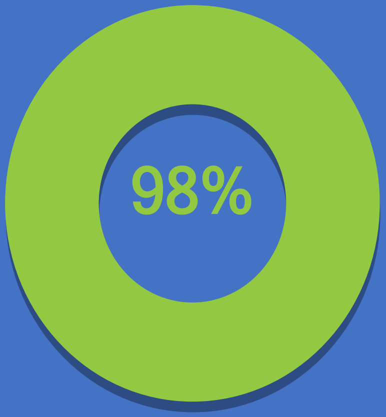
SMART CONTRACT SECURITY AUDIT REPORT

SHELDON

NOVEMBER 13, 2022

Anisimov Vladyslav



PASS

I have concluded that this smart contract passes security qualifications and bear no security or operational risk

Technical Summary

With this report, I have tried to ensure the reliability of the smart contract security by completing the assessment of their system’s architecture and smart contract codebase.

Auditing approach and Methodologies applied

In this audit, I consider the following crucial features of the code.

- Whether the code is secure.
- Whether the code meets the best coding practices.
- Whether the code meets the SWC Registry issue.
- Dos attacks
- Smart Contracts with no upgrade options
- Function default

The audit has been performed according to the following procedure:

❖ *Manual audit*

1. Inspecting the code line by line and revert the initial algorithms of the contracts and then compare them with the specification
2. Manually analyzing the code for security vulnerabilities.
3. Assessing the overall project structure, complexity & quality.
4. Checking SWC Registry issues in the code.
5. Unit testing by writing custom unit testing for each function.
6. Analysis of security on-chain data.

❖ *Automated analysis*

1. Scanning the project's code base with Slither.
2. Manually verifying (reject or confirm) all the issues found by tools.
3. Performing Unit testing.
4. Running the tests and checking their coverage.

Report: All the gathered information is described in this report.

△ Overview

Report Items:

No	Category	Description	Status
1	Basic Coding Bugs	❖ Constructor Mismatch ❖ Ownership Takeover ❖ Redundant Fallback Function ❖ Overflows & Underflows ❖ Reentrancy Money-Giving Bug ❖ Black hole ❖ Unauthorized Self-Destruct ❖ Revert DoS ❖ Unchecked External Call ❖ Gasless Send ❖ Send Instead of Transfer ❖ Costly Loop ❖ (Unsafe) Use of Untrusted Libraries ❖ (Unsafe) Use of Predictable Variables	Passed

		❖ Transaction Ordering Dependence ❖ Deprecated Uses	
2	Semantic Consistency Checks	Semantic Consistency Checks	Passed
3	Advanced Contract Scrutiny	❖ Business Logics Review ❖ Functionality Checks ❖ Authentication Management ❖ Access Control & Authorization ❖ Oracle Security ❖ Digital Asset Escrow ❖ Kill-Switch Mechanism ❖ Operation Trails & Event Generation ❖ ERC20 Idiosyncrasies Handling ❖ Frontend-Contract Integration ❖ Deployment Consistency ❖ Holistic Risk Management	Passed
4	Additional Recommendations	❖ Avoiding Use of Variadic Byte Array ❖ Using Fixed Compiler Version ❖ Making Visibility Level Explicit ❖ Making Type Inference Explicit ❖ Adhering To Function Declaration Strictly ❖ Following Other Best Practices	Passed
5	Configuration	Weaknesses in this category are typically introduced during the configuration of the software.	Passed
6	Data Processing Issues	Weaknesses in this category are typically found in functionality that processes data.	Passed

7	Numeric Errors	Weaknesses in this category are related to improper calculation or conversion of numbers.	Passed
8	Security Features	Weaknesses in this category are concerned with topics like authentication, access control, confidentiality, cryptography, and privilege management. (Software security is not security software.)	Passed
9	Time and State	Weaknesses in this category are related to the improper management of time and state in an environment that supports simultaneous or near-simultaneous computation by multiple systems, processes, or threads.	Passed
10	Error Conditions, Return Values, Status Codes	Weaknesses in this category include weaknesses that occur if a function does not generate the correct return/status code, or if the application does not handle all possible return/status codes that could be generated by a function.	Passed
11	Resource Management	Weaknesses in this category are related to improper management of system resources.	Passed
12	Behavioral Issue	Weaknesses in this category are related to unexpected behaviors from code that an application uses.	Passed
13	Business Logics	Weaknesses in this category identify some of the underlying problems that commonly allow attackers to manipulate the business logic of an application. Errors in business logic	Passed

		can be devastating to an entire application.	
14	Initialization and Cleanup	Weaknesses in this category occur in behaviors that are used for initialization and breakdown.	Passed
15	Arguments and Parameters	Weaknesses in this category are related to improper use arguments or parameters within function calls.	Passed
16	Expression Issues	Weaknesses in this category are related to incorrectly written expressions within code.	Passed
17	Coding Practices	Weaknesses in this category are related to coding practices that are deemed unsafe and increase the chances that an ex pilotable vulnerability will be present in the application. They may not directly introduce a vulnerability, but indicate the product has not been carefully developed or maintained.	Passed

The vulnerability severity level information:

No	Level	Description
1	Critical	Critical severity vulnerabilities will have a significant effect on the security of smart contract, and it is strongly recommended to fix the critical vulnerabilities.

2	High	High severity vulnerabilities will affect the normal operation of the smart contract. It is strongly recommended to fix high-risk vulnerabilities
3	Medium	Medium severity vulnerability will affect the operation of the smart contract. It is recommended to fix medium-risk vulnerabilities
4	Low	Low severity vulnerabilities may affect the operation of the smart contract in certain scenarios. It is suggested that the project party should evaluate and consider whether these vulnerabilities need to be fixed.
5	Lowest	There are safety risks theoretically, but it is extremely difficult to reproduce in engineering.

△ Audit Result

- Scope

- Exchange.sol

  Handles execution of buy and sell orders, domain and currency transfers. Stores the cancellation data.

- Leasing.sol

Handles execution of leasing orders and leasing extensions. Acts as a proxy for the lessee to control their leased domains. Handles leasing pricing calculation - initial period, initial period price, monthly fee, yearly increase.

- RoyaltySplitter.sol

  Handles the distribution of the royalties from above contracts.

  5% - Martin and Rick - 0xbec155ad45b1ba9f0cca374e56681a414320b322

  5% - Ryno and Munz - 0x261c60a2C0AAf92403A7457F112654f1db95Dd98

  5% - Muharrem and team - 0xEeBE3E885097E70E123EAc7296aEDFF8824Aa999

  5% - Wilfred and team - 0xE0949A5aa405229A8F3ED8C950721F97e85526fB

  20% - Jake - 0x1d266890d5Dabd4B5Fe61F9a45e3ae3d5529fd8c

  52% - Sheldon - 0xC13f74C0EE0e75716AA4A59914e32b9bF3F304A8

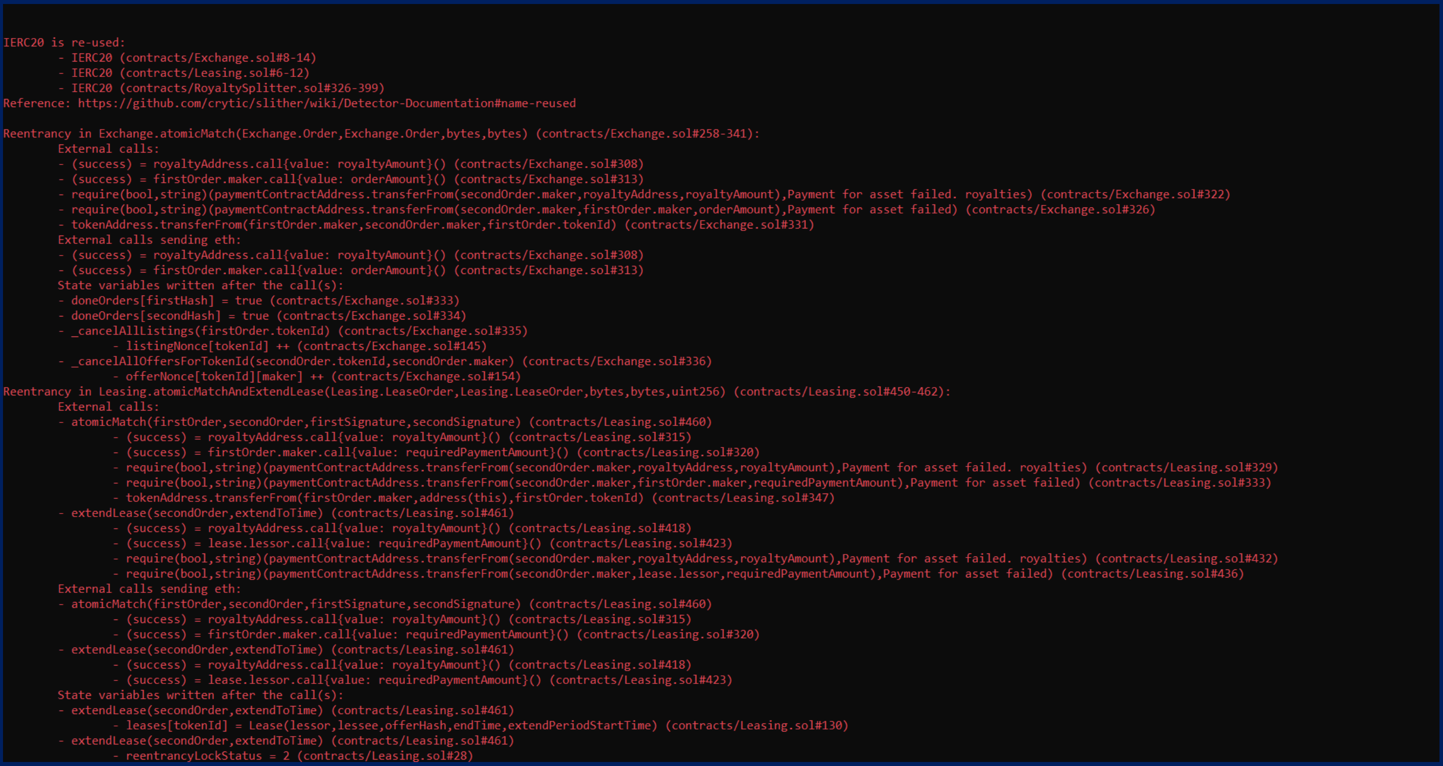
  8% - corporate wallet - 0x15F3A86938d8a5207605B205cc1A801F5bAB8F0c

- IChildRegistry.sol
- IDataReader.sol
- IERC1967.sol
- IMintingManager.sol
- IRecordReader.sol
- IRecordStorage.sol
- IRegistryReader.sol
- IReverseRegistry.sol
- IRootRegistry.sol
- IUNSRegistry.sol
- IChildToken.sol
- IMintableERC721.sol

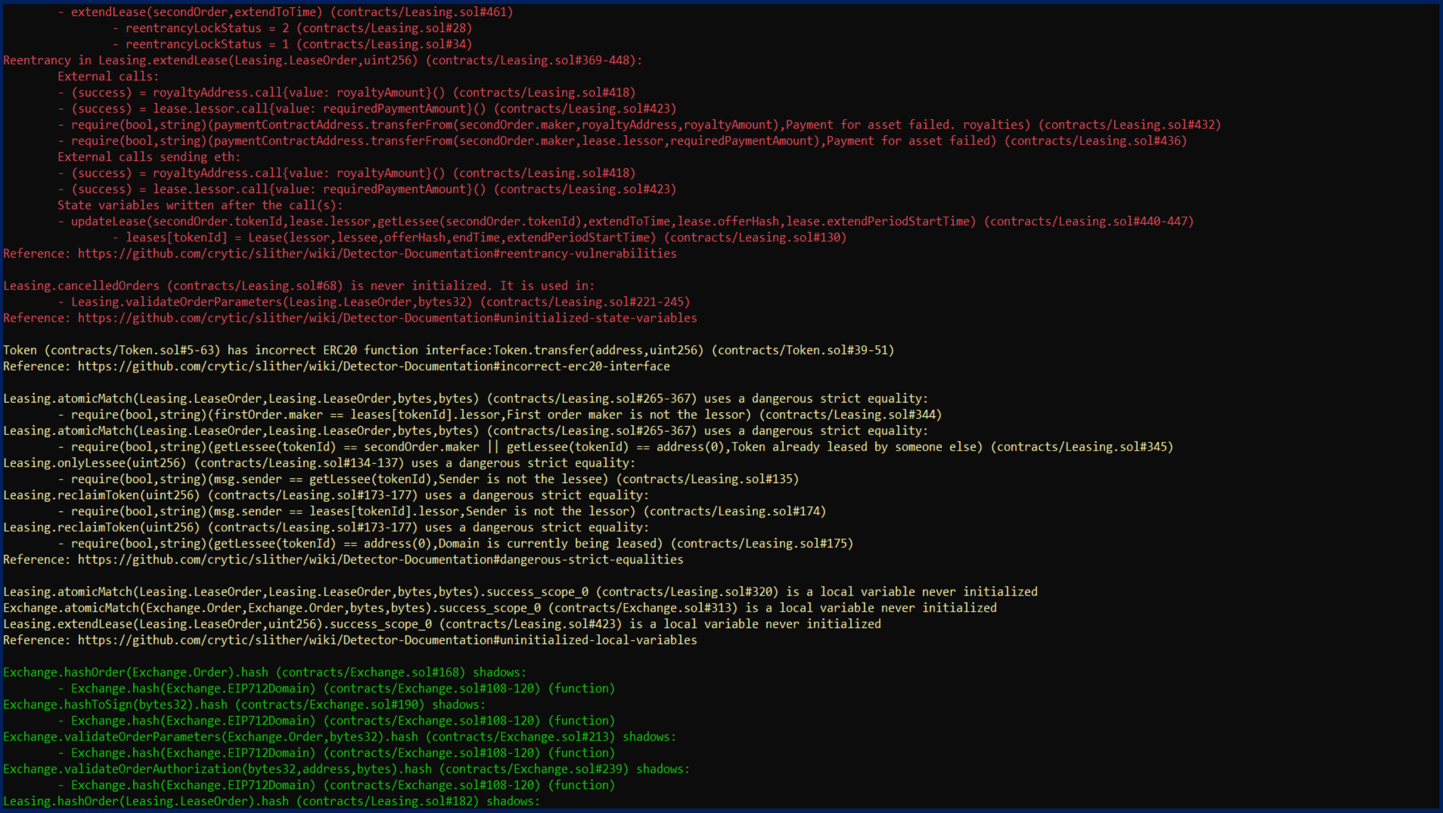
-Report Result

1.Slither Tool

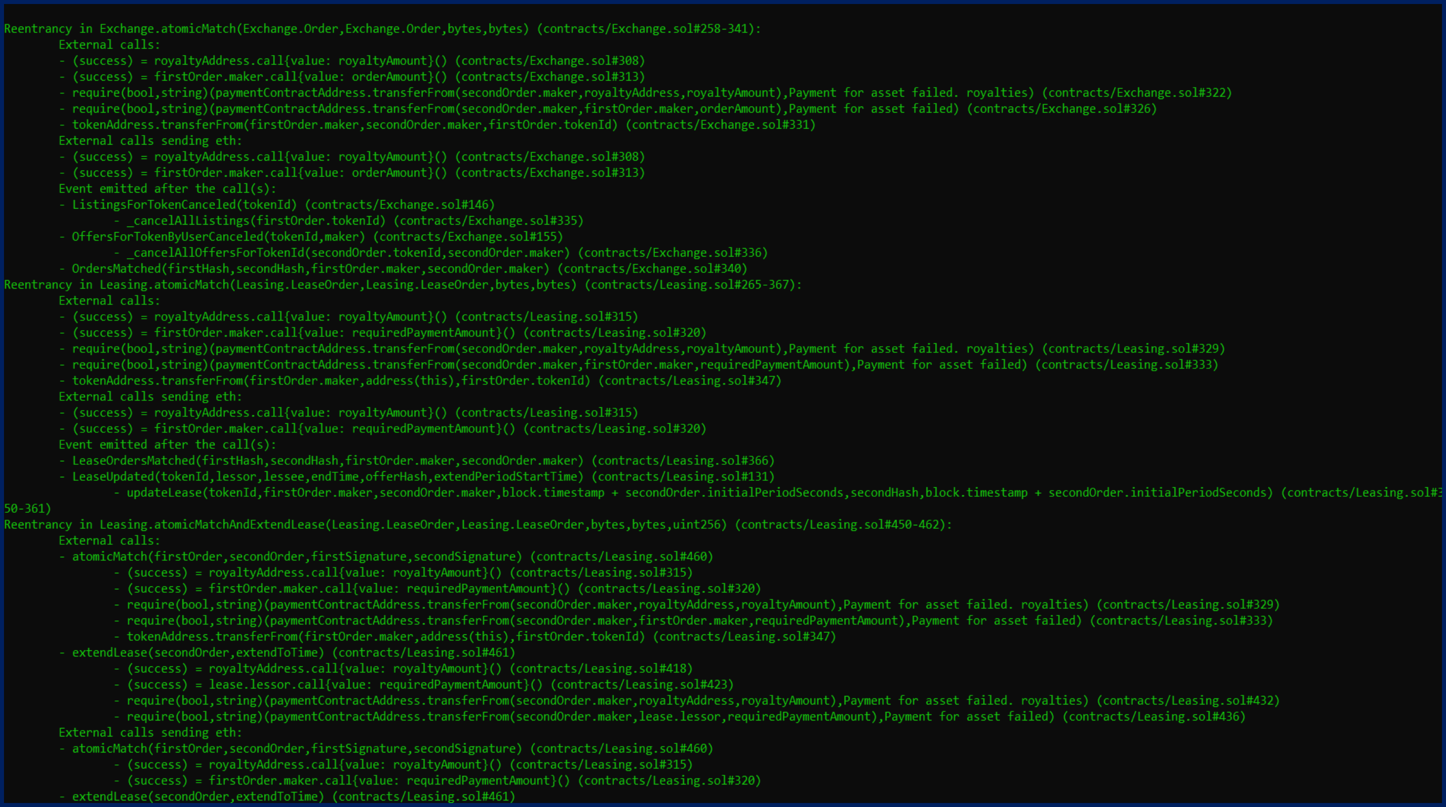
Result as follow:



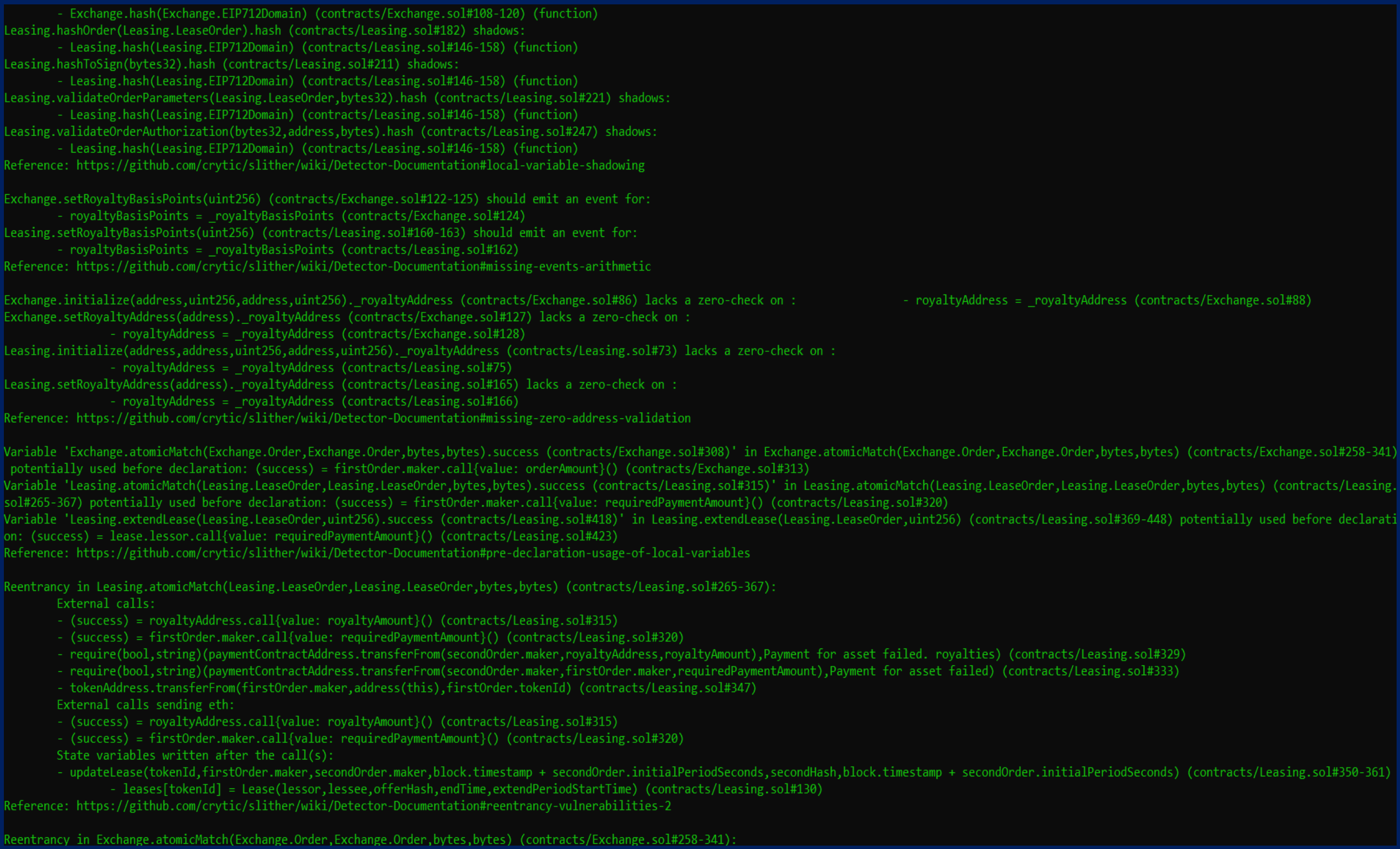
```
IERC20 is re-used:
        - IERC20 (contracts/Exchange.sol#8-14)
        - IERC20 (contracts/Leasing.sol#6-12)
        - IERC20 (contracts/RoyaltySplitter.sol#326-399)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#name-reused

Reentrancy in Exchange.atomicMatch(Exchange.Order,Exchange.Order,bytes,bytes) (contracts/Exchange.sol#258-341):
        External calls:
        - (success) = royaltyAddress.call{value: royaltyAmount}() (contracts/Exchange.sol#308)
        - (success) = firstOrder.maker.call{value: orderAmount}() (contracts/Exchange.sol#313)
        - require(bool,string)(paymentContractAddress.transferFrom(secondOrder.maker,royaltyAddress,royaltyAmount),Payment for asset failed. royalties) (contracts/Exchange.sol#322)
        - require(bool,string)(paymentContractAddress.transferFrom(secondOrder.maker,firstOrder.maker,orderAmount),Payment for asset failed) (contracts/Exchange.sol#326)
        - tokenAddress.transferFrom(firstOrder.maker,secondOrder.maker,firstOrder.tokenId) (contracts/Exchange.sol#331)
        External calls sending eth:
        - (success) = royaltyAddress.call{value: royaltyAmount}() (contracts/Exchange.sol#308)
        - (success) = firstOrder.maker.call{value: orderAmount}() (contracts/Exchange.sol#313)
        State variables written after the call(s):
        - doneOrders[firstHash] = true (contracts/Exchange.sol#333)
        - doneOrders[secondHash] = true (contracts/Exchange.sol#334)
        - _cancelAllListings(firstOrder.tokenId) (contracts/Exchange.sol#335)
                - listingNonce[tokenId] ++ (contracts/Exchange.sol#145)
        - _cancelAllOffersForTokenId(secondOrder.tokenId,secondOrder.maker) (contracts/Exchange.sol#336)
                - offerNonce[tokenId][maker] ++ (contracts/Exchange.sol#154)
Reentrancy in Leasing.atomicMatchAndExtendLease(Leasing.LeaseOrder,Leasing.LeaseOrder,bytes,bytes,uint256) (contracts/Leasing.sol#450-462):
        External calls:
        - atomicMatch(firstOrder,secondOrder,firstSignature,secondSignature) (contracts/Leasing.sol#460)
                - (success) = royaltyAddress.call{value: royaltyAmount}() (contracts/Leasing.sol#315)
                - (success) = firstOrder.maker.call{value: requiredPaymentAmount}() (contracts/Leasing.sol#320)
                - require(bool,string)(paymentContractAddress.transferFrom(secondOrder.maker,royaltyAddress,royaltyAmount),Payment for asset failed. royalties) (contracts/Leasing.sol#329)
                - require(bool,string)(paymentContractAddress.transferFrom(secondOrder.maker,firstOrder.maker,requiredPaymentAmount),Payment for asset failed) (contracts/Leasing.sol#333)
                - tokenAddress.transferFrom(firstOrder.maker,address(this),firstOrder.tokenId) (contracts/Leasing.sol#347)
        - extendLease(secondOrder,extendToTime) (contracts/Leasing.sol#461)
                - (success) = royaltyAddress.call{value: royaltyAmount}() (contracts/Leasing.sol#418)
                - (success) = lease.lessor.call{value: requiredPaymentAmount}() (contracts/Leasing.sol#423)
                - require(bool,string)(paymentContractAddress.transferFrom(secondOrder.maker,royaltyAddress,royaltyAmount),Payment for asset failed. royalties) (contracts/Leasing.sol#432)
                - require(bool,string)(paymentContractAddress.transferFrom(secondOrder.maker,lease.lessor,requiredPaymentAmount),Payment for asset failed) (contracts/Leasing.sol#436)
        External calls sending eth:
        - atomicMatch(firstOrder,secondOrder,firstSignature,secondSignature) (contracts/Leasing.sol#460)
                - (success) = royaltyAddress.call{value: royaltyAmount}() (contracts/Leasing.sol#315)
                - (success) = firstOrder.maker.call{value: requiredPaymentAmount}() (contracts/Leasing.sol#320)
        - extendLease(secondOrder,extendToTime) (contracts/Leasing.sol#461)
                - (success) = royaltyAddress.call{value: royaltyAmount}() (contracts/Leasing.sol#418)
                - (success) = lease.lessor.call{value: requiredPaymentAmount}() (contracts/Leasing.sol#423)
        State variables written after the call(s):
        - extendLease(secondOrder,extendToTime) (contracts/Leasing.sol#461)
                - leases[tokenId] = Lease(lessor,lessee,offerHash,endTime,extendPeriodStartTime) (contracts/Leasing.sol#130)
        - extendLease(secondOrder,extendToTime) (contracts/Leasing.sol#461)
                - reentrancyLockStatus = 2 (contracts/Leasing.sol#28)
```



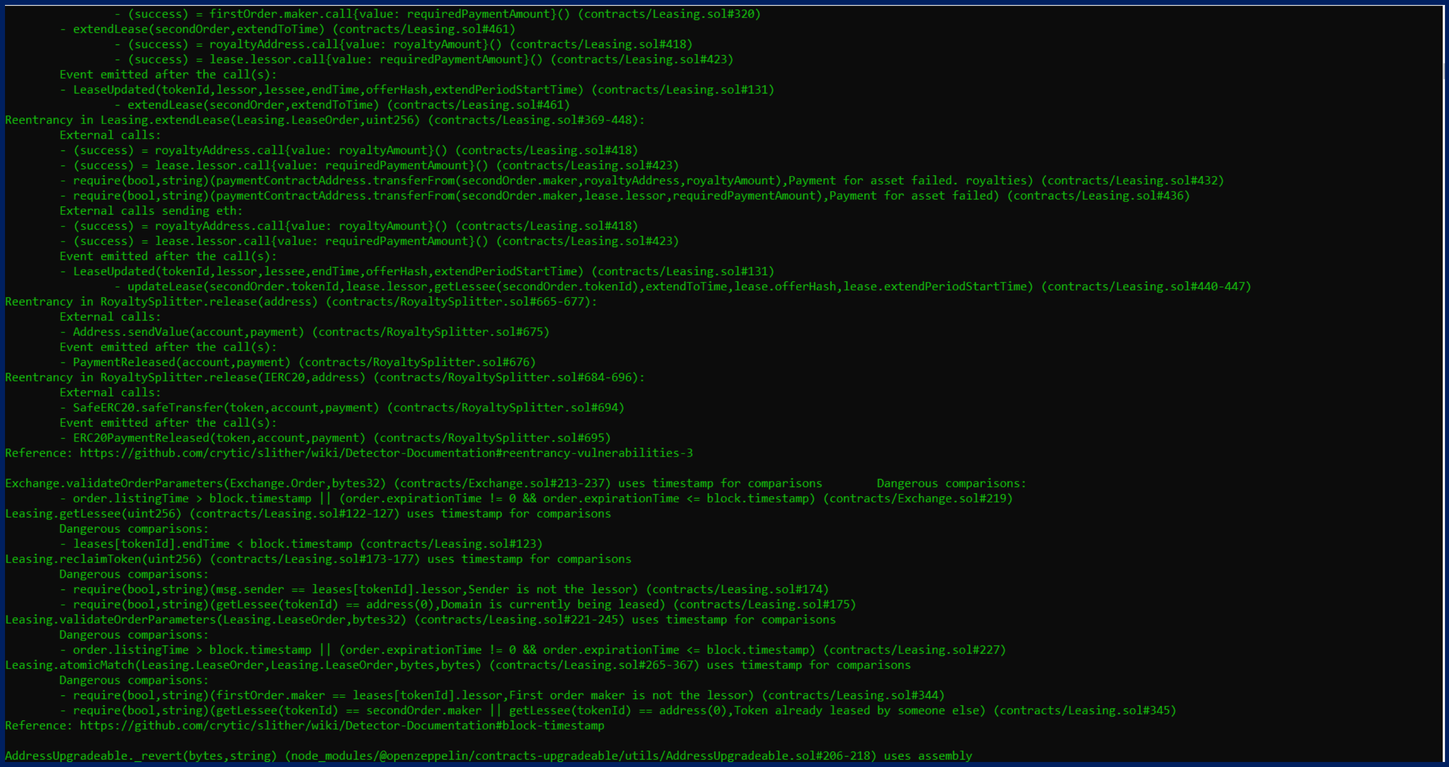
```
        - extendLease(secondOrder,extendToTime) (contracts/Leasing.sol#461)
                - reentrancyLockStatus = 2 (contracts/Leasing.sol#28)
                - reentrancyLockStatus = 1 (contracts/Leasing.sol#34)
Reentrancy in Leasing.extendLease(Leasing.LeaseOrder,uint256) (contracts/Leasing.sol#369-448):
        External calls:
        - (success) = royaltyAddress.call{value: royaltyAmount}() (contracts/Leasing.sol#418)
        - (success) = lease.lessor.call{value: requiredPaymentAmount}() (contracts/Leasing.sol#423)
        - require(bool,string)(paymentContractAddress.transferFrom(secondOrder.maker,royaltyAddress,royaltyAmount),Payment for asset failed. royalties) (contracts/Leasing.sol#432)
        - require(bool,string)(paymentContractAddress.transferFrom(secondOrder.maker,lease.lessor,requiredPaymentAmount),Payment for asset failed) (contracts/Leasing.sol#436)
        External calls sending eth:
        - (success) = royaltyAddress.call{value: royaltyAmount}() (contracts/Leasing.sol#418)
        - (success) = lease.lessor.call{value: requiredPaymentAmount}() (contracts/Leasing.sol#423)
        State variables written after the call(s):
        - updateLease(secondOrder.tokenId,lease.lessor,getLessee(secondOrder.tokenId),extendToTime,lease.offerHash,lease.extendPeriodStartTime) (contracts/Leasing.sol#440-447)
                - leases[tokenId] = Lease(lessor,lessee,offerHash,endTime,extendPeriodStartTime) (contracts/Leasing.sol#130)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities

Leasing.cancelledOrders (contracts/Leasing.sol#68) is never initialized. It is used in:
        - Leasing.validateOrderParameters(Leasing.LeaseOrder,bytes32) (contracts/Leasing.sol#221-245)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables

Token (contracts/Token.sol#5-63) has incorrect ERC20 function interface:Token.transfer(address,uint256) (contracts/Token.sol#39-51)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-erc20-interface

Leasing.atomicMatch(Leasing.LeaseOrder,Leasing.LeaseOrder,bytes,bytes) (contracts/Leasing.sol#265-367) uses a dangerous strict equality:
        - require(bool,string)(firstOrder.maker == leases[tokenId].lessor,First order maker is not the lessor) (contracts/Leasing.sol#344)
Leasing.atomicMatch(Leasing.LeaseOrder,Leasing.LeaseOrder,bytes,bytes) (contracts/Leasing.sol#265-367) uses a dangerous strict equality:
        - require(bool,string)(getLessee(tokenId) == secondOrder.maker || getLessee(tokenId) == address(0),Token already leased by someone else) (contracts/Leasing.sol#345)
Leasing.onlyLessee(uint256) (contracts/Leasing.sol#134-137) uses a dangerous strict equality:
        - require(bool,string)(msg.sender == getLessee(tokenId),Sender is not the lessee) (contracts/Leasing.sol#135)
Leasing.reclaimToken(uint256) (contracts/Leasing.sol#173-177) uses a dangerous strict equality:
        - require(bool,string)(msg.sender == leases[tokenId].lessor,Sender is not the lessor) (contracts/Leasing.sol#174)
Leasing.reclaimToken(uint256) (contracts/Leasing.sol#173-177) uses a dangerous strict equality:
        - require(bool,string)(getLessee(tokenId) == address(0),Domain is currently being leased) (contracts/Leasing.sol#175)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Leasing.atomicMatch(Leasing.LeaseOrder,Leasing.LeaseOrder,bytes,bytes).success_scope_0 (contracts/Leasing.sol#320) is a local variable never initialized
Exchange.atomicMatch(Exchange.Order,Exchange.Order,bytes,bytes).success_scope_0 (contracts/Exchange.sol#313) is a local variable never initialized
Leasing.extendLease(Leasing.LeaseOrder,uint256).success_scope_0 (contracts/Leasing.sol#423) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

Exchange.hashOrder(Exchange.Order).hash (contracts/Exchange.sol#168) shadows:
        - Exchange.hash(Exchange.EIP712Domain) (contracts/Exchange.sol#108-120) (function)
Exchange.hashToSign(bytes32).hash (contracts/Exchange.sol#190) shadows:
        - Exchange.hash(Exchange.EIP712Domain) (contracts/Exchange.sol#108-120) (function)
Exchange.validateOrderParameters(Exchange.Order,bytes32).hash (contracts/Exchange.sol#213) shadows:
        - Exchange.hash(Exchange.EIP712Domain) (contracts/Exchange.sol#108-120) (function)
Exchange.validateOrderAuthorization(bytes32,address,bytes).hash (contracts/Exchange.sol#239) shadows:
        - Exchange.hash(Exchange.EIP712Domain) (contracts/Exchange.sol#108-120) (function)
Leasing.hashOrder(Leasing.LeaseOrder).hash (contracts/Leasing.sol#182) shadows:
```



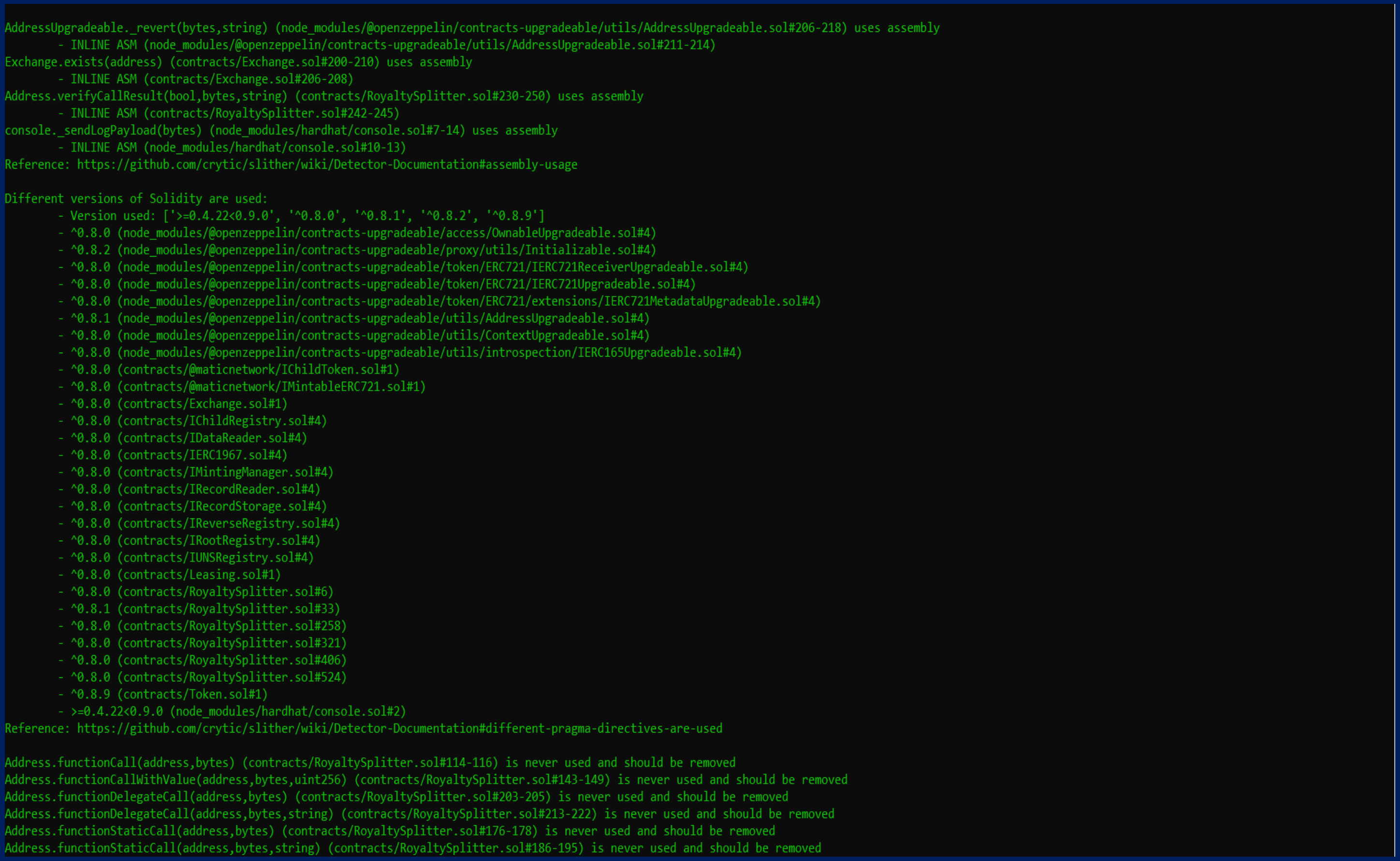
```
Reentrancy in Exchange.atomicMatch(Exchange.Order,Exchange.Order,bytes,bytes) (contracts/Exchange.sol#258-341):
        External calls:
        - (success) = royaltyAddress.call{value: royaltyAmount}() (contracts/Exchange.sol#308)
        - (success) = firstOrder.maker.call{value: orderAmount}() (contracts/Exchange.sol#313)
        - require(bool,string)(paymentContractAddress.transferFrom(secondOrder.maker,royaltyAddress,royaltyAmount),Payment for asset failed. royalties) (contracts/Exchange.sol#322)
        - require(bool,string)(paymentContractAddress.transferFrom(secondOrder.maker,firstOrder.maker,orderAmount),Payment for asset failed) (contracts/Exchange.sol#326)
        - tokenAddress.transferFrom(firstOrder.maker,secondOrder.maker,firstOrder.tokenId) (contracts/Exchange.sol#331)
        External calls sending eth:
        - (success) = royaltyAddress.call{value: royaltyAmount}() (contracts/Exchange.sol#308)
        - (success) = firstOrder.maker.call{value: orderAmount}() (contracts/Exchange.sol#313)
        Event emitted after the call(s):
        - ListingsForTokenCanceled(tokenId) (contracts/Exchange.sol#146)
                - _cancelAllListings(firstOrder.tokenId) (contracts/Exchange.sol#335)
        - OffersForTokenByUserCanceled(tokenId,maker) (contracts/Exchange.sol#155)
                - _cancelAllOffersForTokenId(secondOrder.tokenId,secondOrder.maker) (contracts/Exchange.sol#336)
        - OrdersMatched(firstHash,secondHash,firstOrder.maker,secondOrder.maker) (contracts/Exchange.sol#340)
Reentrancy in Leasing.atomicMatch(Leasing.LeaseOrder,Leasing.LeaseOrder,bytes,bytes) (contracts/Leasing.sol#265-367):
        External calls:
        - (success) = royaltyAddress.call{value: royaltyAmount}() (contracts/Leasing.sol#315)
        - (success) = firstOrder.maker.call{value: requiredPaymentAmount}() (contracts/Leasing.sol#320)
        - require(bool,string)(paymentContractAddress.transferFrom(secondOrder.maker,royaltyAddress,royaltyAmount),Payment for asset failed. royalties) (contracts/Leasing.sol#329)
        - require(bool,string)(paymentContractAddress.transferFrom(secondOrder.maker,firstOrder.maker,requiredPaymentAmount),Payment for asset failed) (contracts/Leasing.sol#333)
        - tokenAddress.transferFrom(firstOrder.maker,address(this),firstOrder.tokenId) (contracts/Leasing.sol#347)
        External calls sending eth:
        - (success) = royaltyAddress.call{value: royaltyAmount}() (contracts/Leasing.sol#315)
        - (success) = firstOrder.maker.call{value: requiredPaymentAmount}() (contracts/Leasing.sol#320)
        Event emitted after the call(s):
        - LeaseOrdersMatched(firstHash,secondHash,firstOrder.maker,secondOrder.maker) (contracts/Leasing.sol#366)
        - LeaseUpdated(tokenId,lessor,lessee,endTime,offerHash,extendPeriodStartTime) (contracts/Leasing.sol#131)
                - updateLease(tokenId,firstOrder.maker,secondOrder.maker,block.timestamp + secondOrder.initialPeriodSeconds,secondHash,block.timestamp + secondOrder.initialPeriodSeconds) (contracts/Leasing.sol#3
50-361)
Reentrancy in Leasing.atomicMatchAndExtendLease(Leasing.LeaseOrder,Leasing.LeaseOrder,bytes,bytes,uint256) (contracts/Leasing.sol#450-462):
        External calls:
        - atomicMatch(firstOrder,secondOrder,firstSignature,secondSignature) (contracts/Leasing.sol#460)
                - (success) = royaltyAddress.call{value: royaltyAmount}() (contracts/Leasing.sol#315)
                - (success) = firstOrder.maker.call{value: requiredPaymentAmount}() (contracts/Leasing.sol#320)
                - require(bool,string)(paymentContractAddress.transferFrom(secondOrder.maker,royaltyAddress,royaltyAmount),Payment for asset failed. royalties) (contracts/Leasing.sol#329)
                - require(bool,string)(paymentContractAddress.transferFrom(secondOrder.maker,firstOrder.maker,requiredPaymentAmount),Payment for asset failed) (contracts/Leasing.sol#333)
                - tokenAddress.transferFrom(firstOrder.maker,address(this),firstOrder.tokenId) (contracts/Leasing.sol#347)
        - extendLease(secondOrder,extendToTime) (contracts/Leasing.sol#461)
                - (success) = royaltyAddress.call{value: royaltyAmount}() (contracts/Leasing.sol#418)
                - (success) = lease.lessor.call{value: requiredPaymentAmount}() (contracts/Leasing.sol#423)
                - require(bool,string)(paymentContractAddress.transferFrom(secondOrder.maker,royaltyAddress,royaltyAmount),Payment for asset failed. royalties) (contracts/Leasing.sol#432)
                - require(bool,string)(paymentContractAddress.transferFrom(secondOrder.maker,lease.lessor,requiredPaymentAmount),Payment for asset failed) (contracts/Leasing.sol#436)
        External calls sending eth:
        - atomicMatch(firstOrder,secondOrder,firstSignature,secondSignature) (contracts/Leasing.sol#460)
                - (success) = royaltyAddress.call{value: royaltyAmount}() (contracts/Leasing.sol#315)
                - (success) = firstOrder.maker.call{value: requiredPaymentAmount}() (contracts/Leasing.sol#320)
        - extendLease(secondOrder,extendToTime) (contracts/Leasing.sol#461)
```



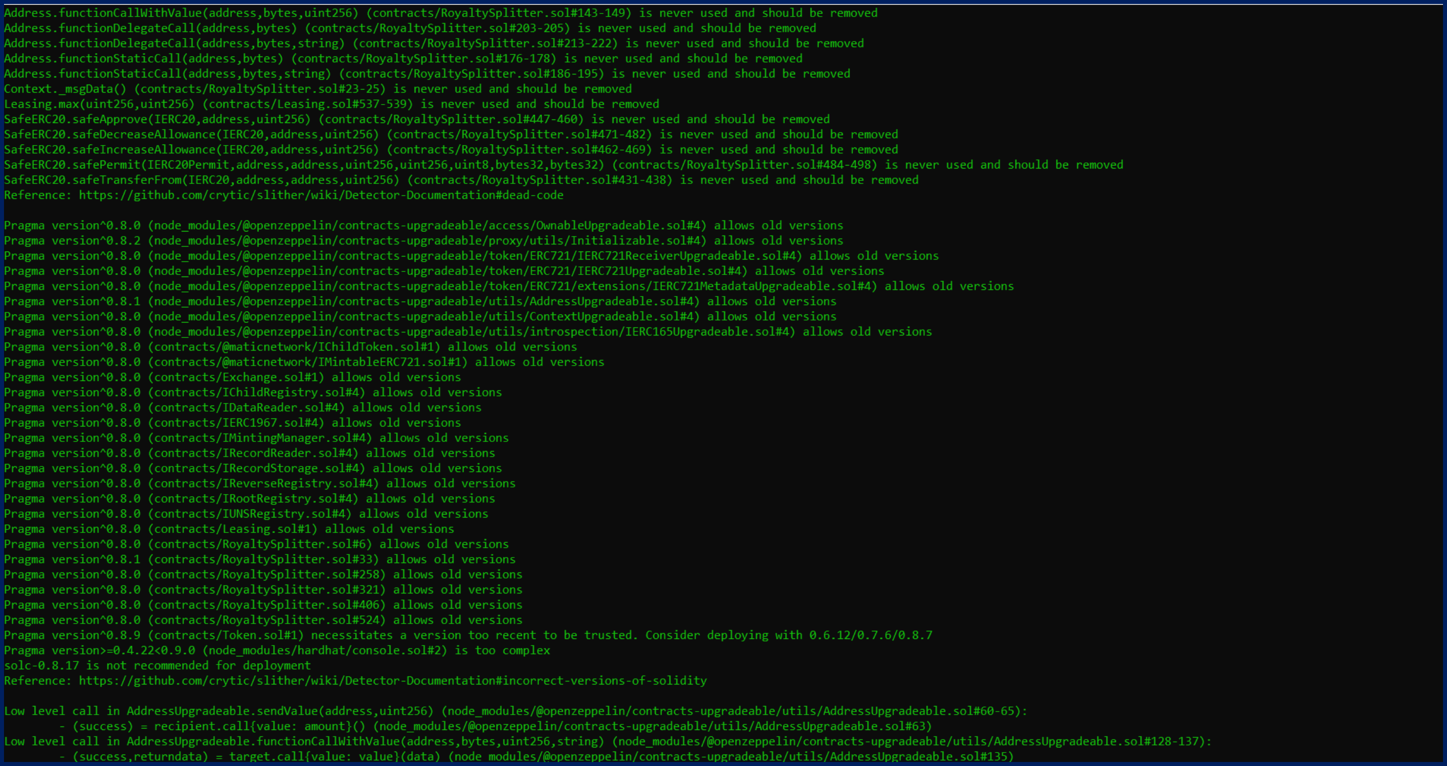
```
        - Exchange.hash(Exchange.EIP712Domain) (contracts/Exchange.sol#108-120) (function)
Leasing.hashOrder(Leasing.LeaseOrder).hash (contracts/Leasing.sol#182) shadows:
        - Leasing.hash(Leasing.EIP712Domain) (contracts/Leasing.sol#146-158) (function)
Leasing.hashToSign(bytes32).hash (contracts/Leasing.sol#211) shadows:
        - Leasing.hash(Leasing.EIP712Domain) (contracts/Leasing.sol#146-158) (function)
Leasing.validateOrderParameters(Leasing.LeaseOrder,bytes32).hash (contracts/Leasing.sol#221) shadows:
        - Leasing.hash(Leasing.EIP712Domain) (contracts/Leasing.sol#146-158) (function)
Leasing.validateOrderAuthorization(bytes32,address,bytes).hash (contracts/Leasing.sol#247) shadows:
        - Leasing.hash(Leasing.EIP712Domain) (contracts/Leasing.sol#146-158) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

Exchange.setRoyaltyBasisPoints(uint256) (contracts/Exchange.sol#122-125) should emit an event for:
        - royaltyBasisPoints = _royaltyBasisPoints (contracts/Exchange.sol#124)
Leasing.setRoyaltyBasisPoints(uint256) (contracts/Leasing.sol#160-163) should emit an event for:
        - royaltyBasisPoints = _royaltyBasisPoints (contracts/Leasing.sol#162)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

Exchange.initialize(address,uint256,address,uint256)._royaltyAddress (contracts/Exchange.sol#86) lacks a zero-check on :                - royaltyAddress = _royaltyAddress (contracts/Exchange.sol#88)
Exchange.setRoyaltyAddress(address)._royaltyAddress (contracts/Exchange.sol#127) lacks a zero-check on :
                - royaltyAddress = _royaltyAddress (contracts/Exchange.sol#128)
Leasing.initialize(address,address,uint256,address,uint256)._royaltyAddress (contracts/Leasing.sol#73) lacks a zero-check on :
                - royaltyAddress = _royaltyAddress (contracts/Leasing.sol#75)
Leasing.setRoyaltyAddress(address)._royaltyAddress (contracts/Leasing.sol#165) lacks a zero-check on :
                - royaltyAddress = _royaltyAddress (contracts/Leasing.sol#166)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Variable 'Exchange.atomicMatch(Exchange.Order,Exchange.Order,bytes,bytes).success (contracts/Exchange.sol#308)' in Exchange.atomicMatch(Exchange.Order,Exchange.Order,bytes,bytes) (contracts/Exchange.sol#258-341)
 potentially used before declaration: (success) = firstOrder.maker.call{value: orderAmount}() (contracts/Exchange.sol#313)
Variable 'Leasing.atomicMatch(Leasing.LeaseOrder,Leasing.LeaseOrder,bytes,bytes).success (contracts/Leasing.sol#315)' in Leasing.atomicMatch(Leasing.LeaseOrder,Leasing.LeaseOrder,bytes,bytes) (contracts/Leasing.
sol#265-367) potentially used before declaration: (success) = firstOrder.maker.call{value: requiredPaymentAmount}() (contracts/Leasing.sol#320)
Variable 'Leasing.extendLease(Leasing.LeaseOrder,uint256).success (contracts/Leasing.sol#418)' in Leasing.extendLease(Leasing.LeaseOrder,uint256) (contracts/Leasing.sol#369-448) potentially used before declarati
on: (success) = lease.lessor.call{value: requiredPaymentAmount}() (contracts/Leasing.sol#423)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

Reentrancy in Leasing.atomicMatch(Leasing.LeaseOrder,Leasing.LeaseOrder,bytes,bytes) (contracts/Leasing.sol#265-367):
        External calls:
        - (success) = royaltyAddress.call{value: royaltyAmount}() (contracts/Leasing.sol#315)
        - (success) = firstOrder.maker.call{value: requiredPaymentAmount}() (contracts/Leasing.sol#320)
        - require(bool,string)(paymentContractAddress.transferFrom(secondOrder.maker,royaltyAddress,royaltyAmount),Payment for asset failed. royalties) (contracts/Leasing.sol#329)
        - require(bool,string)(paymentContractAddress.transferFrom(secondOrder.maker,firstOrder.maker,requiredPaymentAmount),Payment for asset failed) (contracts/Leasing.sol#333)
        - tokenAddress.transferFrom(firstOrder.maker,address(this),firstOrder.tokenId) (contracts/Leasing.sol#347)
        External calls sending eth:
        - (success) = royaltyAddress.call{value: royaltyAmount}() (contracts/Leasing.sol#315)
        - (success) = firstOrder.maker.call{value: requiredPaymentAmount}() (contracts/Leasing.sol#320)
        State variables written after the call(s):
        - updateLease(tokenId,firstOrder.maker,secondOrder.maker,block.timestamp + secondOrder.initialPeriodSeconds,secondHash,block.timestamp + secondOrder.initialPeriodSeconds) (contracts/Leasing.sol#350-361)
                - leases[tokenId] = Lease(lessor,lessee,offerHash,endTime,extendPeriodStartTime) (contracts/Leasing.sol#130)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in Exchange.atomicMatch(Exchange.Order,Exchange.Order,bytes,bytes) (contracts/Exchange.sol#258-341):
```



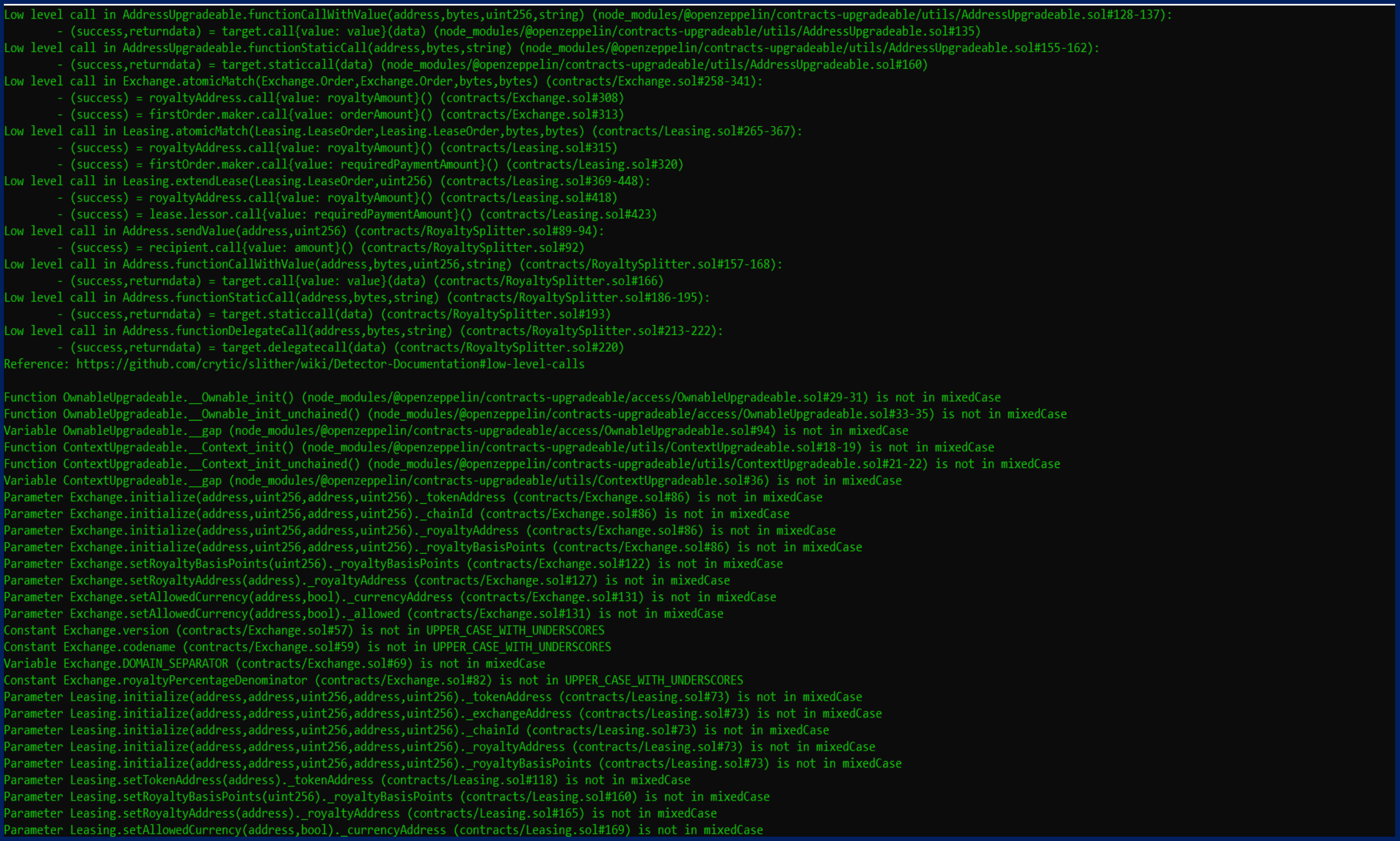
```
                - (success) = firstOrder.maker.call{value: requiredPaymentAmount}() (contracts/Leasing.sol#320)
        - extendLease(secondOrder,extendToTime) (contracts/Leasing.sol#461)
                - (success) = royaltyAddress.call{value: royaltyAmount}() (contracts/Leasing.sol#418)
                - (success) = lease.lessor.call{value: requiredPaymentAmount}() (contracts/Leasing.sol#423)
        Event emitted after the call(s):
        - LeaseUpdated(tokenId,lessor,lessee,endTime,offerHash,extendPeriodStartTime) (contracts/Leasing.sol#131)
                - extendLease(secondOrder,extendToTime) (contracts/Leasing.sol#461)
Reentrancy in Leasing.extendLease(Leasing.LeaseOrder,uint256) (contracts/Leasing.sol#369-448):
        External calls:
        - (success) = royaltyAddress.call{value: royaltyAmount}() (contracts/Leasing.sol#418)
        - (success) = lease.lessor.call{value: requiredPaymentAmount}() (contracts/Leasing.sol#423)
        - require(bool,string)(paymentContractAddress.transferFrom(secondOrder.maker,royaltyAddress,royaltyAmount),Payment for asset failed. royalties) (contracts/Leasing.sol#432)
        - require(bool,string)(paymentContractAddress.transferFrom(secondOrder.maker,lease.lessor,requiredPaymentAmount),Payment for asset failed) (contracts/Leasing.sol#436)
        External calls sending eth:
        - (success) = royaltyAddress.call{value: royaltyAmount}() (contracts/Leasing.sol#418)
        - (success) = lease.lessor.call{value: requiredPaymentAmount}() (contracts/Leasing.sol#423)
        Event emitted after the call(s):
        - LeaseUpdated(tokenId,lessor,lessee,endTime,offerHash,extendPeriodStartTime) (contracts/Leasing.sol#131)
                - updateLease(secondOrder.tokenId,lease.lessor,getLessee(secondOrder.tokenId),extendToTime,lease.offerHash,lease.extendPeriodStartTime) (contracts/Leasing.sol#440-447)
Reentrancy in RoyaltySplitter.release(address) (contracts/RoyaltySplitter.sol#665-677):
        External calls:
        - Address.sendValue(account,payment) (contracts/RoyaltySplitter.sol#675)
        Event emitted after the call(s):
        - PaymentReleased(account,payment) (contracts/RoyaltySplitter.sol#676)
Reentrancy in RoyaltySplitter.release(IERC20,address) (contracts/RoyaltySplitter.sol#684-696):
        External calls:
        - SafeERC20.safeTransfer(token,account,payment) (contracts/RoyaltySplitter.sol#694)
        Event emitted after the call(s):
        - ERC20PaymentReleased(token,account,payment) (contracts/RoyaltySplitter.sol#695)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Exchange.validateOrderParameters(Exchange.Order,bytes32) (contracts/Exchange.sol#213-237) uses timestamp for comparisons        Dangerous comparisons:
        - order.listingTime > block.timestamp || (order.expirationTime != 0 && order.expirationTime <= block.timestamp) (contracts/Exchange.sol#219)
Leasing.getLessee(uint256) (contracts/Leasing.sol#122-127) uses timestamp for comparisons
        Dangerous comparisons:
        - leases[tokenId].endTime < block.timestamp (contracts/Leasing.sol#123)
Leasing.reclaimToken(uint256) (contracts/Leasing.sol#173-177) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(msg.sender == leases[tokenId].lessor,Sender is not the lessor) (contracts/Leasing.sol#174)
        - require(bool,string)(getLessee(tokenId) == address(0),Domain is currently being leased) (contracts/Leasing.sol#175)
Leasing.validateOrderParameters(Leasing.LeaseOrder,bytes32) (contracts/Leasing.sol#221-245) uses timestamp for comparisons
        Dangerous comparisons:
        - order.listingTime > block.timestamp || (order.expirationTime != 0 && order.expirationTime <= block.timestamp) (contracts/Leasing.sol#227)
Leasing.atomicMatch(Leasing.LeaseOrder,Leasing.LeaseOrder,bytes,bytes) (contracts/Leasing.sol#265-367) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(firstOrder.maker == leases[tokenId].lessor,First order maker is not the lessor) (contracts/Leasing.sol#344)
        - require(bool,string)(getLessee(tokenId) == secondOrder.maker || getLessee(tokenId) == address(0),Token already leased by someone else) (contracts/Leasing.sol#345)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

AddressUpgradeable._revert(bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#206-218) uses assembly
```



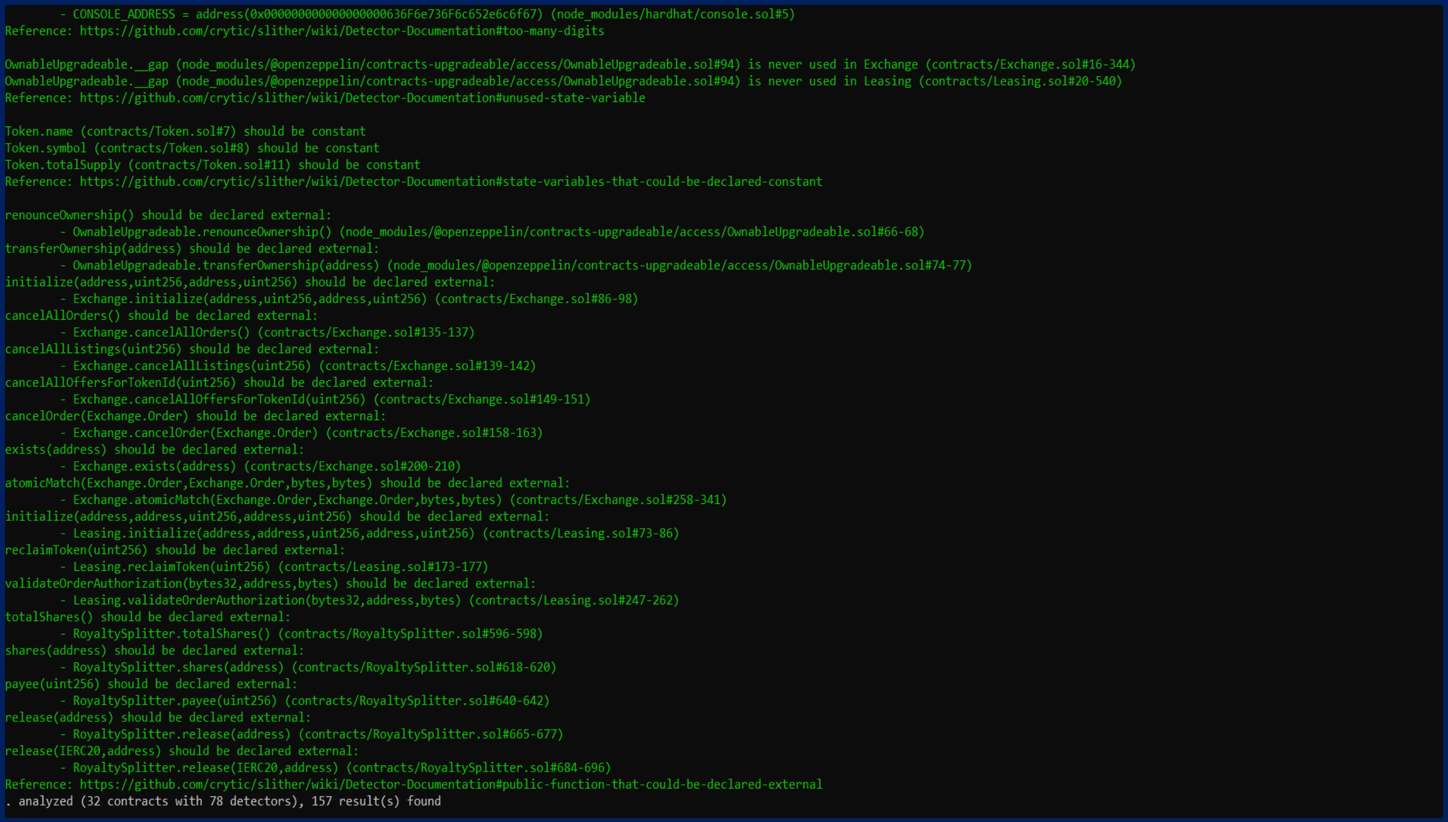
```
AddressUpgradeable._revert(bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#206-218) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#211-214)
Exchange.exists(address) (contracts/Exchange.sol#200-210) uses assembly
        - INLINE ASM (contracts/Exchange.sol#206-208)
Address.verifyCallResult(bool,bytes,string) (contracts/RoyaltySplitter.sol#230-250) uses assembly
        - INLINE ASM (contracts/RoyaltySplitter.sol#242-245)
console._sendLogPayload(bytes) (node_modules/hardhat/console.sol#7-14) uses assembly
        - INLINE ASM (node_modules/hardhat/console.sol#10-13)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity are used:
        - Version used: ['>=0.4.22<0.9.0', '^0.8.0', '^0.8.1', '^0.8.2', '^0.8.9']
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#4)
        - ^0.8.2 (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/IERC721ReceiverUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/IERC721Upgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/extensions/IERC721MetadataUpgradeable.sol#4)
        - ^0.8.1 (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/IERC165Upgradeable.sol#4)
        - ^0.8.0 (contracts/@maticnetwork/IChildToken.sol#1)
        - ^0.8.0 (contracts/@maticnetwork/IMintableERC721.sol#1)
        - ^0.8.0 (contracts/Exchange.sol#1)
        - ^0.8.0 (contracts/IChildRegistry.sol#4)
        - ^0.8.0 (contracts/IDataReader.sol#4)
        - ^0.8.0 (contracts/IERC1967.sol#4)
        - ^0.8.0 (contracts/IMintingManager.sol#4)
        - ^0.8.0 (contracts/IRecordReader.sol#4)
        - ^0.8.0 (contracts/IRecordStorage.sol#4)
        - ^0.8.0 (contracts/IReverseRegistry.sol#4)
        - ^0.8.0 (contracts/IRootRegistry.sol#4)
        - ^0.8.0 (contracts/IUNSRegistry.sol#4)
        - ^0.8.0 (contracts/Leasing.sol#1)
        - ^0.8.0 (contracts/RoyaltySplitter.sol#6)
        - ^0.8.1 (contracts/RoyaltySplitter.sol#33)
        - ^0.8.0 (contracts/RoyaltySplitter.sol#258)
        - ^0.8.0 (contracts/RoyaltySplitter.sol#321)
        - ^0.8.0 (contracts/RoyaltySplitter.sol#406)
        - ^0.8.0 (contracts/RoyaltySplitter.sol#524)
        - ^0.8.9 (contracts/Token.sol#1)
        - >=0.4.22<0.9.0 (node_modules/hardhat/console.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Address.functionCall(address,bytes) (contracts/RoyaltySplitter.sol#114-116) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (contracts/RoyaltySplitter.sol#143-149) is never used and should be removed
Address.functionDelegateCall(address,bytes) (contracts/RoyaltySplitter.sol#203-205) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (contracts/RoyaltySplitter.sol#213-222) is never used and should be removed
Address.functionStaticCall(address,bytes) (contracts/RoyaltySplitter.sol#176-178) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (contracts/RoyaltySplitter.sol#186-195) is never used and should be removed
```



```
Address.functionCallWithValue(address,bytes,uint256) (contracts/RoyaltySplitter.sol#143-149) is never used and should be removed
Address.functionDelegateCall(address,bytes) (contracts/RoyaltySplitter.sol#203-205) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (contracts/RoyaltySplitter.sol#213-222) is never used and should be removed
Address.functionStaticCall(address,bytes) (contracts/RoyaltySplitter.sol#176-178) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (contracts/RoyaltySplitter.sol#186-195) is never used and should be removed
Context._msgData() (contracts/RoyaltySplitter.sol#23-25) is never used and should be removed
Leasing.max(uint256,uint256) (contracts/Leasing.sol#537-539) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (contracts/RoyaltySplitter.sol#447-460) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (contracts/RoyaltySplitter.sol#471-482) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (contracts/RoyaltySplitter.sol#462-469) is never used and should be removed
SafeERC20.safePermit(IERC20Permit,address,address,uint256,uint256,uint8,bytes32,bytes32) (contracts/RoyaltySplitter.sol#484-498) is never used and should be removed
SafeERC20.safeTransferFrom(IERC20,address,address,uint256) (contracts/RoyaltySplitter.sol#431-438) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#4) allows old versions
Pragma version^0.8.2 (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/IERC721ReceiverUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/IERC721Upgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/extensions/IERC721MetadataUpgradeable.sol#4) allows old versions
Pragma version^0.8.1 (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/IERC165Upgradeable.sol#4) allows old versions
Pragma version^0.8.0 (contracts/@maticnetwork/IChildToken.sol#1) allows old versions
Pragma version^0.8.0 (contracts/@maticnetwork/IMintableERC721.sol#1) allows old versions
Pragma version^0.8.0 (contracts/Exchange.sol#1) allows old versions
Pragma version^0.8.0 (contracts/IChildRegistry.sol#4) allows old versions
Pragma version^0.8.0 (contracts/IDataReader.sol#4) allows old versions
Pragma version^0.8.0 (contracts/IERC1967.sol#4) allows old versions
Pragma version^0.8.0 (contracts/IMintingManager.sol#4) allows old versions
Pragma version^0.8.0 (contracts/IRecordReader.sol#4) allows old versions
Pragma version^0.8.0 (contracts/IRecordStorage.sol#4) allows old versions
Pragma version^0.8.0 (contracts/IReverseRegistry.sol#4) allows old versions
Pragma version^0.8.0 (contracts/IRootRegistry.sol#4) allows old versions
Pragma version^0.8.0 (contracts/IUNSRegistry.sol#4) allows old versions
Pragma version^0.8.0 (contracts/Leasing.sol#1) allows old versions
Pragma version^0.8.0 (contracts/RoyaltySplitter.sol#6) allows old versions
Pragma version^0.8.1 (contracts/RoyaltySplitter.sol#33) allows old versions
Pragma version^0.8.0 (contracts/RoyaltySplitter.sol#258) allows old versions
Pragma version^0.8.0 (contracts/RoyaltySplitter.sol#321) allows old versions
Pragma version^0.8.0 (contracts/RoyaltySplitter.sol#406) allows old versions
Pragma version^0.8.0 (contracts/RoyaltySplitter.sol#524) allows old versions
Pragma version^0.8.9 (contracts/Token.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version>=0.4.22<0.9.0 (node_modules/hardhat/console.sol#2) is too complex
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in AddressUpgradeable.sendValue(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#60-65):
        - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#63)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#128-137):
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#135)
```



```
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#128-137):
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#135)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#155-162):
        - (success,returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#160)
Low level call in Exchange.atomicMatch(Exchange.Order,Exchange.Order,bytes,bytes) (contracts/Exchange.sol#258-341):
        - (success) = royaltyAddress.call{value: royaltyAmount}() (contracts/Exchange.sol#308)
        - (success) = firstOrder.maker.call{value: orderAmount}() (contracts/Exchange.sol#313)
Low level call in Leasing.atomicMatch(Leasing.LeaseOrder,Leasing.LeaseOrder,bytes,bytes) (contracts/Leasing.sol#265-367):
        - (success) = royaltyAddress.call{value: royaltyAmount}() (contracts/Leasing.sol#315)
        - (success) = firstOrder.maker.call{value: requiredPaymentAmount}() (contracts/Leasing.sol#320)
Low level call in Leasing.extendLease(Leasing.LeaseOrder,uint256) (contracts/Leasing.sol#369-448):
        - (success) = royaltyAddress.call{value: royaltyAmount}() (contracts/Leasing.sol#418)
        - (success) = lease.lessor.call{value: requiredPaymentAmount}() (contracts/Leasing.sol#423)
Low level call in Address.sendValue(address,uint256) (contracts/RoyaltySplitter.sol#89-94):
        - (success) = recipient.call{value: amount}() (contracts/RoyaltySplitter.sol#92)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (contracts/RoyaltySplitter.sol#157-168):
        - (success,returndata) = target.call{value: value}(data) (contracts/RoyaltySplitter.sol#166)
Low level call in Address.functionStaticCall(address,bytes,string) (contracts/RoyaltySplitter.sol#186-195):
        - (success,returndata) = target.staticcall(data) (contracts/RoyaltySplitter.sol#193)
Low level call in Address.functionDelegateCall(address,bytes,string) (contracts/RoyaltySplitter.sol#213-222):
        - (success,returndata) = target.delegatecall(data) (contracts/RoyaltySplitter.sol#220)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function OwnableUpgradeable.__Ownable_init() (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#29-31) 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#94) is not in mixedCase
Function ContextUpgradeable.__Context_init() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#18-19) 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#36) is not in mixedCase
Parameter Exchange.initialize(address,uint256,address,uint256)._tokenAddress (contracts/Exchange.sol#86) is not in mixedCase
Parameter Exchange.initialize(address,uint256,address,uint256)._chainId (contracts/Exchange.sol#86) is not in mixedCase
Parameter Exchange.initialize(address,uint256,address,uint256)._royaltyAddress (contracts/Exchange.sol#86) is not in mixedCase
Parameter Exchange.initialize(address,uint256,address,uint256)._royaltyBasisPoints (contracts/Exchange.sol#86) is not in mixedCase
Parameter Exchange.setRoyaltyBasisPoints(uint256)._royaltyBasisPoints (contracts/Exchange.sol#122) is not in mixedCase
Parameter Exchange.setRoyaltyAddress(address)._royaltyAddress (contracts/Exchange.sol#127) is not in mixedCase
Parameter Exchange.setAllowedCurrency(address,bool)._currencyAddress (contracts/Exchange.sol#131) is not in mixedCase
Parameter Exchange.setAllowedCurrency(address,bool)._allowed (contracts/Exchange.sol#131) is not in mixedCase
Constant Exchange.version (contracts/Exchange.sol#57) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Exchange.codename (contracts/Exchange.sol#59) is not in UPPER_CASE_WITH_UNDERSCORES
Variable Exchange.DOMAIN_SEPARATOR (contracts/Exchange.sol#69) is not in mixedCase
Constant Exchange.royaltyPercentageDenominator (contracts/Exchange.sol#82) is not in UPPER_CASE_WITH_UNDERSCORES
Parameter Leasing.initialize(address,address,uint256,address,uint256)._tokenAddress (contracts/Leasing.sol#73) is not in mixedCase
Parameter Leasing.initialize(address,address,uint256,address,uint256)._exchangeAddress (contracts/Leasing.sol#73) is not in mixedCase
Parameter Leasing.initialize(address,address,uint256,address,uint256)._chainId (contracts/Leasing.sol#73) is not in mixedCase
Parameter Leasing.initialize(address,address,uint256,address,uint256)._royaltyAddress (contracts/Leasing.sol#73) is not in mixedCase
Parameter Leasing.initialize(address,address,uint256,address,uint256)._royaltyBasisPoints (contracts/Leasing.sol#73) is not in mixedCase
Parameter Leasing.setTokenAddress(address)._tokenAddress (contracts/Leasing.sol#118) is not in mixedCase
Parameter Leasing.setRoyaltyBasisPoints(uint256)._royaltyBasisPoints (contracts/Leasing.sol#160) is not in mixedCase
Parameter Leasing.setRoyaltyAddress(address)._royaltyAddress (contracts/Leasing.sol#165) is not in mixedCase
Parameter Leasing.setAllowedCurrency(address,bool)._currencyAddress (contracts/Leasing.sol#169) is not in mixedCase
```



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

OwnableUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#94) is never used in Exchange (contracts/Exchange.sol#16-344)
OwnableUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#94) is never used in Leasing (contracts/Leasing.sol#20-540)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

Token.name (contracts/Token.sol#7) should be constant
Token.symbol (contracts/Token.sol#8) should be constant
Token.totalSupply (contracts/Token.sol#11) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

renounceOwnership() should be declared external:
        - OwnableUpgradeable.renounceOwnership() (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#66-68)
transferOwnership(address) should be declared external:
        - OwnableUpgradeable.transferOwnership(address) (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#74-77)
initialize(address,uint256,address,uint256) should be declared external:
        - Exchange.initialize(address,uint256,address,uint256) (contracts/Exchange.sol#86-98)
cancelAllOrders() should be declared external:
        - Exchange.cancelAllOrders() (contracts/Exchange.sol#135-137)
cancelAllListings(uint256) should be declared external:
        - Exchange.cancelAllListings(uint256) (contracts/Exchange.sol#139-142)
cancelAllOffersForTokenId(uint256) should be declared external:
        - Exchange.cancelAllOffersForTokenId(uint256) (contracts/Exchange.sol#149-151)
cancelOrder(Exchange.Order) should be declared external:
        - Exchange.cancelOrder(Exchange.Order) (contracts/Exchange.sol#158-163)
exists(address) should be declared external:
        - Exchange.exists(address) (contracts/Exchange.sol#200-210)
atomicMatch(Exchange.Order,Exchange.Order,bytes,bytes) should be declared external:
        - Exchange.atomicMatch(Exchange.Order,Exchange.Order,bytes,bytes) (contracts/Exchange.sol#258-341)
initialize(address,address,uint256,address,uint256) should be declared external:
        - Leasing.initialize(address,address,uint256,address,uint256) (contracts/Leasing.sol#73-86)
reclaimToken(uint256) should be declared external:
        - Leasing.reclaimToken(uint256) (contracts/Leasing.sol#173-177)
validateOrderAuthorization(bytes32,address,bytes) should be declared external:
        - Leasing.validateOrderAuthorization(bytes32,address,bytes) (contracts/Leasing.sol#247-262)
totalShares() should be declared external:
        - RoyaltySplitter.totalShares() (contracts/RoyaltySplitter.sol#596-598)
shares(address) should be declared external:
        - RoyaltySplitter.shares(address) (contracts/RoyaltySplitter.sol#618-620)
payee(uint256) should be declared external:
        - RoyaltySplitter.payee(uint256) (contracts/RoyaltySplitter.sol#640-642)
release(address) should be declared external:
        - RoyaltySplitter.release(address) (contracts/RoyaltySplitter.sol#665-677)
release(IERC20,address) should be declared external:
        - RoyaltySplitter.release(IERC20,address) (contracts/RoyaltySplitter.sol#684-696)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
. analyzed (32 contracts with 78 detectors), 157 result(s) found
```

2. Manual Audit Result

- Technical Result

File	Contract	Category	Result
Exchange.sol	Exchange	▪ Initialize	Valid
		▪ hash	Valid
		▪ setRoyaltyBasisPoints	Valid
		▪ setRoyaltyAddress	Valid
		▪ setAllowedCurrency	Valid
		▪ cancelAllOrders	Valid
		▪ cancelAllListings	Valid
		▪ _cancelAllListings	Valid
		▪ cancelAllOffersForTokenId	Valid

		▪ _cancelAllOffersForTokenId ▪ hashOrder ▪ hashToSign ▪ exists ▪ validateOrderParameters ▪ validateOrderAuthorization ▪ atomicMatch	Valid Valid Valid Valid Valid Valid Valid
Leasing.sol	Leasing	▪ initialize ▪ setTokenAddress ▪ getLessee ▪ updateLease ▪ onlyLessee ▪ hash ▪ setRoyaltyBasisPoints ▪ setRoyaltyAddress ▪ setAllowedCurrency ▪ reclaimToken ▪ hashOrder ▪ hashToSign ▪ validateOrderParameters ▪ validateOrderAuthorization ▪ atomicMatch ▪ extendLease ▪ atomicMatchAndExtendLease ▪ unleaseDomain ▪ getLeaseInfo ▪ set ▪ setMany	Valid Valid

		▪ setByHash ▪ setManyByHash ▪ reconfigure ▪ reset ▪ max	Valid Valid Valid Valid Valid
RoyaltySplitter.sol	RoyaltySplitter	▪ constructor payable ▪ totalShares ▪ totalReleased ▪ totalReleased ▪ shares ▪ released ▪ payee ▪ releasable ▪ release ▪ _pendingPayment ▪ _addPayee	Valid Valid Valid Valid Valid Valid Valid Valid Valid Valid Valid
IChildRegistry.sol	-	-	Valid
IDataReader.sol	-	-	Valid
IERC1967.sol	-	-	Valid
IMintingManager.sol	-	-	Valid
IRecordReader.sol	-	-	Valid
IRecordStorage.sol	-	-	Valid
IRegistryReader.sol	-	-	Valid
IReverseRegistry.sol	-	-	Valid
IRootRegistry.sol	-	-	Valid
IUNSRegistry.sol	-	-	Valid
IChildToken.sol	-	-	Valid
IMintableERC721.sol	-	-	Valid

- Code Quality

This audit scope has 3 main smart contract files. Smart contracts contain Libraries, Smart contracts, inherits and Interfaces. This is a compact and well written smart contract.

△ Conclusion

I was given all contract codes in the form of .sol files.

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools.

Smart contracts have been developed exactly as claimed features and don't contain high severity issues and vulnerabilities.

Altogether the code is well-written and demonstrates effective use of abstraction, separation of concern, and modularity.

Security state of the reviewed contract, based on standard audit procedure scope, is “Well-Secured”.

So, it’s good to go to production.