:
        - ERC20Upgradeable.approve(address,uint256) (@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#138-141)
transferFrom(address,address,uint256) should be declared external:
        - ERC20Upgradeable.transferFrom(address,address,uint256) (@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#156-170)
increaseAllowance(address,uint256) should be declared external:
        - ERC20Upgradeable.increaseAllowance(address,uint256) (@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#184-187)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20Upgradeable.decreaseAllowance(address,uint256) (@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#203-211)
grantRole(bytes32,address) should be declared external:
        - AccessControl.grantRole(bytes32,address) (@openzeppelin/contracts/access/AccessControl.sol#130-132)
revokeRole(bytes32,address) should be declared external:
        - AccessControl.revokeRole(bytes32,address) (@openzeppelin/contracts/access/AccessControl.sol#143-145)
renounceRole(bytes32,address) should be declared external:
        - AccessControl.renounceRole(bytes32,address) (@openzeppelin/contracts/access/AccessControl.sol#161-165)
balanceOf(address) should be declared external:
        - ERC721Upgradeable.balanceOf(address) (@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#69-72)
name() should be declared external:
        - ERC721Upgradeable.name() (@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#86-88)
symbol() should be declared external:
        - ERC721Upgradeable.symbol() (@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#93-95)
approve(address,uint256) should be declared external:
        - ERC721Upgradeable.approve(address,uint256) (@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#119-129)
setApprovalForAll(address,bool) should be declared external:
        - ERC721Upgradeable.setApprovalForAll(address,bool) (@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#143-145)
transferFrom(address,address,uint256) should be declared external:
        - ERC721Upgradeable.transferFrom(address,address,uint256) (@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#157-166)
safeTransferFrom(address,address,uint256) should be declared external:
        - ERC721Upgradeable.safeTransferFrom(address,address,uint256) (@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#171-177)
renounceOwnership() should be declared external:
        - OwnableUpgradeable.renounceOwnership() (@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#60-62)
transferOwnership(address) should be declared external:
        - OwnableUpgradeable.transferOwnership(address) (@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#68-71)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```



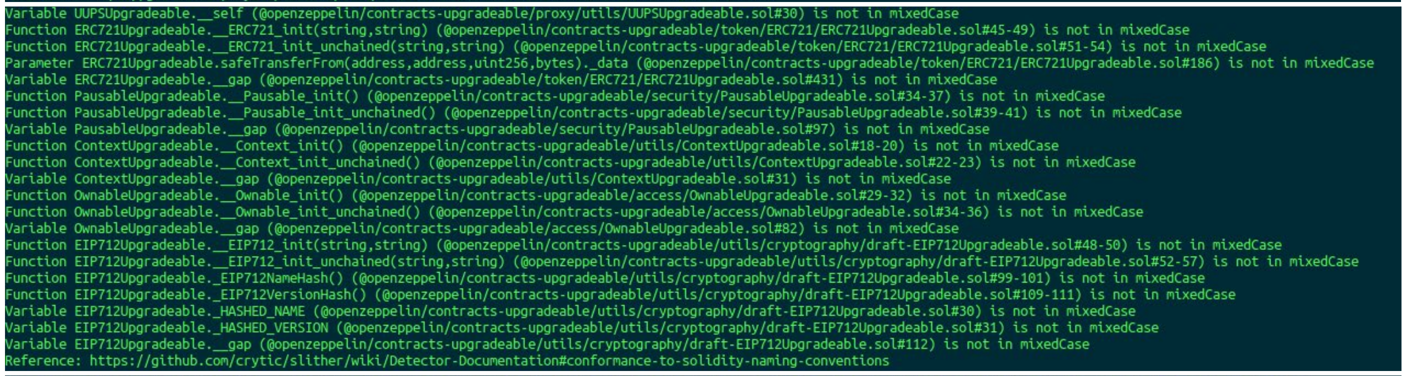
```
burn(uint256) should be declared external:
        - ERC721BurnableUpgradeable.burn(uint256) (@openzeppelin/contracts-upgradeable/token/ERC721/extensions/ERC721BurnableUpgradeable.sol#30-34)
implementation() should be declared external:
        - UpgradeableBeacon.implementation() (@openzeppelin/contracts/proxy/beacon/UpgradeableBeacon.sol#35-37)
upgradeTo(address) should be declared external:
        - UpgradeableBeacon.upgradeTo(address) (@openzeppelin/contracts/proxy/beacon/UpgradeableBeacon.sol#49-52)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (@openzeppelin/contracts/access/Ownable.sol#54-56)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (@openzeppelin/contracts/access/Ownable.sol#62-65)
grantRole(bytes32,address) should be declared external:
        - AccessControlUpgradeable.grantRole(bytes32,address) (@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#139-141)
revokeRole(bytes32,address) should be declared external:
        - AccessControlUpgradeable.revokeRole(bytes32,address) (@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#152-154)
renounceRole(bytes32,address) should be declared external:
        - AccessControlUpgradeable.renounceRole(bytes32,address) (@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#170-174)
getAddress(string,string,uint256) should be declared external:
        - ERC721Factory.getAddress(string,string,uint256) (TokenFactory/factories/ERC721TokenFactory.sol#127-139)
initialize(string,string) should be declared external:
        - EKTANFT.initialize(string,string) (TokenFactory/tokens/ERC721.sol#20-22)
pause() should be declared external:
        - EKTANFT.pause() (TokenFactory/tokens/ERC721.sol#30-32)
unpause() should be declared external:
        - EKTANFT.unpause() (TokenFactory/tokens/ERC721.sol#34-36)
safeMint(address,string) should be declared external:
        - EKTANFT.safeMint(address,string) (TokenFactory/tokens/ERC721.sol#38-43)
initialize() should be declared external:
        - Exchange.initialize() (Exchange/exchange.sol#89-98)
name() should be declared external:
        - ERC20Upgradeable.name() (@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#68-70)
symbol() should be declared external:
        - ERC20Upgradeable.symbol() (@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#76-78)
decimals() should be declared external:
        - ERC20Upgradeable.decimals() (@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#93-95)
totalSupply() should be declared external:
        - ERC20Upgradeable.totalSupply() (@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#100-102)
balanceOf(address) should be declared external:
        - ERC20Upgradeable.balanceOf(address) (@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#107-109)
transfer(address,uint256) should be declared external:
        - ERC20Upgradeable.transfer(address,uint256) (@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#119-122)
allowance(address,address) should be declared external:
        - ERC20Upgradeable.allowance(address,address) (@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#127-129)
approve(address,uint256) should be declared external:
```



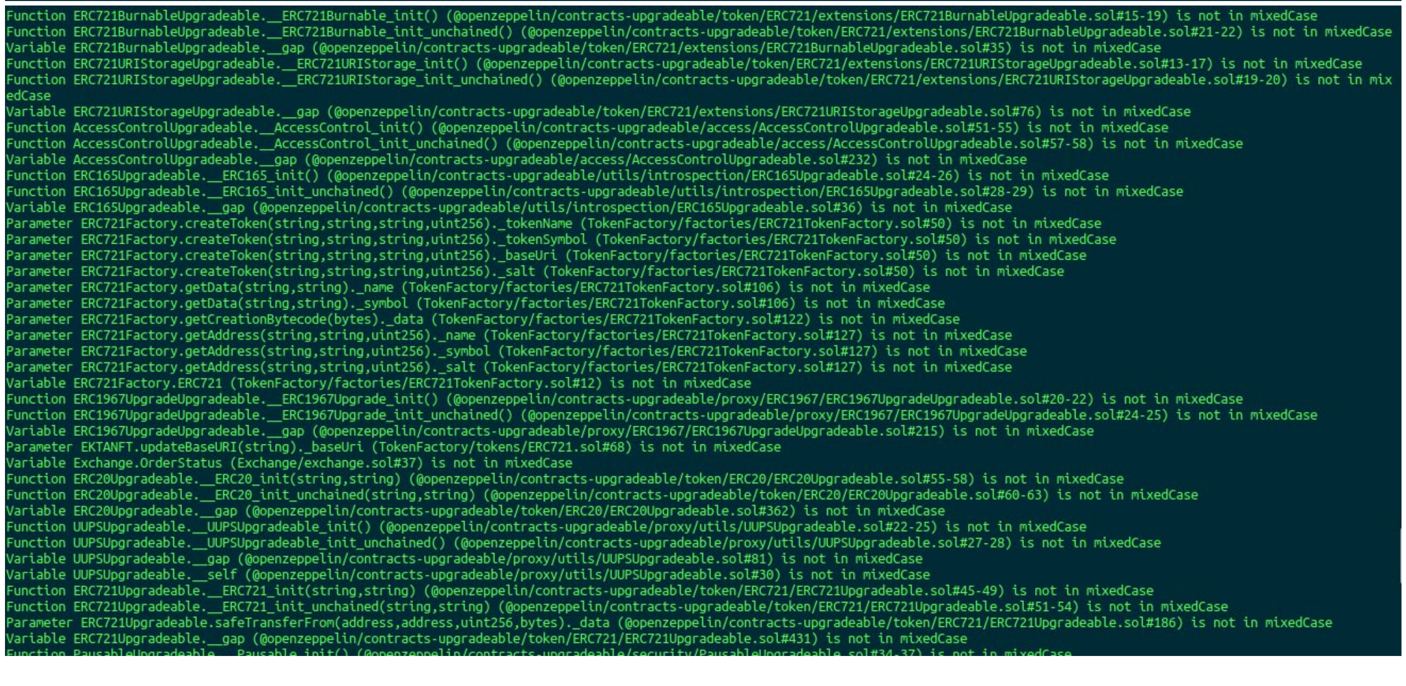
```
ERC721BurnableUpgradeable.__gap (@openzeppelin/contracts-upgradeable/token/ERC721/extensions/ERC721BurnableUpgradeable.sol#35) is never used in EKTANFT (TokenFactory/tokens/ERC721.sol#13
-71)
AccessControlUpgradeable.__gap (@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#232) is never used in Exchange (Exchange/exchange.sol#22-381)
ERC20Upgradeable.__gap (@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#362) is never used in ERC20Upgradeable (@openzeppelin/contracts-upgradeable/token/ERC20/ERC20
Upgradeable.sol#36-363)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable
```



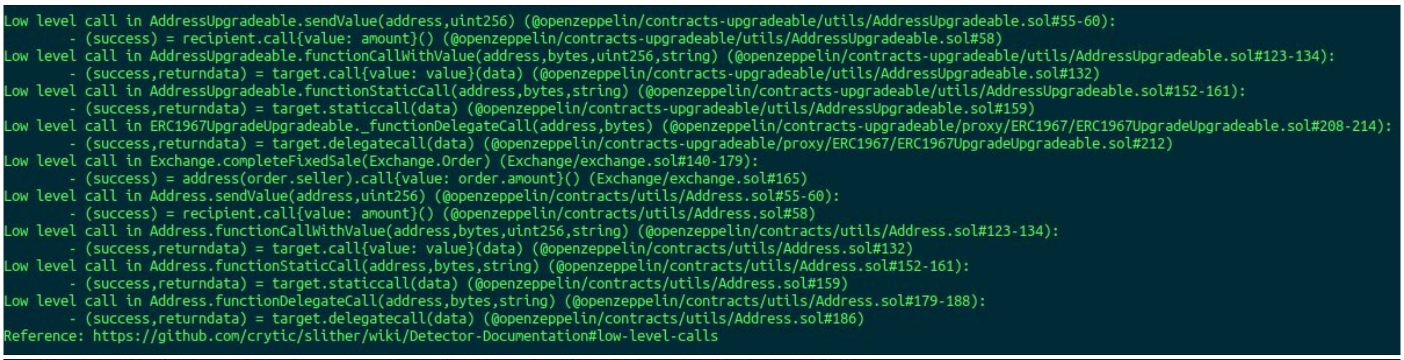
```
Variable UpgradeableBeacon._implementation (@openzeppelin/contracts/proxy/beacon/UpgradeableBeacon.sol#17) is too similar to UpgradeableBeacon.constructor(address).implementation_ (@open
zeppelin/contracts/proxy/beacon/UpgradeableBeacon.sol#28)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
```



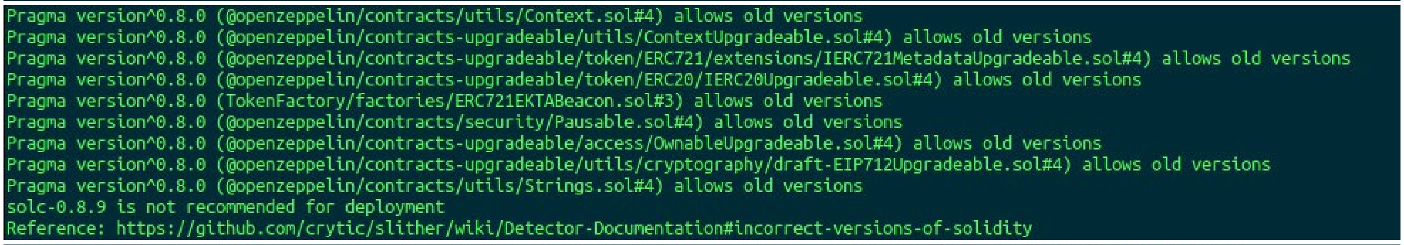
```
Variable UUPSUpgradeable.__self (@openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol#30) is not in mixedCase
Function ERC721Upgradeable.__ERC721_init(string,string) (@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#45-49) is not in mixedCase
Function ERC721Upgradeable.__ERC721_init_unchained(string,string) (@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#51-54) is not in mixedCase
Parameter ERC721Upgradeable.safeTransferFrom(address,address,uint256,bytes)._data (@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#186) is not in mixedCase
Variable ERC721Upgradeable.__gap (@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#431) is not in mixedCase
Function PausableUpgradeable.__Pausable_init() (@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#34-37) is not in mixedCase
Function PausableUpgradeable.__Pausable_init_unchained() (@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#39-41) is not in mixedCase
Variable PausableUpgradeable.__gap (@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#97) is not in mixedCase
Function ContextUpgradeable.__Context_init() (@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#18-20) is not in mixedCase
Function ContextUpgradeable.__Context_init_unchained() (@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#22-23) is not in mixedCase
Variable ContextUpgradeable.__gap (@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#31) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init() (@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#29-32) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init_unchained() (@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#34-36) is not in mixedCase
Variable OwnableUpgradeable.__gap (@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#82) is not in mixedCase
Function EIP712Upgradeable.__EIP712_init(string,string) (@openzeppelin/contracts-upgradeable/utils/cryptography/draft-EIP712Upgradeable.sol#48-50) is not in mixedCase
Function EIP712Upgradeable.__EIP712_init_unchained(string,string) (@openzeppelin/contracts-upgradeable/utils/cryptography/draft-EIP712Upgradeable.sol#52-57) is not in mixedCase
Function EIP712Upgradeable._EIP712NameHash() (@openzeppelin/contracts-upgradeable/utils/cryptography/draft-EIP712Upgradeable.sol#99-101) is not in mixedCase
Function EIP712Upgradeable._EIP712VersionHash() (@openzeppelin/contracts-upgradeable/utils/cryptography/draft-EIP712Upgradeable.sol#109-111) is not in mixedCase
Variable EIP712Upgradeable._HASHED_NAME (@openzeppelin/contracts-upgradeable/utils/cryptography/draft-EIP712Upgradeable.sol#30) is not in mixedCase
Variable EIP712Upgradeable._HASHED_VERSION (@openzeppelin/contracts-upgradeable/utils/cryptography/draft-EIP712Upgradeable.sol#31) is not in mixedCase
Variable EIP712Upgradeable.__gap (@openzeppelin/contracts-upgradeable/utils/cryptography/draft-EIP712Upgradeable.sol#112) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```



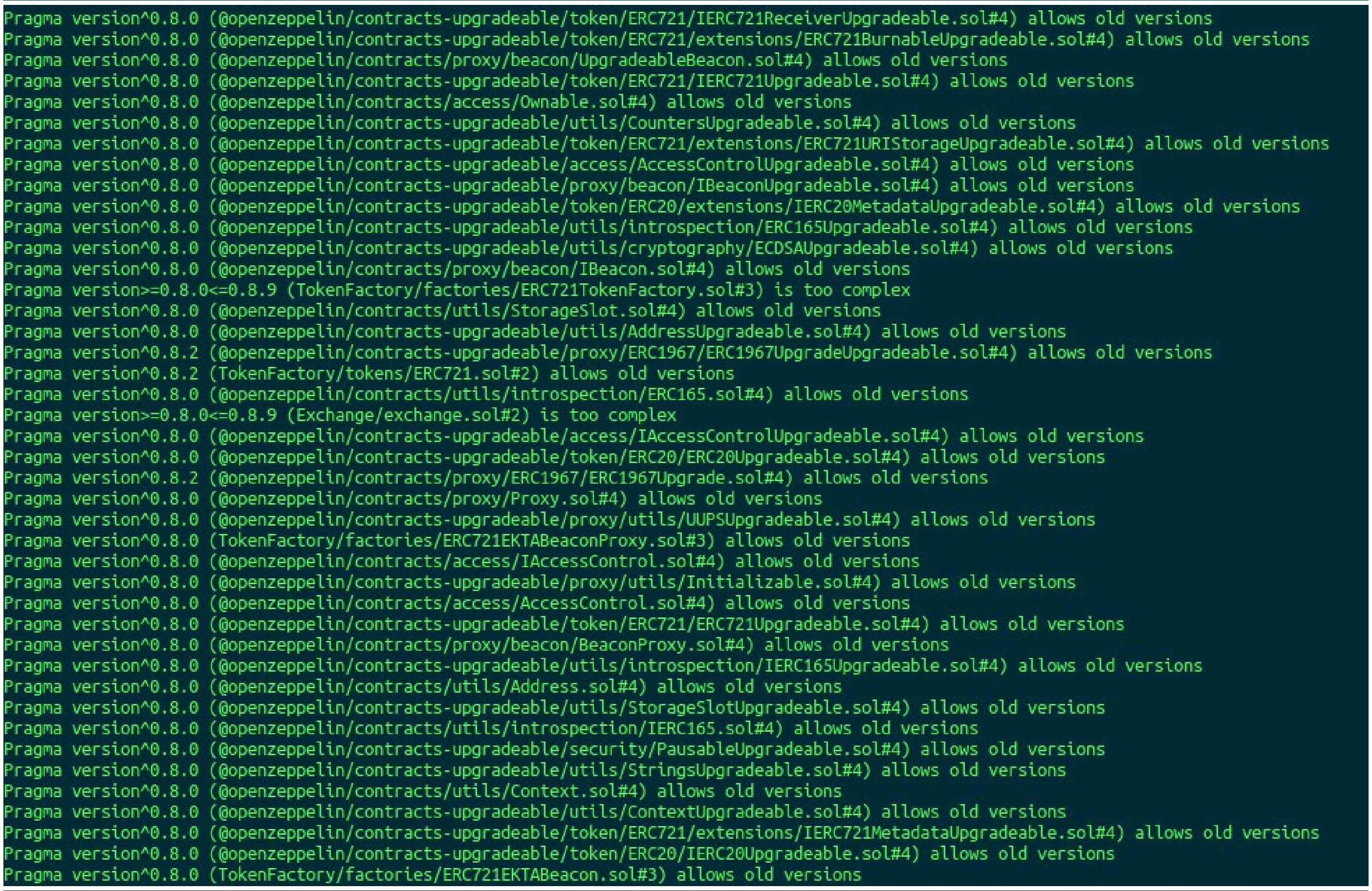
```
Function ERC721BurnableUpgradeable.__ERC721Burnable_init() (@openzeppelin/contracts-upgradeable/token/ERC721/extensions/ERC721BurnableUpgradeable.sol#15-19) is not in mixedCase
Function ERC721BurnableUpgradeable.__ERC721Burnable_init_unchained() (@openzeppelin/contracts-upgradeable/token/ERC721/extensions/ERC721BurnableUpgradeable.sol#21-22) is not in mixedCase
Variable ERC721BurnableUpgradeable.__gap (@openzeppelin/contracts-upgradeable/token/ERC721/extensions/ERC721BurnableUpgradeable.sol#35) is not in mixedCase
Function ERC721URIStorageUpgradeable.__ERC721URIStorage_init() (@openzeppelin/contracts-upgradeable/token/ERC721/extensions/ERC721URIStorageUpgradeable.sol#13-17) is not in mixedCase
Function ERC721URIStorageUpgradeable.__ERC721URIStorage_init_unchained() (@openzeppelin/contracts-upgradeable/token/ERC721/extensions/ERC721URIStorageUpgradeable.sol#19-20) is not in mix
edCase
Variable ERC721URIStorageUpgradeable.__gap (@openzeppelin/contracts-upgradeable/token/ERC721/extensions/ERC721URIStorageUpgradeable.sol#76) is not in mixedCase
Function AccessControlUpgradeable.__AccessControl_init() (@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#51-55) is not in mixedCase
Function AccessControlUpgradeable.__AccessControl_init_unchained() (@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#57-58) is not in mixedCase
Variable AccessControlUpgradeable.__gap (@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#232) is not in mixedCase
Function ERC165Upgradeable.__ERC165_init() (@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#24-26) is not in mixedCase
Function ERC165Upgradeable.__ERC165_init_unchained() (@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#28-29) is not in mixedCase
Variable ERC165Upgradeable.__gap (@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#36) is not in mixedCase
Parameter ERC721Factory.createToken(string,string,string,uint256)._tokenName (TokenFactory/factories/ERC721TokenFactory.sol#50) is not in mixedCase
Parameter ERC721Factory.createToken(string,string,string,uint256)._tokenSymbol (TokenFactory/factories/ERC721TokenFactory.sol#50) is not in mixedCase
Parameter ERC721Factory.createToken(string,string,string,uint256)._baseUri (TokenFactory/factories/ERC721TokenFactory.sol#50) is not in mixedCase
Parameter ERC721Factory.createToken(string,string,string,uint256)._salt (TokenFactory/factories/ERC721TokenFactory.sol#50) is not in mixedCase
Parameter ERC721Factory.getData(string,string)._name (TokenFactory/factories/ERC721TokenFactory.sol#106) is not in mixedCase
Parameter ERC721Factory.getData(string,string)._symbol (TokenFactory/factories/ERC721TokenFactory.sol#106) is not in mixedCase
Parameter ERC721Factory.getCreationBytecode(bytes)._data (TokenFactory/factories/ERC721TokenFactory.sol#122) is not in mixedCase
Parameter ERC721Factory.getAddress(string,string,uint256)._name (TokenFactory/factories/ERC721TokenFactory.sol#127) is not in mixedCase
Parameter ERC721Factory.getAddress(string,string,uint256)._symbol (TokenFactory/factories/ERC721TokenFactory.sol#127) is not in mixedCase
Parameter ERC721Factory.getAddress(string,string,uint256)._salt (TokenFactory/factories/ERC721TokenFactory.sol#127) is not in mixedCase
Variable ERC721Factory.ERC721 (TokenFactory/factories/ERC721TokenFactory.sol#12) is not in mixedCase
Function ERC1967UpgradeUpgradeable.__ERC1967Upgrade_init() (@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#20-22) is not in mixedCase
Function ERC1967UpgradeUpgradeable.__ERC1967Upgrade_init_unchained() (@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#24-25) is not in mixedCase
Variable ERC1967UpgradeUpgradeable.__gap (@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#215) is not in mixedCase
Parameter EKTANFT.updateBaseURI(string)._baseUri (TokenFactory/tokens/ERC721.sol#68) is not in mixedCase
Variable Exchange.OrderStatus (Exchange/exchange.sol#37) is not in mixedCase
Function ERC20Upgradeable.__ERC20_init(string,string) (@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#55-58) is not in mixedCase
Function ERC20Upgradeable.__ERC20_init_unchained(string,string) (@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#60-63) is not in mixedCase
Variable ERC20Upgradeable.__gap (@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#362) is not in mixedCase
Function UUPSUpgradeable.__UUPSUpgradeable_init() (@openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol#22-25) is not in mixedCase
Function UUPSUpgradeable.__UUPSUpgradeable_init_unchained() (@openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol#27-28) is not in mixedCase
Variable UUPSUpgradeable.__gap (@openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol#81) is not in mixedCase
Variable UUPSUpgradeable.__self (@openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol#30) is not in mixedCase
Function ERC721Upgradeable.__ERC721_init(string,string) (@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#45-49) is not in mixedCase
Function ERC721Upgradeable.__ERC721_init_unchained(string,string) (@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#51-54) is not in mixedCase
Parameter ERC721Upgradeable.safeTransferFrom(address,address,uint256,bytes)._data (@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#186) is not in mixedCase
Variable ERC721Upgradeable.__gap (@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#431) is not in mixedCase
Function PausableUpgradeable.__Pausable_init() (@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#34-37) is not in mixedCase
```



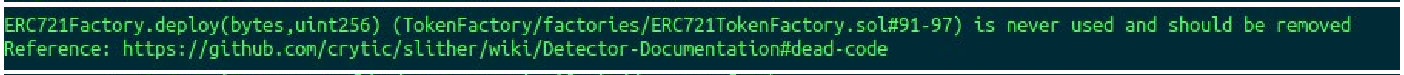
```
Low level call in AddressUpgradeable.sendValue(address,uint256) (@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#55-60):
        - (success) = recipient.call{value: amount}() (@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#58)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#123-134):
        - (success,returndata) = target.call{value: value}(data) (@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#132)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#152-161):
        - (success,returndata) = target.staticcall(data) (@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#159)
Low level call in ERC1967UpgradeUpgradeable._functionDelegateCall(address,bytes) (@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#208-214):
        - (success,returndata) = target.delegatecall(data) (@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#212)
Low level call in Exchange.completeFixedSale(Exchange.Order) (Exchange/exchange.sol#140-179):
        - (success) = address(order.seller).call{value: order.amount}() (Exchange/exchange.sol#165)
Low level call in Address.sendValue(address,uint256) (@openzeppelin/contracts/utils/Address.sol#55-60):
        - (success) = recipient.call{value: amount}() (@openzeppelin/contracts/utils/Address.sol#58)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (@openzeppelin/contracts/utils/Address.sol#123-134):
        - (success,returndata) = target.call{value: value}(data) (@openzeppelin/contracts/utils/Address.sol#132)
Low level call in Address.functionStaticCall(address,bytes,string) (@openzeppelin/contracts/utils/Address.sol#152-161):
        - (success,returndata) = target.staticcall(data) (@openzeppelin/contracts/utils/Address.sol#159)
Low level call in Address.functionDelegateCall(address,bytes,string) (@openzeppelin/contracts/utils/Address.sol#179-188):
        - (success,returndata) = target.delegatecall(data) (@openzeppelin/contracts/utils/Address.sol#186)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```



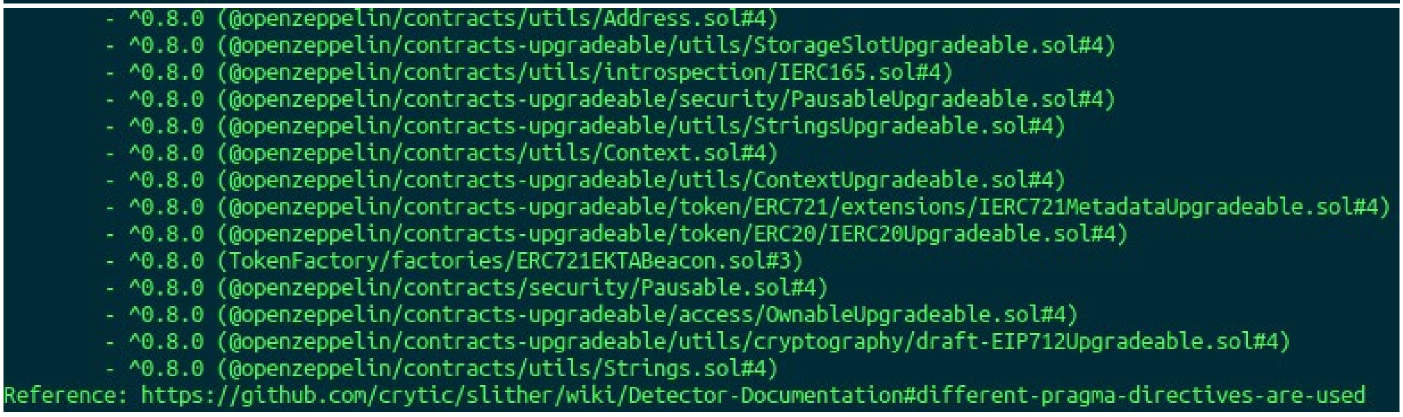
```
Pragma version^0.8.0 (@openzeppelin/contracts/utils/Context.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts-upgradeable/token/ERC721/extensions/IERC721MetadataUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol#4) allows old versions
Pragma version^0.8.0 (TokenFactory/factories/ERC721EKTABeacon.sol#3) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts/security/Pausable.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts-upgradeable/utils/cryptography/draft-EIP712Upgradeable.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts/utils/Strings.sol#4) allows old versions
solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```



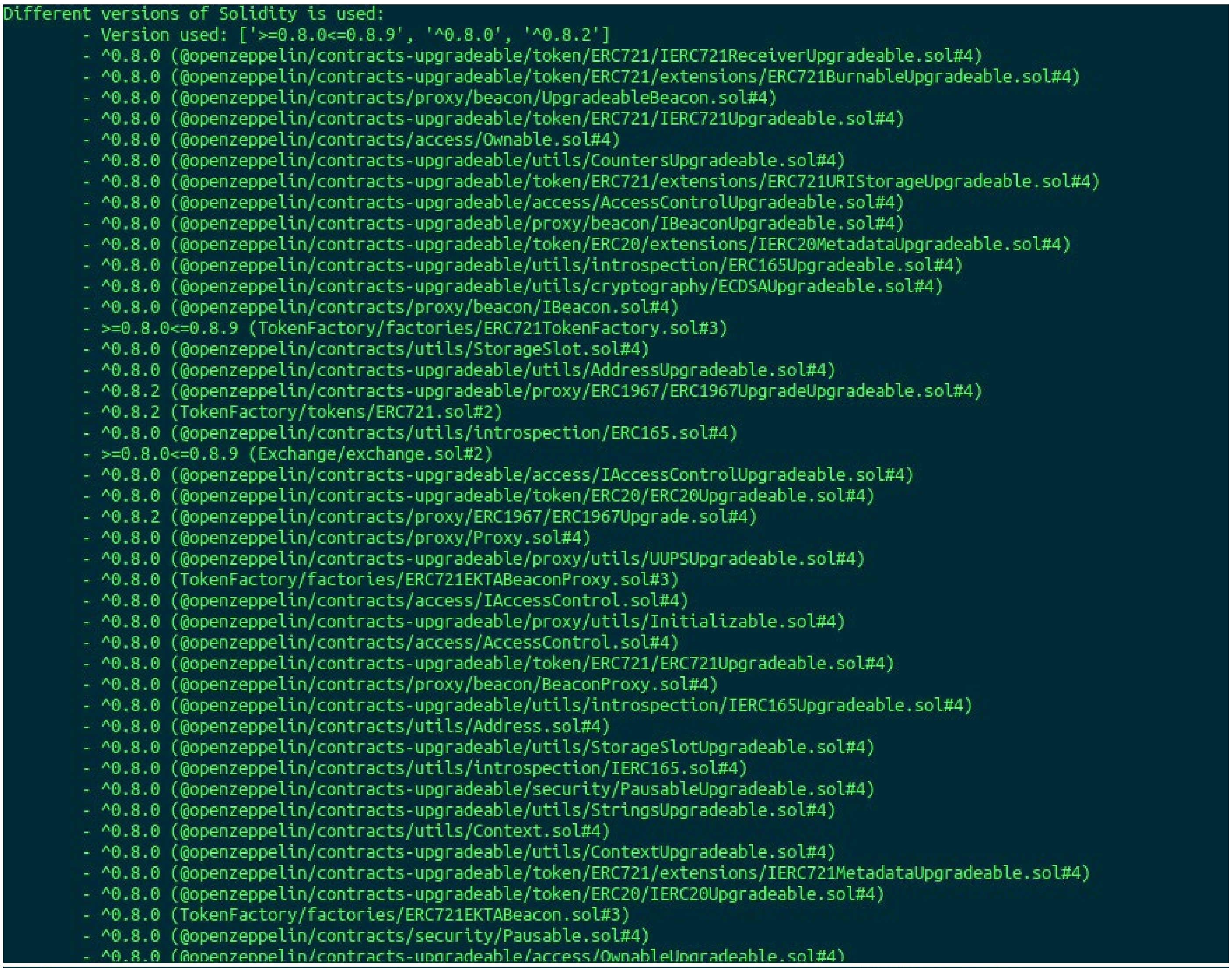
```
Pragma version^0.8.0 (@openzeppelin/contracts-upgradeable/token/ERC721/IERC721ReceiverUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts-upgradeable/token/ERC721/extensions/ERC721BurnableUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts/proxy/beacon/UpgradeableBeacon.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts-upgradeable/token/ERC721/IERC721Upgradeable.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts/access/Ownable.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts-upgradeable/utils/CountersUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts-upgradeable/token/ERC721/extensions/ERC721URIStorageUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts-upgradeable/proxy/beacon/IBeaconUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts-upgradeable/token/ERC20/extensions/IERC20MetadataUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts-upgradeable/utils/cryptography/ECDSAUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts/proxy/beacon/IBeacon.sol#4) allows old versions
Pragma version>=0.8.0<=0.8.9 (TokenFactory/factories/ERC721TokenFactory.sol#3) is too complex
Pragma version^0.8.0 (@openzeppelin/contracts/utils/StorageSlot.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#4) allows old versions
Pragma version^0.8.2 (@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#4) allows old versions
Pragma version^0.8.2 (TokenFactory/tokens/ERC721.sol#2) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts/utils/introspection/ERC165.sol#4) allows old versions
Pragma version>=0.8.0<=0.8.9 (Exchange/exchange.sol#2) is too complex
Pragma version^0.8.0 (@openzeppelin/contracts-upgradeable/access/IAccessControlUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#4) allows old versions
Pragma version^0.8.2 (@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts/proxy/Proxy.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (TokenFactory/factories/ERC721EKTABeaconProxy.sol#3) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts/access/IAccessControl.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts/access/AccessControl.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts/proxy/beacon/BeaconProxy.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts-upgradeable/utils/introspection/IERC165Upgradeable.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts/utils/Address.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts/utils/introspection/IERC165.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts-upgradeable/utils/StringsUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts/utils/Context.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts-upgradeable/token/ERC721/extensions/IERC721MetadataUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol#4) allows old versions
Pragma version^0.8.0 (TokenFactory/factories/ERC721EKTABeacon.sol#3) allows old versions
```



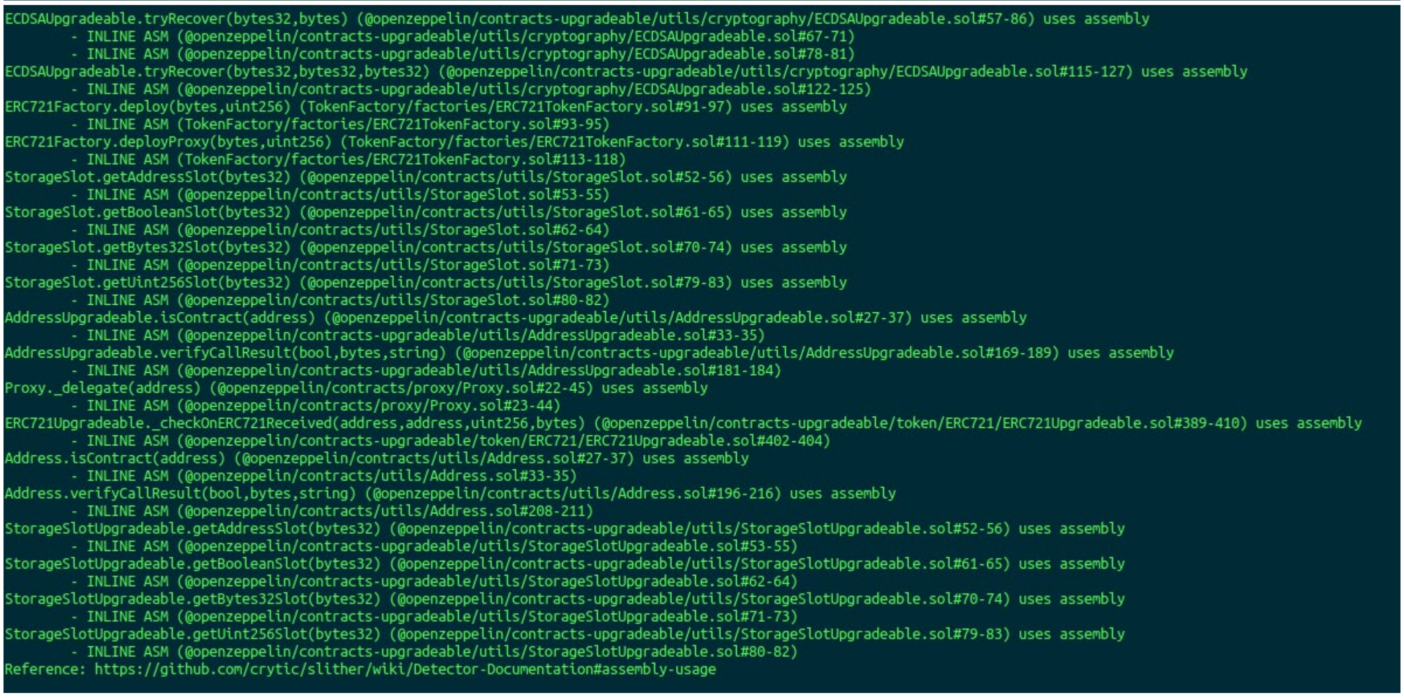
```
ERC721Factory.deploy(bytes,uint256) (TokenFactory/factories/ERC721TokenFactory.sol#91-97) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
```



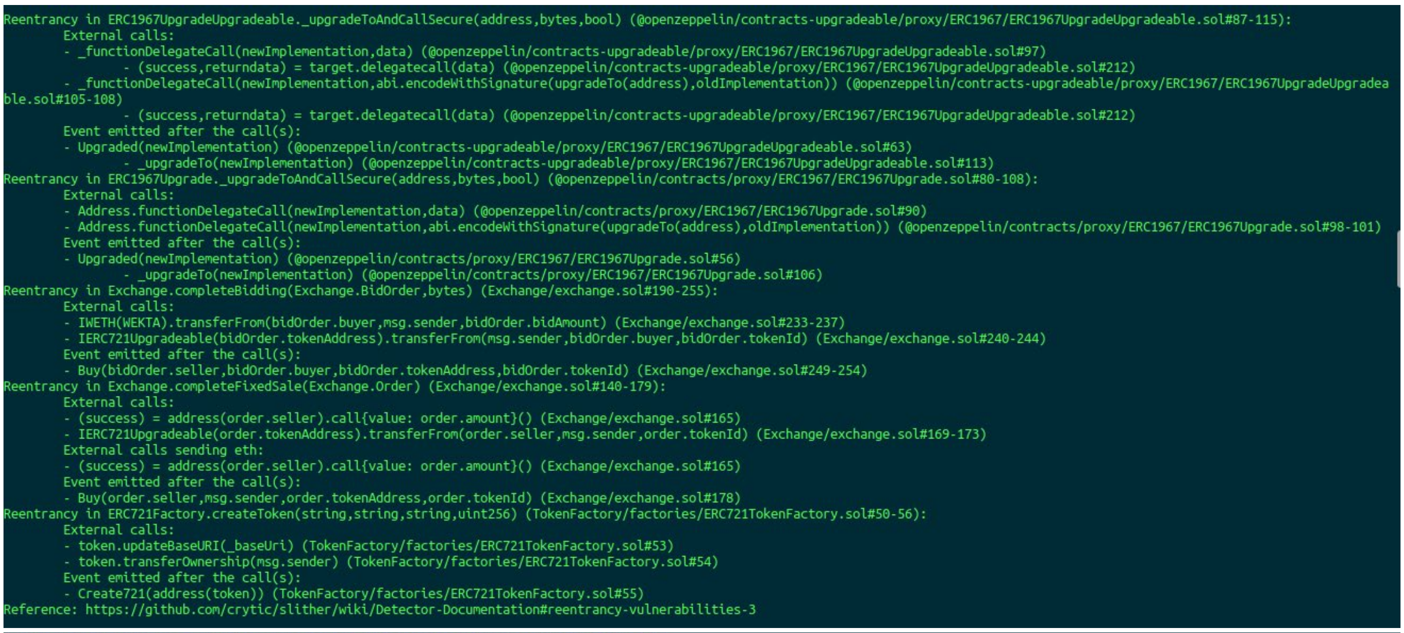
```
        - ^0.8.0 (@openzeppelin/contracts/utils/Address.sol#4)
        - ^0.8.0 (@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/utils/introspection/IERC165.sol#4)
        - ^0.8.0 (@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#4)
        - ^0.8.0 (@openzeppelin/contracts-upgradeable/utils/StringsUpgradeable.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/utils/Context.sol#4)
        - ^0.8.0 (@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#4)
        - ^0.8.0 (@openzeppelin/contracts-upgradeable/token/ERC721/extensions/IERC721MetadataUpgradeable.sol#4)
        - ^0.8.0 (@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol#4)
        - ^0.8.0 (TokenFactory/factories/ERC721EKTABeacon.sol#3)
        - ^0.8.0 (@openzeppelin/contracts/security/Pausable.sol#4)
        - ^0.8.0 (@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#4)
        - ^0.8.0 (@openzeppelin/contracts-upgradeable/utils/cryptography/draft-EIP712Upgradeable.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/utils/Strings.sol#4)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
```



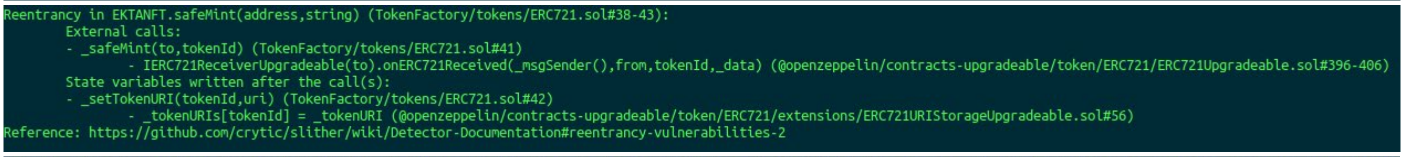
```
Different versions of Solidity is used:
        - Version used: ['>=0.8.0<=0.8.9', '^0.8.0', '^0.8.2']
        - ^0.8.0 (@openzeppelin/contracts-upgradeable/token/ERC721/IERC721ReceiverUpgradeable.sol#4)
        - ^0.8.0 (@openzeppelin/contracts-upgradeable/token/ERC721/extensions/ERC721BurnableUpgradeable.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/proxy/beacon/UpgradeableBeacon.sol#4)
        - ^0.8.0 (@openzeppelin/contracts-upgradeable/token/ERC721/IERC721Upgradeable.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/access/Ownable.sol#4)
        - ^0.8.0 (@openzeppelin/contracts-upgradeable/utils/CountersUpgradeable.sol#4)
        - ^0.8.0 (@openzeppelin/contracts-upgradeable/token/ERC721/extensions/ERC721URIStorageUpgradeable.sol#4)
        - ^0.8.0 (@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#4)
        - ^0.8.0 (@openzeppelin/contracts-upgradeable/proxy/beacon/IBeaconUpgradeable.sol#4)
        - ^0.8.0 (@openzeppelin/contracts-upgradeable/token/ERC20/extensions/IERC20MetadataUpgradeable.sol#4)
        - ^0.8.0 (@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#4)
        - ^0.8.0 (@openzeppelin/contracts-upgradeable/utils/cryptography/ECDSAUpgradeable.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/proxy/beacon/IBeacon.sol#4)
        - >=0.8.0<=0.8.9 (TokenFactory/factories/ERC721TokenFactory.sol#3)
        - ^0.8.0 (@openzeppelin/contracts/utils/StorageSlot.sol#4)
        - ^0.8.0 (@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#4)
        - ^0.8.2 (@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#4)
        - ^0.8.2 (TokenFactory/tokens/ERC721.sol#2)
        - ^0.8.0 (@openzeppelin/contracts/utils/introspection/ERC165.sol#4)
        - >=0.8.0<=0.8.9 (Exchange/exchange.sol#2)
        - ^0.8.0 (@openzeppelin/contracts-upgradeable/access/IAccessControlUpgradeable.sol#4)
        - ^0.8.0 (@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#4)
        - ^0.8.2 (@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/proxy/Proxy.sol#4)
        - ^0.8.0 (@openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol#4)
        - ^0.8.0 (TokenFactory/factories/ERC721EKTABeaconProxy.sol#3)
        - ^0.8.0 (@openzeppelin/contracts/access/IAccessControl.sol#4)
        - ^0.8.0 (@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/access/AccessControl.sol#4)
        - ^0.8.0 (@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/proxy/beacon/BeaconProxy.sol#4)
        - ^0.8.0 (@openzeppelin/contracts-upgradeable/utils/introspection/IERC165Upgradeable.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/utils/Address.sol#4)
        - ^0.8.0 (@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/utils/introspection/IERC165.sol#4)
        - ^0.8.0 (@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#4)
        - ^0.8.0 (@openzeppelin/contracts-upgradeable/utils/StringsUpgradeable.sol#4)
        - ^0.8.0 (@openzeppelin/contracts/utils/Context.sol#4)
        - ^0.8.0 (@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#4)
        - ^0.8.0 (@openzeppelin/contracts-upgradeable/token/ERC721/extensions/IERC721MetadataUpgradeable.sol#4)
        - ^0.8.0 (@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol#4)
        - ^0.8.0 (TokenFactory/factories/ERC721EKTABeacon.sol#3)
        - ^0.8.0 (@openzeppelin/contracts/security/Pausable.sol#4)
        - ^0.8.0 (@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#4)
```



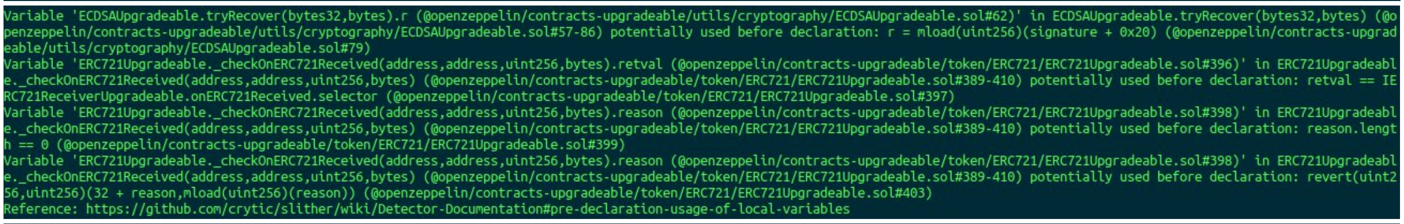
```
ECDSAUpgradeable.tryRecover(bytes32,bytes) (@openzeppelin/contracts-upgradeable/utils/cryptography/ECDSAUpgradeable.sol#57-86) uses assembly
        - INLINE ASM (@openzeppelin/contracts-upgradeable/utils/cryptography/ECDSAUpgradeable.sol#67-71)
        - INLINE ASM (@openzeppelin/contracts-upgradeable/utils/cryptography/ECDSAUpgradeable.sol#78-81)
ECDSAUpgradeable.tryRecover(bytes32,bytes32,bytes32) (@openzeppelin/contracts-upgradeable/utils/cryptography/ECDSAUpgradeable.sol#115-127) uses assembly
        - INLINE ASM (@openzeppelin/contracts-upgradeable/utils/cryptography/ECDSAUpgradeable.sol#122-125)
ERC721Factory.deploy(bytes,uint256) (TokenFactory/factories/ERC721TokenFactory.sol#91-97) uses assembly
        - INLINE ASM (TokenFactory/factories/ERC721TokenFactory.sol#93-95)
ERC721Factory.deployProxy(bytes,uint256) (TokenFactory/factories/ERC721TokenFactory.sol#111-119) uses assembly
        - INLINE ASM (TokenFactory/factories/ERC721TokenFactory.sol#113-118)
StorageSlot.getAddressSlot(bytes32) (@openzeppelin/contracts/utils/StorageSlot.sol#52-56) uses assembly
        - INLINE ASM (@openzeppelin/contracts/utils/StorageSlot.sol#53-55)
StorageSlot.getBooleanSlot(bytes32) (@openzeppelin/contracts/utils/StorageSlot.sol#61-65) uses assembly
        - INLINE ASM (@openzeppelin/contracts/utils/StorageSlot.sol#62-64)
StorageSlot.getBytes32Slot(bytes32) (@openzeppelin/contracts/utils/StorageSlot.sol#70-74) uses assembly
        - INLINE ASM (@openzeppelin/contracts/utils/StorageSlot.sol#71-73)
StorageSlot.getUint256Slot(bytes32) (@openzeppelin/contracts/utils/StorageSlot.sol#79-83) uses assembly
        - INLINE ASM (@openzeppelin/contracts/utils/StorageSlot.sol#80-82)
AddressUpgradeable.isContract(address) (@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#27-37) uses assembly
        - INLINE ASM (@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#33-35)
AddressUpgradeable.verifyCallResult(bool,bytes,string) (@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#169-189) uses assembly
        - INLINE ASM (@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#181-184)
Proxy._delegate(address) (@openzeppelin/contracts/proxy/Proxy.sol#22-45) uses assembly
        - INLINE ASM (@openzeppelin/contracts/proxy/Proxy.sol#23-44)
ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes) (@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#389-410) uses assembly
        - INLINE ASM (@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#402-404)
Address.isContract(address) (@openzeppelin/contracts/utils/Address.sol#27-37) uses assembly
        - INLINE ASM (@openzeppelin/contracts/utils/Address.sol#33-35)
Address.verifyCallResult(bool,bytes,string) (@openzeppelin/contracts/utils/Address.sol#196-216) uses assembly
        - INLINE ASM (@openzeppelin/contracts/utils/Address.sol#208-211)
StorageSlotUpgradeable.getAddressSlot(bytes32) (@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#52-56) uses assembly
        - INLINE ASM (@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#53-55)
StorageSlotUpgradeable.getBooleanSlot(bytes32) (@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#61-65) uses assembly
        - INLINE ASM (@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#62-64)
StorageSlotUpgradeable.getBytes32Slot(bytes32) (@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#70-74) uses assembly
        - INLINE ASM (@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#71-73)
StorageSlotUpgradeable.getUint256Slot(bytes32) (@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#79-83) uses assembly
        - INLINE ASM (@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#80-82)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```



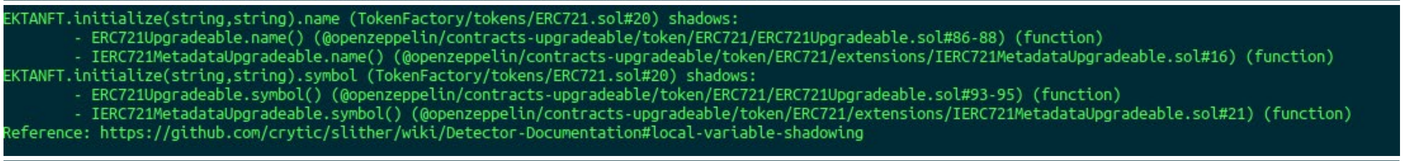
```
Reentrancy in ERC1967UpgradeUpgradeable._upgradeToAndCallSecure(address,bytes,bool) (@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#87-115):
        External calls:
        - _functionDelegateCall(newImplementation,data) (@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#97)
                - (success,returndata) = target.delegatecall(data) (@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#212)
        - _functionDelegateCall(newImplementation,abi.encodeWithSignature(upgradeTo(address),oldImplementation)) (@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradea
ble.sol#105-108)
                - (success,returndata) = target.delegatecall(data) (@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#212)
        Event emitted after the call(s):
        - Upgraded(newImplementation) (@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#63)
                - _upgradeTo(newImplementation) (@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#113)
Reentrancy in ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#80-108):
        External calls:
        - Address.functionDelegateCall(newImplementation,data) (@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#90)
        - Address.functionDelegateCall(newImplementation,abi.encodeWithSignature(upgradeTo(address),oldImplementation)) (@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#98-101)
        Event emitted after the call(s):
        - Upgraded(newImplementation) (@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#56)
                - _upgradeTo(newImplementation) (@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#106)
Reentrancy in Exchange.completeBidding(Exchange.BidOrder,bytes) (Exchange/exchange.sol#190-255):
        External calls:
        - IWETH(WEKTA).transferFrom(bidOrder.buyer,msg.sender,bidOrder.bidAmount) (Exchange/exchange.sol#233-237)
        - IERC721Upgradeable(bidOrder.tokenAddress).transferFrom(msg.sender,bidOrder.buyer,bidOrder.tokenId) (Exchange/exchange.sol#240-244)
        Event emitted after the call(s):
        - Buy(bidOrder.seller,bidOrder.buyer,bidOrder.tokenAddress,bidOrder.tokenId) (Exchange/exchange.sol#249-254)
Reentrancy in Exchange.completeFixedSale(Exchange.Order) (Exchange/exchange.sol#140-179):
        External calls:
        - (success) = address(order.seller).call{value: order.amount}() (Exchange/exchange.sol#165)
        - IERC721Upgradeable(order.tokenAddress).transferFrom(order.seller,msg.sender,order.tokenId) (Exchange/exchange.sol#169-173)
        External calls sending eth:
        - (success) = address(order.seller).call{value: order.amount}() (Exchange/exchange.sol#165)
        Event emitted after the call(s):
        - Buy(order.seller,msg.sender,order.tokenAddress,order.tokenId) (Exchange/exchange.sol#178)
Reentrancy in ERC721Factory.createToken(string,string,string,uint256) (TokenFactory/factories/ERC721TokenFactory.sol#50-56):
        External calls:
        - token.updateBaseURI(_baseUri) (TokenFactory/factories/ERC721TokenFactory.sol#53)
        - token.transferOwnership(msg.sender) (TokenFactory/factories/ERC721TokenFactory.sol#54)
        Event emitted after the call(s):
        - Create721(address(token)) (TokenFactory/factories/ERC721TokenFactory.sol#55)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```



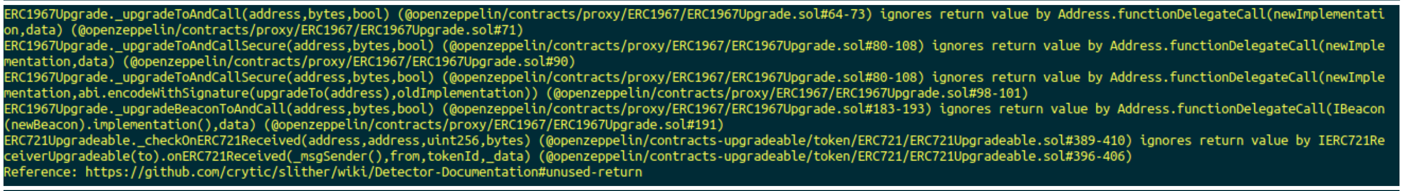
```
Reentrancy in EKTANFT.safeMint(address,string) (TokenFactory/tokens/ERC721.sol#38-43):
        External calls:
        - _safeMint(to,tokenId) (TokenFactory/tokens/ERC721.sol#41)
                - IERC721ReceiverUpgradeable(to).onERC721Received(_msgSender(),from,tokenId,_data) (@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#396-406)
        State variables written after the call(s):
        - _setTokenURI(tokenId,uri) (TokenFactory/tokens/ERC721.sol#42)
                - _tokenURIs[tokenId] = _tokenURI (@openzeppelin/contracts-upgradeable/token/ERC721/extensions/ERC721URIStorageUpgradeable.sol#56)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```



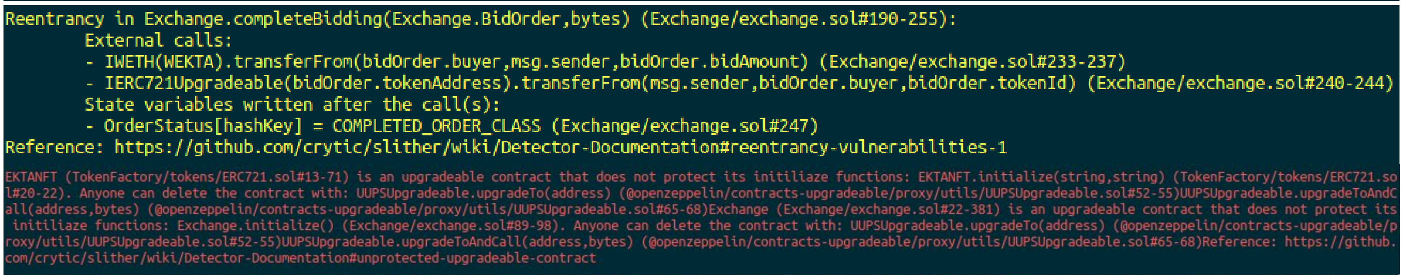
```
Variable 'ECDSAUpgradeable.tryRecover(bytes32,bytes).r (@openzeppelin/contracts-upgradeable/utils/cryptography/ECDSAUpgradeable.sol#62)' in ECDSAUpgradeable.tryRecover(bytes32,bytes) (@o
penzeppelin/contracts-upgradeable/utils/cryptography/ECDSAUpgradeable.sol#57-86) potentially used before declaration: r = mload(uint256)(signature + 0x20) (@openzeppelin/contracts-upgrad
eable/utils/cryptography/ECDSAUpgradeable.sol#79)
Variable 'ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes).retval (@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#396)' in ERC721Upgradeabl
e._checkOnERC721Received(address,address,uint256,bytes) (@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#389-410) potentially used before declaration: retval == IE
RC721ReceiverUpgradeable.onERC721Received.selector (@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#397)
Variable 'ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes).reason (@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#398)' in ERC721Upgradeabl
e._checkOnERC721Received(address,address,uint256,bytes) (@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#389-410) potentially used before declaration: reason.lengt
h == 0 (@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#399)
Variable 'ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes).reason (@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#398)' in ERC721Upgradeabl
e._checkOnERC721Received(address,address,uint256,bytes) (@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#389-410) potentially used before declaration: revert(uint2
56,uint256)(32 + reason,mload(uint256)(reason)) (@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#403)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables
```



```
EKTANFT.initialize(string,string).name (TokenFactory/tokens/ERC721.sol#20) shadows:
        - ERC721Upgradeable.name() (@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#86-88) (function)
        - IERC721MetadataUpgradeable.name() (@openzeppelin/contracts-upgradeable/token/ERC721/extensions/IERC721MetadataUpgradeable.sol#16) (function)
EKTANFT.initialize(string,string).symbol (TokenFactory/tokens/ERC721.sol#20) shadows:
        - ERC721Upgradeable.symbol() (@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#93-95) (function)
        - IERC721MetadataUpgradeable.symbol() (@openzeppelin/contracts-upgradeable/token/ERC721/extensions/IERC721MetadataUpgradeable.sol#21) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
```

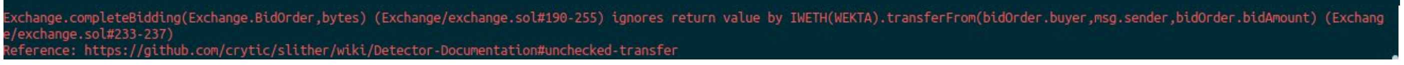


```
ERC1967Upgrade._upgradeToAndCall(address,bytes,bool) (@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#64-73) ignores return value by Address.functionDelegateCall(newImplementati
on,data) (@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#71)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#80-108) ignores return value by Address.functionDelegateCall(newImple
mentation,data) (@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#90)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#80-108) ignores return value by Address.functionDelegateCall(newImple
mentation,abi.encodeWithSignature(upgradeTo(address),oldImplementation)) (@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#98-101)
ERC1967Upgrade._upgradeBeaconToAndCall(address,bytes,bool) (@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#183-193) ignores return value by Address.functionDelegateCall(IBeacon
(newBeacon).implementation(),data) (@openzeppelin/contracts/proxy/ERC1967/ERC1967Upgrade.sol#191)
ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes) (@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#389-410) ignores return value by IERC721Re
ceiverUpgradeable(to).onERC721Received(_msgSender(),from,tokenId,_data) (@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#396-406)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

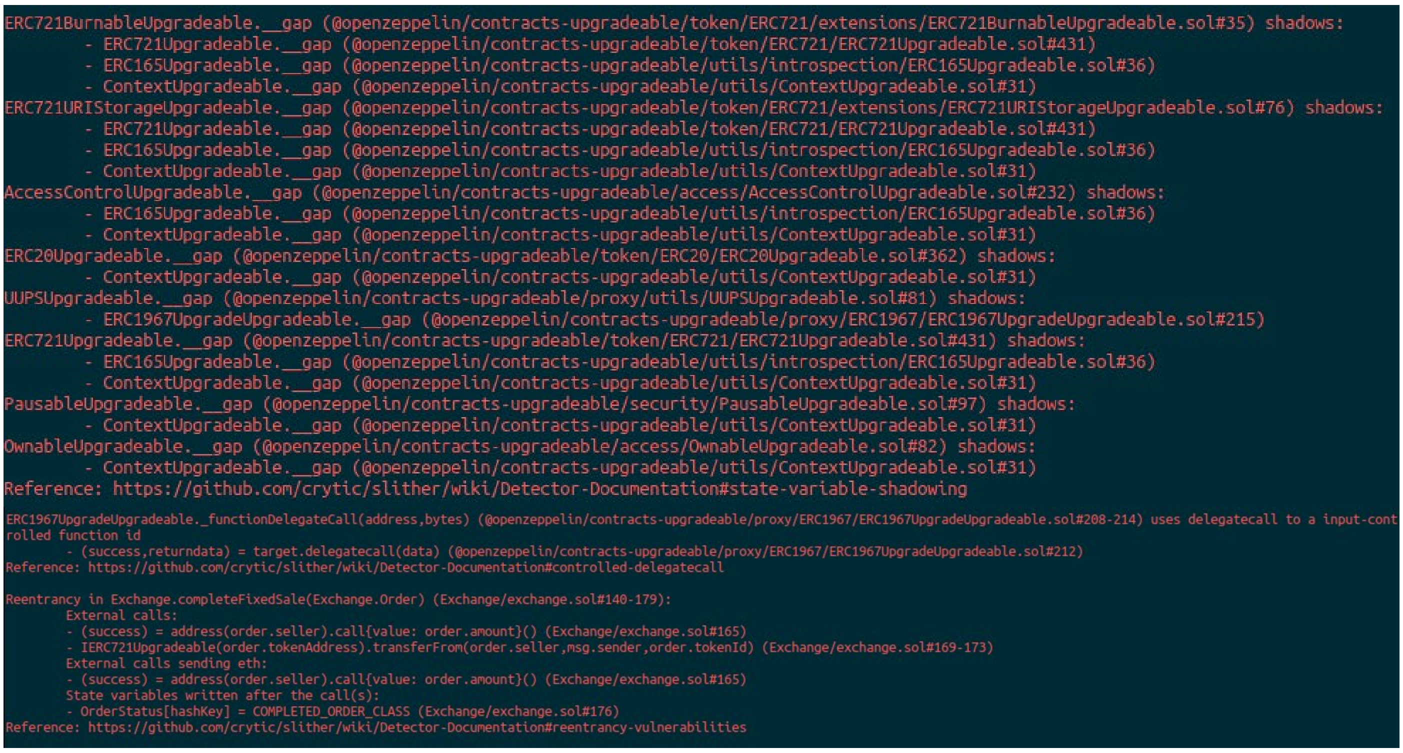


```
Reentrancy in Exchange.completeBidding(Exchange.BidOrder,bytes) (Exchange/exchange.sol#190-255):
        External calls:
        - IWETH(WEKTA).transferFrom(bidOrder.buyer,msg.sender,bidOrder.bidAmount) (Exchange/exchange.sol#233-237)
        - IERC721Upgradeable(bidOrder.tokenAddress).transferFrom(msg.sender,bidOrder.buyer,bidOrder.tokenId) (Exchange/exchange.sol#240-244)
        State variables written after the call(s):
        - OrderStatus[hashKey] = COMPLETED_ORDER_CLASS (Exchange/exchange.sol#247)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

EKTANFT (TokenFactory/tokens/ERC721.sol#13-71) is an upgradeable contract that does not protect its initiliaze functions: EKTANFT.initialize(string,string) (TokenFactory/tokens/ERC721.so
l#20-22). Anyone can delete the contract with: UUPSUpgradeable.upgradeTo(address) (@openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol#52-55)UUPSUpgradeable.upgradeToAndC
all(address,bytes) (@openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol#65-68)Exchange (Exchange/exchange.sol#22-381) is an upgradeable contract that does not protect its
 initiliaze functions: Exchange.initialize() (Exchange/exchange.sol#89-98). Anyone can delete the contract with: UUPSUpgradeable.upgradeTo(address) (@openzeppelin/contracts-upgradeable/p
roxy/utils/UUPSUpgradeable.sol#52-55)UUPSUpgradeable.upgradeToAndCall(address,bytes) (@openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol#65-68)Reference: https://github.
com/crytic/slither/wiki/Detector-Documentation#unprotected-upgradeable-contract
```



```
Exchange.completeBidding(Exchange.BidOrder,bytes) (Exchange/exchange.sol#190-255) ignores return value by IWETH(WEKTA).transferFrom(bidOrder.buyer,msg.sender,bidOrder.bidAmount) (Exchang
e/exchange.sol#233-237)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
```

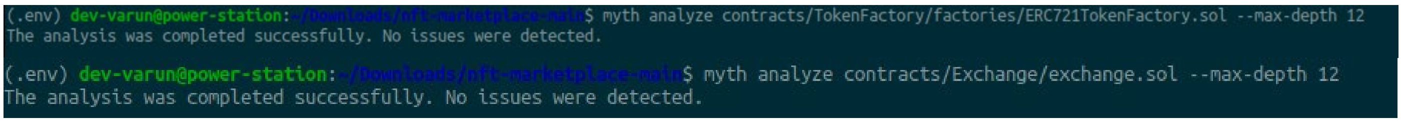


```
ERC721BurnableUpgradeable.__gap (@openzeppelin/contracts-upgradeable/token/ERC721/extensions/ERC721BurnableUpgradeable.sol#35) shadows:
        - ERC721Upgradeable.__gap (@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#431)
        - ERC165Upgradeable.__gap (@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#36)
        - ContextUpgradeable.__gap (@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#31)
ERC721URIStorageUpgradeable.__gap (@openzeppelin/contracts-upgradeable/token/ERC721/extensions/ERC721URIStorageUpgradeable.sol#76) shadows:
        - ERC721Upgradeable.__gap (@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#431)
        - ERC165Upgradeable.__gap (@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#36)
        - ContextUpgradeable.__gap (@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#31)
AccessControlUpgradeable.__gap (@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#232) shadows:
        - ERC165Upgradeable.__gap (@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#36)
        - ContextUpgradeable.__gap (@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#31)
ERC20Upgradeable.__gap (@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#362) shadows:
        - ContextUpgradeable.__gap (@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#31)
UUPSUpgradeable.__gap (@openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol#81) shadows:
        - ERC1967UpgradeUpgradeable.__gap (@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#215)
ERC721Upgradeable.__gap (@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#431) shadows:
        - ERC165Upgradeable.__gap (@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#36)
        - ContextUpgradeable.__gap (@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#31)
PausableUpgradeable.__gap (@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#97) shadows:
        - ContextUpgradeable.__gap (@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#31)
OwnableUpgradeable.__gap (@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#82) shadows:
        - ContextUpgradeable.__gap (@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#31)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing

ERC1967UpgradeUpgradeable._functionDelegateCall(address,bytes) (@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#208-214) uses delegatecall to a input-cont
rolled function id
        - (success,returndata) = target.delegatecall(data) (@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#212)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#controlled-delegatecall

Reentrancy in Exchange.completeFixedSale(Exchange.Order) (Exchange/exchange.sol#140-179):
        External calls:
        - (success) = address(order.seller).call{value: order.amount}() (Exchange/exchange.sol#165)
        - IERC721Upgradeable(order.tokenAddress).transferFrom(order.seller,msg.sender,order.tokenId) (Exchange/exchange.sol#169-173)
        External calls sending eth:
        - (success) = address(order.seller).call{value: order.amount}() (Exchange/exchange.sol#165)
        State variables written after the call(s):
        - OrderStatus[hashKey] = COMPLETED_ORDER_CLASS (Exchange/exchange.sol#176)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities
```

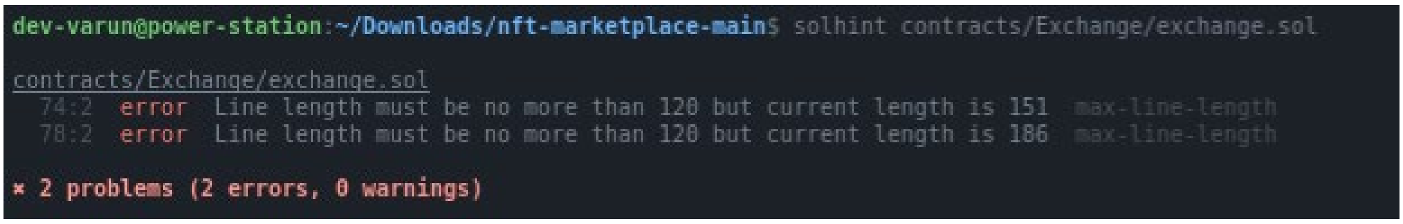
6.2 Mythril



```
(.env) dev-varun@power-station:~/Downloads/nft-marketplace-main$ myth analyze contracts/TokenFactory/factories/ERC721TokenFactory.sol --max-depth 12
The analysis was completed successfully. No issues were detected.

(.env) dev-varun@power-station:~/Downloads/nft-marketplace-main$ myth analyze contracts/Exchange/exchange.sol --max-depth 12
The analysis was completed successfully. No issues were detected.
```

6.3 Solhint

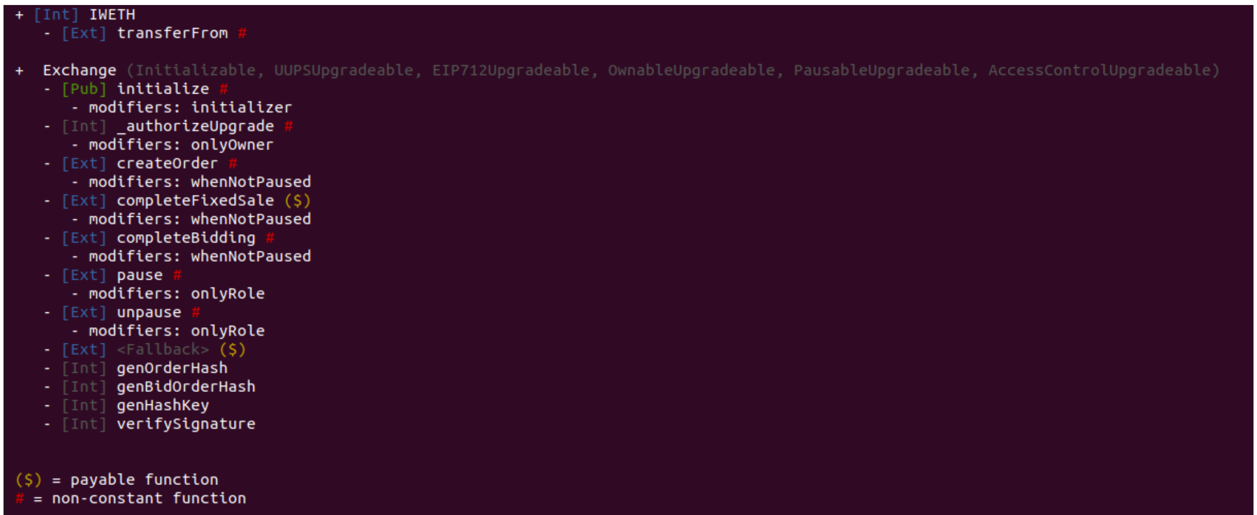


```
dev-varun@power-station:~/Downloads/nft-marketplace-main$ solhint contracts/Exchange/exchange.sol

contracts/Exchange/exchange.sol
  74:2  error  Line length must be no more than 120 but current length is 151  max-line-length
  78:2  error  Line length must be no more than 120 but current length is 186  max-line-length

✖ 2 problems (2 errors, 0 warnings)
```

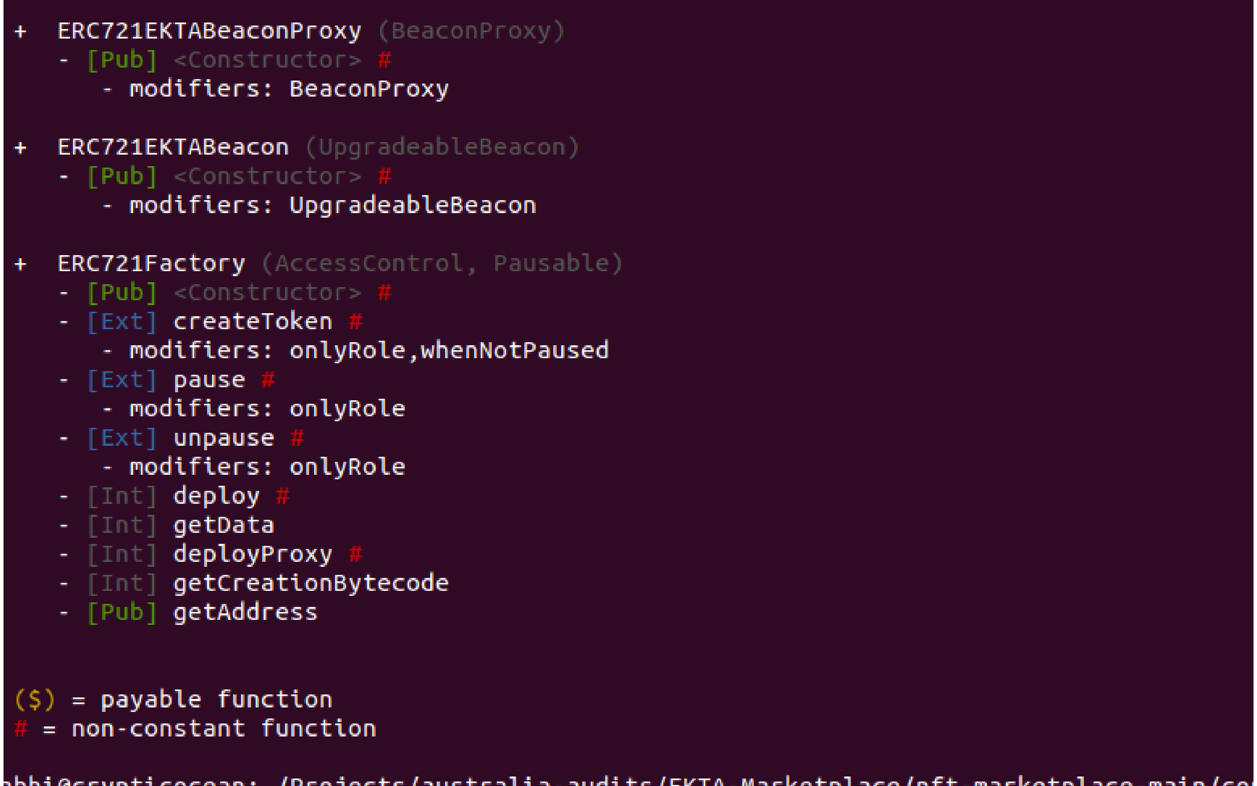
6.4 Surya



```
+ [Int] IWETH
   - [Ext] transferFrom #

+  Exchange (Initializable, UUPSUpgradeable, EIP712Upgradeable, OwnableUpgradeable, PausableUpgradeable, AccessControlUpgradeable)
   - [Pub] initialize #
      - modifiers: initializer
   - [Int] _authorizeUpgrade #
      - modifiers: onlyOwner
   - [Ext] createOrder #
      - modifiers: whenNotPaused
   - [Ext] completeFixedSale ($)
      - modifiers: whenNotPaused
   - [Ext] completeBidding #
      - modifiers: whenNotPaused
   - [Ext] pause #
      - modifiers: onlyRole
   - [Ext] unpause #
      - modifiers: onlyRole
   - [Ext] <Fallback> ($)
   - [Int] genOrderHash
   - [Int] genBidOrderHash
   - [Int] genHashKey
   - [Int] verifySignature


($) = payable function
# = non-constant function
```



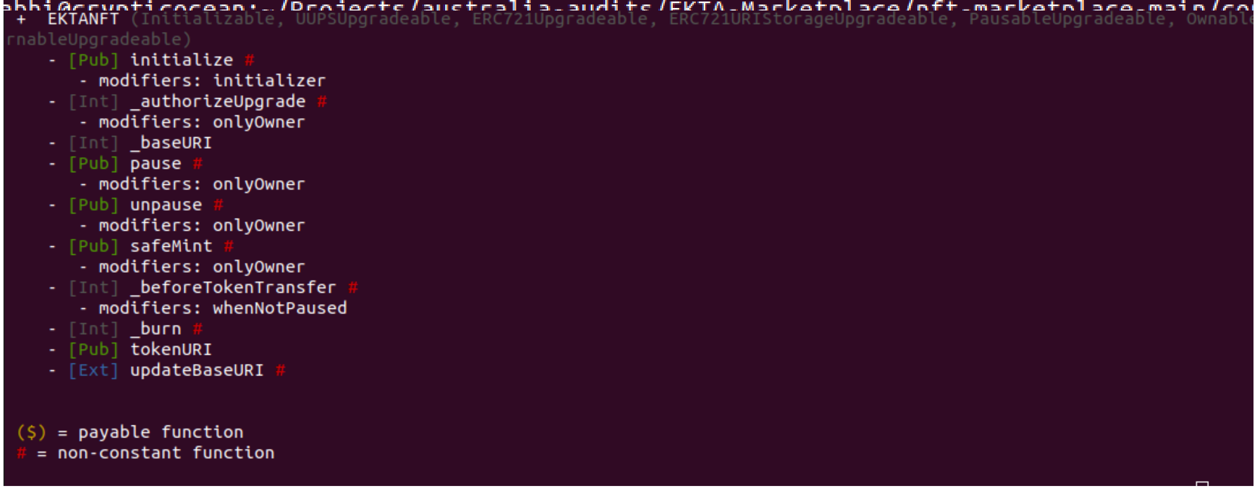
```
+  ERC721EKTABeaconProxy (BeaconProxy)
   - [Pub] <Constructor> #
      - modifiers: BeaconProxy

+  ERC721EKTABeacon (UpgradeableBeacon)
   - [Pub] <Constructor> #
      - modifiers: UpgradeableBeacon

+  ERC721Factory (AccessControl, Pausable)
   - [Pub] <Constructor> #
   - [Ext] createToken #
      - modifiers: onlyRole,whenNotPaused
   - [Ext] pause #
      - modifiers: onlyRole
   - [Ext] unpause #
      - modifiers: onlyRole
   - [Int] deploy #
   - [Int] getData
   - [Int] deployProxy #
   - [Int] getCreationBytecode
   - [Pub] getAddress


($) = payable function
# = non-constant function

abhi@crypticocean:~/Projects/australia-audits/EKTA-Marketplace/nft-marketplace-main/co
```



```
 +  EKTANFT (Initializable, UUPSUpgradeable, ERC721Upgradeable, ERC721URIStorageUpgradeable, PausableUpgradeable, Ownable
rnableUpgradeable)
    - [Pub] initialize #
       - modifiers: initializer
    - [Int] _authorizeUpgrade #
       - modifiers: onlyOwner
    - [Int] _baseURI
    - [Pub] pause #
       - modifiers: onlyOwner
    - [Pub] unpause #
       - modifiers: onlyOwner
    - [Pub] safeMint #
       - modifiers: onlyOwner
    - [Int] _beforeTokenTransfer #
       - modifiers: whenNotPaused
    - [Int] _burn #
    - [Pub] tokenURI
    - [Ext] updateBaseURI #


($) = payable function
# = non-constant function
```

7.0 Auditing Approach and Methodologies applied

Throughout the audit of the smart contract, care was taken to ensure:

- Overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behavior.
- Token distribution and calculations are as per intended behavior mentioned in the whitepaper.
- Implementation of token standards.
- Efficient use of gas.
- Code is safe from Re-entrancy and other vulnerabilities.

A combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy regarding the scope of the smart contract audit. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of smart contracts and can quickly identify items that do not follow security best practices. The following phases and associated tools were used throughout the term of the audit:

7.1 Structural Analysis

In this step we have analysed the design patterns and structure of all smart contracts. A thorough check was completed to ensure all Smart contracts are structured in a way that will not result in future problems.

7.2 Static Analysis

Static Analysis of smart contracts was undertaken to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

7.3 Code Review / Manual Analysis

Manual Analysis or review of done to identify new vulnerabilities or to verify the vulnerabilities found during the Static Analysis. The contracts were completely manually analysed, and their logic was checked and compared with the one described in the whitepaper. It should also be noted that the results of the automated analysis were verified manually.

7.4 Gas Consumption

In this step, we checked the behaviour of all smart contracts in production. Checks were completed to understand how much gas gets consumed, along with the possibilities of optimisation of code to reduce gas consumption.

7.5 Tools & Platforms Used For Audit

VSCode, Remix IDE, Truffle, Truffle Team, Ganache, Solhint, Mythril, Manticore, Slither.

7.6 Checked Vulnerabilities

We have scanned Ekta smart contracts for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that we considered:

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- ERC-20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- Unsafe type inference
- Implicit visibility level

8.0 Limitations on Disclosure and Use of this Report

This report contains information concerning potential details of Ekta and methods for exploiting them. Entersoft recommends that special precautions be taken to protect the confidentiality of both this document and the information contained herein. Security Assessment is an uncertain process, based on past experiences, currently available information, and known threats. All information security systems, which by their nature are dependent on human beings, are vulnerable to some degree. Therefore, while Entersoft considers the major security vulnerabilities of the analyzed systems to have been identified, there can be no assurance that any exercise of this nature will identify all possible vulnerabilities or propose exhaustive and operationally viable recommendations to mitigate those exposures. In addition, the analysis set forth herein is based on the technologies and known threats as of the date of this report. As technologies and risks change over time, the vulnerabilities associated with the operation of the Smart Contract described in this report, as well as the actions necessary to reduce the exposure to such vulnerabilities will also change. Entersoft makes no undertaking to supplement or update this report based on changed circumstances or facts of which Entersoft becomes aware after the date hereof, absent a specific written agreement to perform the supplemental or updated analysis. This report may recommend that Entersoft use certain software or hardware products manufactured or maintained by other vendors. Entersoft bases these recommendations upon its prior experience with the capabilities of those products. Nonetheless, Entersoft does not and cannot warrant that a particular product will work as advertised by the vendor, nor that it will operate in the manner intended. This report was prepared by Entersoft for the exclusive benefit of Ekta and is proprietary information. The Non-Disclosure Agreement (NDA) in effect between Entersoft and Ekta governs the disclosure of this report to all other parties including product vendors and suppliers.