Guard

Smart contracts security assessment

Final report

Tariff: Standard

Nifty Row

May 2022

0xguard.com

hello@0xguard.com

Contents

Introduction

The report has been prepared for Nifty Row. The code available in the Github repo has been audited after the 08f2053b425825d2dda252ce747efa91aa0fe7b0 commit.

Name	Nifty Row
Audit date	2022-05-23 - 2022-05-23
Language	Solidity
Platform	Harmony

Contracts checked

Name	Address
LegalTender.sol	https://github.com/Nifty-Row/yasuke/blob/08f2053b425825d2dda252ce747efa91aa0fe7b0/yasuke-contracts/contracts/LegalTender.sol
Storage.sol	https://github.com/Nifty-Row/yasuke/blob/08f2053b425825d2dda252ce747efa91aa0fe7b0/yasuke-contracts/contracts/Storage.sol
Token.sol	https://github.com/Nifty-Row/yasuke/blob/08f2053b425825d2dda252ce747efa91aa0fe7b0/yasuke-contracts/contracts/Token.sol
YASUKE.sol	https://github.com/Nifty-Row/yasuke/blob/08f2053b425825d2dda252ce747efa91aa0fe7b0/yasuke-contracts/contracts/YASUKE.sol
models.sol, console.sol, StorageInterface.sol, YasukeInterface.sol	https://github.com/Nifty-Row/yasuke/tree/08f2053b425825d2dda252ce747efa91aa0fe7b0/yasuke-contracts/contracts

Procedure

We perform our audit according to the following procedure:

Automated analysis

- Scanning the project's smart contracts with several publicly available automated Solidity analysis tools
- Manual verification (reject or confirm) all the issues found by the tools

Manual audit

- Manually analyze smart contracts for security vulnerabilities
- Smart contracts' logic check

Known vulnerabilities checked

Title	Check result
Unencrypted Private Data On-Chain	passed
Code With No Effects	passed
Message call with hardcoded gas amount	passed
Typographical Error	passed
DoS With Block Gas Limit	passed
Presence of unused variables	not passed
Incorrect Inheritance Order	passed
Requirement Violation	passed
Weak Sources of Randomness from Chain Attributes	passed
Shadowing State Variables	passed

Incorrect Constructor Name	passed
Block values as a proxy for time	passed
Authorization through tx.origin	passed
DoS with Failed Call	not passed
Delegatecall to Untrusted Callee	passed
Use of Deprecated Solidity Functions	passed
Assert Violation	passed
State Variable Default Visibility	passed
Reentrancy	not passed
Unprotected SELFDESTRUCT Instruction	passed
Unprotected Ether Withdrawal	passed
Unchecked Call Return Value	passed
Floating Pragma	not passed
Outdated Compiler Version	passed
Integer Overflow and Underflow	passed
Function Default Visibility	passed

Classification of issue severity

High severity High severity issues can cause a significant or full loss of funds, change of contract ownership, major interference with contract logic. Such issues require immediate attention.

Medium severity Medium severity issues do not pose an immediate risk, but can be detrimental to the client's reputation if exploited. Medium severity issues may lead to a contract failure and can be fixed by modifying the contract state or redeployment. Such issues require attention.

Low severity	Low severity issues do not cause significant destruction to the contract's functionality. Such issues are recommended to be taken into consideration.

Issues

High severity issues

1. transferFrom() ignores allowance (LegalTender.sol)

The transferFrom() function doesn't read or update the allowance of the sender for recipient.

Recommendation: We recommend sticking with the standard ERC20 implementation.

2. Mintable token (LegalTender.sol)

The `mint()` function is accessible for a whitelist of accounts under the owner governance.

Recommendation: Filland freeze the minters list by renouncing the DEFAULT_ADMIN_ROLE.

3. Possibly unlimited fees (Storage.sol)

`setXendFeesPercentage()` and `setIssuerFeesPercentage()` functions could be used to set unlimited fees.

Recommendation: Add sanity checks for new values.

4. No authentication (Storage.sol)

The `setAdmin()` function has no authentication.

Recommendation: Function's access must be restricted.

5. tokens[] mapping (Storage.sol)

The logic of `tokens[]` mapping is broken. `addToken()` rewrites the `tokens[tokenId]` address with each call of `YASUKE.issueToken(tokenId,…)`. `getToken()` returns the last token address

written.

Recommendation: Refactor the contract and add proper tests.

6. Token.changeOwnership() may fail (YASUKE.sol)

`Token.changeOwnership()` reverts if current owner resets the approval to the YASUKE contract. In that case `buyNow()` and `_withdrawal()` functions become inaccessible and the last bidder losts funds.

7. No authorization (YASUKE.sol)

The `upgrade()` function has public access.

`cancelAuction()` is a public method without authorization

Recommendation: Functions' access must be restricted.

8. Native currency recipients can sabotage (YASUKE.sol)

The current highest bidder can sabotage any future bids by spending all the gas in the bid refund call or reverting any native transfers. `t.getIssuer()` and `store.getXendFeesAddress()` can sabotage `_withdrawOwner()` the same way.

Recommendation: Restrict bidding to EOA only by requiring `tx.origin == msg.sender`. Limit the forwarded gas amount by a reasonable value in `_withdrawOwner()` function.

9. Blocked withdrawal (YASUKE.sol)

Calling `_withdrawal()` by the highest bidder makes it impossible for the seller to collect the payment.

Recommendation: The contract's logic must be refactored. Proper testing should be performed.

10. Wrong payment recipient (YASUKE.sol)

`buyNow()` returns native payment to the buyer, not the token owner (seller). It also burns the ERC20

payment instead of transferring it to the seller.

Recommendation: Fix the recipient. Add documentation if burning ERC20 tokens is desired behavior.

Medium severity issues

1. Possible reentrancy (YASUKE.sol)

Transferring native EVM currency via `call()` is susceptible to reentrancy. We recommend adding ReentrancyGuard to all external user-interacting functions.

2. Token minting (YASUKE.sol)

`issueToken()` logic is broken. The new contract is deployed with only 1 id minted with every call of `issueToken()` function.

3. Not all functions are implemented (YASUKE.sol)

The Yasuke contract is meant to be admin of the Storage contract but has no methods for `setIssuerFeesPercentage()`, `setXendFeesAddress()`, `setBuyWithToken()` calls.

4. Contract logic (YASUKE.sol)

When contract is cancelled highestBidder can `changeOwnership` of the token.

Low severity issues

1. AccessControl wrong initialization (LegalTender.sol)

It is recommended by the OpenZeppelin documentation to use `_setupRole()` in the constructor.

2. Inheritance order (LegalTender.sol)

No need to inherit ERC20 alongside ERC20Burnable (and to override `transferFrom()`).

3. Authorization modifier (Storage.sol)

Admin authorization could be moved to a modifier.

4. Too wide pragma (Storage.sol)

Pragma version is set >= 0.7.0 < 0.9.0, allowing both pre-0.8 versions and post-0.8 with internal SafeMath. We recommend avoiding such a wide pragma range.

5. Double import (Storage.sol)

Token.sol is imported twice.

6. Require statements in view functions (Storage.sol)

No need for requirements inside view functions. Such functions should return zeroes if the caller isn't authorized. All the require statements should be moved to user-interacting functions that call the viewers.

7. Admin variable (Token.sol)

Admin account could be declared as immutable and should have a getter or public visibility.

8. Owner variable (Token.sol)

The `owner` variable has no getter. `ownerOf()` function of the ERC721 contract returns the value of `_owners[tokenId]` mapping.

9. Too wide pragma (Token.sol)

Pragma version is set >= 0.7.0 < 0.9.0, allowing both pre-0.8 versions and post-0.8 with internal SafeMath. We recommend avoiding such a wide pragma range.

10. allowSpending() is redundant (Token.sol)

A single call of `_setApprovalForAll()` could replace multiple calls of `allowSpending()` in the `mint()` function

11. Locked native currency (YASUKE.sol)

The `buyNow()` function should return excessive EVM currency (also in `if(shouldBuyWithToken)` block).

12. Too wide pragma (YASUKE.sol)

Pragma version is set >= 0.7.0 < 0.9.0, allowing both pre-0.8 versions and post-0.8 with internal SafeMath. We recommend avoiding such a wide pragma range.

13. Testing functionality and comments (YASUKE.sol)

We recommend removing testing functionality and TODO comments before mainnet deployment.

14. No events (YASUKE.sol)

The `startAuction()` function should emit events in order to ease the use of the contract.

15. Gas optimization (YASUKE.sol)

In order to optimize gas external calls to the Storage contract should be minimized. Consider reading the data from `store` only once to a local variable.

16. ERC721.safeTransferFrom() is used (YASUKE.sol)

`buyNow()` and `placeBid()` functions may be reverted if called from the contract that hasn't implemented `onERC721Received()` method.

17. Reverting without message (YASUKE.sol)

Consider adding second parameter for `require` in `cancelAuction()` function.

18. Code for testing (models.sol, console.sol, StorageInterface.sol, YasukeInterface.sol)

The Console library should be removed before the mainnet deployment to save gas.

19. Too wide pragma (models.sol, console.sol, StorageInterface.sol, YasukeInterface.sol)

Pragma version is set >= 0.7.0 < 0.9.0, allowing both pre-0.8 versions and post-0.8 with internal

SafeMath. We recommend avoiding such a wide pragma range.

20. Gas optimisation (models.sol, console.sol, StorageInterface.sol, YasukeInterface.sol)

Structs of the Models library should be optimized by packing order.

Conclusion

Nifty Row LegalTender.sol, Storage.sol, Token.sol, YASUKE.sol, models.sol, console.sol, StorageInterface.sol, YasukeInterface.sol contracts were audited. 10 high, 4 medium, 20 low severity issues were found.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability)set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without 0xGuard prior written consent.

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts 0xGuard to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Slither output

```
pause() should be declared external:

- LegalTender.pause() (LegalTender.sol#20-22)

unpause() should be declared external:

- LegalTender.unpause() (LegalTender.sol#24-26)

mint(address,uint256) should be declared external:

- LegalTender.mint(address,uint256) (LegalTender.sol#28-30)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external

Reentrancy in Token.changeOwnership(uint256,address,address) (Token.sol#34-40):

External calls:

- safeTransferFrom(from,to,tokenId) (Token.sol#36)

State variables written after the call(s):

- allowSpending(tokenId) (Token.sol#37)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

Token.constructor(address,string,string,string)._owner (Token.sol#16) lacks a zero-check on :

  - owner = _owner (Token.sol#18)

  - issuer = _owner (Token.sol#19)

Token.changeOwnership(uint256,address,address).to (Token.sol#34) lacks a zero-check on :
```

```
- owner = to (Token.sol#38)

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

Reentrancy in Token.changeOwnership(uint256,address,address) (Token.sol#34-40):

External calls:

- safeTransferFrom(from,to,tokenId) (Token.sol#36)

State variables written after the call(s):

- owner = to (Token.sol#38)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in Token.changeOwnership(uint256,address,address) (Token.sol#34-40):

External calls:

- safeTransferFrom(from,to,tokenId) (Token.sol#36)

Event emitted after the call(s):

- allowSpending(tokenId) (Token.sol#37)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

mint(uint256) should be declared external:

- Token.mint(uint256) (Token.sol#23-28)

changeOwnership(uint256,address,address) should be declared external:

- Token.changeOwnership(uint256,address,address) (Token.sol#34-40)

getIssuer() should be declared external:
```

```
- Token.getIssuer() (Token.sol#47-49)

Reentrancy in Token.changeOwnership(uint256,address,address) (Token.sol#34-40):

External calls:

- safeTransferFrom(from,to,tokenId) (Token.sol#36)

State variables written after the call(s):

- allowSpending(tokenId) (Token.sol#37)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

Storage.setXendFeesPercentage(uint256) (Storage.sol#39-42) should emit an event for:

- xendFeesPercentage = _percentage (Storage.sol#41)

Storage.setIssuerFeesPercentage(uint256) (Storage.sol#48-51) should emit an event for:

- issuerFeesPercentage = _percentage (Storage.sol#50)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

Storage.setXendFeesAddress(address)._feesAddress (Storage.sol#57) lacks a zero-check on :

- xendFeesAddress = _feesAddress (Storage.sol#59)

Storage.setAdmin(address,address)._parent (Storage.sol#74) lacks a zero-check on :

- parent = _parent (Storage.sol#76)

Storage.setAdmin(address,address)._admin (Storage.sol#74) lacks a zero-check on :

- admin = _admin (Storage.sol#77)

- admin = _admin (Storage.sol#79)
```

Token.constructor(address,string,string,string)._owner (Token.sol#16) lacks a zero-check on :

- owner = _owner (Token.sol#18)

- issuer = _owner (Token.sol#19)

Token.changeOwnership(uint256,address,address).to (Token.sol#34) lacks a zero-check on :

- owner = to (Token.sol#38)

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

Reentrancy in Token.changeOwnership(uint256,address,address) (Token.sol#34-40):

External calls:

- safeTransferFrom(from,to,tokenId) (Token.sol#36)

State variables written after the call(s):

- owner = to (Token.sol#38)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in Token.changeOwnership(uint256,address,address) (Token.sol#34-40):

External calls:

- safeTransferFrom(from,to,tokenId) (Token.sol#36)

Event emitted after the call(s):

- allowSpending(tokenId) (Token.sol#37)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

```
Different versions of Solidity are used:

- Version used: ['>=0.4.22<0.9.0', '>=0.7.0<0.9.0', '^0.8.0', '^0.8.1']

- >=0.7.0<0.9.0 (Storage.sol#2)

- ABIEncoderV2 (Storage.sol#3)

- >=0.7.0<0.9.0 (Token.sol#2)

- ABIEncoderV2 (Token.sol#3)

- >=0.7.0<0.9.0 (interfaces/StorageInterface.sol#2)

- ABIEncoderV2 (interfaces/StorageInterface.sol#3)

- >=0.7.0<0.9.0 (library/models.sol#2)

Pragma version>=0.7.0<0.9.0 (Storage.sol#2) is too complex

Pragma version>=0.7.0<0.9.0 (Token.sol#2) is too complex

Pragma version>=0.7.0<0.9.0 (interfaces/StorageInterface.sol#2) is too complex

Pragma version>=0.7.0<0.9.0 (library/models.sol#2) is too complex

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-
versions-of-solidity

Parameter Storage.setXendFeesPercentage(uint256)._percentage (Storage.sol#39) is not in
mixedCase

Parameter Storage.setIssuerFeesPercentage(uint256)._percentage (Storage.sol#48) is not
in mixedCase

Parameter Storage.setXendFeesAddress(address)._feesAddress (Storage.sol#57) is not in
mixedCase

Parameter Storage.setAdmin(address,address)._admin (Storage.sol#74) is not in mixedCase
```

```
Parameter Storage.setAdmin(address,address)._parent (Storage.sol#74) is not in
mixedCase

Parameter Storage.setEndBlock(uint256,uint256,uint256)._endBlock (Storage.sol#195) is
not in mixedCase

Parameter Storage.setOwner(uint256,address)._owner (Storage.sol#243) is not in
mixedCase

Parameter Storage.setCancelled(uint256,uint256,bool)._cancelled (Storage.sol#255) is
not in mixedCase

Parameter Storage.setStarted(uint256,uint256,bool)._started (Storage.sol#277) is not in
mixedCase

Parameter Storage.setSellNowTriggered(uint256,uint256,bool)._sellNowTriggered
(Storage.sol#287) is not in mixedCase

Parameter Storage.setFinished(uint256,uint256,bool)._finished (Storage.sol#297) is not
in mixedCase

Parameter Storage.setInAuction(uint256,bool)._inAuction (Storage.sol#304) is not in
mixedCase

Parameter Storage.setInSale(uint256,bool)._inSale (Storage.sol#309) is not in mixedCase

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-
solidity-naming-conventions

setXendFeesPercentage(uint256) should be declared external:

- Storage.setXendFeesPercentage(uint256) (Storage.sol#39-42)

getXendFeesPercentage() should be declared external:

- Storage.getXendFeesPercentage() (Storage.sol#44-46)

setIssuerFeesPercentage(uint256) should be declared external:

- Storage.setIssuerFeesPercentage(uint256) (Storage.sol#48-51)
```

```
getIssuerFeesPercentage() should be declared external:

- Storage.getIssuerFeesPercentage() (Storage.sol#53-55)

setXendFeesAddress(address) should be declared external:

- Storage.setXendFeesAddress(address) (Storage.sol#57-60)

getXendFeesAddress() should be declared external:

- Storage.getXendFeesAddress() (Storage.sol#62-64)

getParent() should be declared external:

- Storage.getParent() (Storage.sol#66-68)

getAdmin() should be declared external:

- Storage.getAdmin() (Storage.sol#70-72)

setAdmin(address,address) should be declared external:

- Storage.setAdmin(address,address) (Storage.sol#74-83)

addToken(uint256,Token) should be declared external:

- Storage.addToken(uint256,Token) (Storage.sol#85-88)

getToken(uint256) should be declared external:

- Storage.getToken(uint256) (Storage.sol#90-92)

startAuction(Models.AuctionInfo) should be declared external:

- Storage.startAuction(Models.AuctionInfo) (Storage.sol#94-111)

startSale(uint256,uint256,bool) should be declared external:

- Storage.startSale(uint256,uint256,bool) (Storage.sol#113-119)

getNoBiddingPrice(uint256) should be declared external:
```

```
- Storage.getNoBiddingPrice(uint256) (Storage.sol#121-123)

getBuyWithToken(uint256) should be declared external:

- Storage.getBuyWithToken(uint256) (Storage.sol#125-127)

setBuyWithToken(uint256,bool) should be declared external:

- Storage.setBuyWithToken(uint256,bool) (Storage.sol#129-132)

getAuction(uint256,uint256) should be declared external:

- Storage.getAuction(uint256,uint256) (Storage.sol#134-156)

getSellNowPrice(uint256,uint256) should be declared external:

- Storage.getSellNowPrice(uint256,uint256) (Storage.sol#158-160)

getHighestBid(uint256,uint256) should be declared external:

- Storage.getHighestBid(uint256,uint256) (Storage.sol#162-164)

getMinimumBid(uint256,uint256) should be declared external:

- Storage.getMinimumBid(uint256,uint256) (Storage.sol#166-168)

setHighestBid(uint256,uint256,uint256) should be declared external:

- Storage.setHighestBid(uint256,uint256,uint256) (Storage.sol#170-177)

setHighestBidder(uint256,uint256,address) should be declared external:

- Storage.setHighestBidder(uint256,uint256,address) (Storage.sol#179-186)

getHighestBidder(uint256,uint256) should be declared external:

- Storage.getHighestBidder(uint256,uint256) (Storage.sol#188-190)

setEndBlock(uint256,uint256,uint256) should be declared external:
```

```
- Storage.setEndBlock(uint256,uint256,uint256) (Storage.sol#192-199)

getEndBlock(uint256,uint256) should be declared external:

- Storage.getEndBlock(uint256,uint256) (Storage.sol#201-203)

getStartBlock(uint256,uint256) should be declared external:

- Storage.getStartBlock(uint256,uint256) (Storage.sol#205-207)

getCurrentBlock(uint256,uint256) should be declared external:

- Storage.getCurrentBlock(uint256,uint256) (Storage.sol#209-211)

addBidder(uint256,uint256,address) should be declared external:

- Storage.addBidder(uint256,uint256,address) (Storage.sol#213-220)

getBidders(uint256,uint256) should be declared external:

- Storage.getBidders(uint256,uint256) (Storage.sol#222-224)

addBid(uint256,uint256,uint256) should be declared external:

- Storage.addBid(uint256,uint256,uint256) (Storage.sol#226-233)

getBids(uint256,uint256) should be declared external:

- Storage.getBids(uint256,uint256) (Storage.sol#235-237)

getOwner(uint256) should be declared external:

- Storage.getOwner(uint256) (Storage.sol#239-241)

setOwner(uint256,address) should be declared external:

- Storage.setOwner(uint256,address) (Storage.sol#243-246)

isCancelled(uint256,uint256) should be declared external:

- Storage.isCancelled(uint256,uint256) (Storage.sol#248-250)
```

setCancelled(uint256,uint256,bool) should be declared external:

- Storage.setCancelled(uint256,uint256,bool) (Storage.sol#252-260)

isStarted(uint256,uint256) should be declared external:

- Storage.isStarted(uint256,uint256) (Storage.sol#262-264)

isFinished(uint256,uint256) should be declared external:

- Storage.isFinished(uint256,uint256) (Storage.sol#266-268)

isSellNowTriggered(uint256,uint256) should be declared external:

- Storage.isSellNowTriggered(uint256,uint256) (Storage.sol#270-272)

setStarted(uint256,uint256,bool) should be declared external:

- Storage.setStarted(uint256,uint256,bool) (Storage.sol#274-282)

setSellNowTriggered(uint256,uint256,bool) should be declared external:

- Storage.setSellNowTriggered(uint256,uint256,bool) (Storage.sol#284-292)

setFinished(uint256,uint256,bool) should be declared external:

- Storage.setFinished(uint256,uint256,bool) (Storage.sol#294-302)

setInAuction(uint256,bool) should be declared external:

- Storage.setInAuction(uint256,bool) (Storage.sol#304-307)

setInSale(uint256,bool) should be declared external:

- Storage.setInSale(uint256,bool) (Storage.sol#309-312)

setNoBiddingPrice(uint256,uint256) should be declared external:

- Storage.setNoBiddingPrice(uint256,uint256) (Storage.sol#314-317)

```
isInAuction(uint256) should be declared external:

- Storage.isInAuction(uint256) (Storage.sol#319-321)

isInSale(uint256) should be declared external:

- Storage.isInSale(uint256) (Storage.sol#323-325)

echo() should be declared external:

- Storage.echo() (Storage.sol#327-331)

mint(uint256) should be declared external:

- Token.mint(uint256) (Token.sol#23-28)

changeOwnership(uint256,address,address) should be declared external:

- Token.changeOwnership(uint256,address,address) (Token.sol#34-40)

getIssuer() should be declared external:

- Token.getIssuer() (Token.sol#47-49)

Yasuke._withdrawOwner(uint256,uint256) (YASUKE.sol#230-269) sends eth to arbitrary user

Dangerous calls:

- (sent,None) = address(t.getIssuer()).call{value: issuerFees}() (YASUKE.sol#258)

- (sent,None) = address(store.getXendFeesAddress()).call{value: xendFees}()
(YASUKE.sol#263)

- (sent,None) = address(owner).call{value: withdrawalAmount}() (YASUKE.sol#267)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-
send-ether-to-arbitrary-destinations[0m

Reentrancy in Token.changeOwnership(uint256,address,address) (Token.sol#34-40):

External calls:
```

```
- safeTransferFrom(from,to,tokenId) (Token.sol#36)

State variables written after the call(s):

- allowSpending(tokenId) (Token.sol#37)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

Yasuke.placeBid(uint256,uint256).sent_scope_0 (YASUKE.sol#175) is a local variable never initialized

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

Storage.setXendFeesPercentage(uint256) (Storage.sol#39-42) should emit an event for:

- xendFeesPercentage = _percentage (Storage.sol#41)

Storage.setIssuerFeesPercentage(uint256) (Storage.sol#48-51) should emit an event for:

- issuerFeesPercentage = _percentage (Storage.sol#50)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

Storage.setXendFeesAddress(address)._feesAddress (Storage.sol#57) lacks a zero-check on :

- xendFeesAddress = _feesAddress (Storage.sol#59)

Storage.setAdmin(address,address)._parent (Storage.sol#74) lacks a zero-check on :

- parent = _parent (Storage.sol#76)

Storage.setAdmin(address,address)._admin (Storage.sol#74) lacks a zero-check on :

- admin = _admin (Storage.sol#77)

- admin = _admin (Storage.sol#79)
```

```
Token.constructor(address,string,string,string)._owner (Token.sol#16) lacks a zero-check on :

- owner = _owner (Token.sol#18)

- issuer = _owner (Token.sol#19)

Token.changeOwnership(uint256,address,address).to (Token.sol#34) lacks a zero-check on :

- owner = to (Token.sol#38)

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

Variable 'Yasuke.placeBid(uint256,uint256).sent (YASUKE.sol#157)' in Yasuke.placeBid(uint256,uint256) (YASUKE.sol#141-189) potentially used before declaration: (sent) = address(highestBidder).call{value: highestBid}() (YASUKE.sol#175)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

Reentrancy in Token.changeOwnership(uint256,address,address) (Token.sol#34-40):

External calls:

- safeTransferFrom(from,to,tokenId) (Token.sol#36)

State variables written after the call(s):

- owner = to (Token.sol#38)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in Yasuke._withdrawal(uint256,uint256,bool) (YASUKE.sol#191-228):

External calls:

- _withdrawOwner(tokenId,auctionId) (YASUKE.sol#210)
```

```
        - store.setHighestBid(tokenId,auctionId,0) (YASUKE.sol#239)

        - (sent,None) = address(t.getIssuer()).call{value: issuerFees}() (YASUKE.sol#258)

        - (sent,None) = address(store.getXendFeesAddress()).call{value: xendFees}()
(YASUKE.sol#263)

        - (sent,None) = address(owner).call{value: withdrawalAmount}() (YASUKE.sol#267)

    - require(bool,string)(t.changeOwnership(tokenId,owner,highestBidder),CNCO)
(YASUKE.sol#213)

    - _withdrawOwner(tokenId,auctionId) (YASUKE.sol#217)

        - store.setHighestBid(tokenId,auctionId,0) (YASUKE.sol#239)

        - (sent,None) = address(t.getIssuer()).call{value: issuerFees}() (YASUKE.sol#258)

        - (sent,None) = address(store.getXendFeesAddress()).call{value: xendFees}()
(YASUKE.sol#263)

        - (sent,None) = address(owner).call{value: withdrawalAmount}() (YASUKE.sol#267)

    - store.setInAuction(tokenId,false) (YASUKE.sol#219)

    - store.setOwner(tokenId,highestBidder) (YASUKE.sol#220)

    - store.setFinished(tokenId,auctionId,true) (YASUKE.sol#221)

    - store.setStarted(tokenId,auctionId,false) (YASUKE.sol#222)

    - store.setHighestBidder(tokenId,auctionId,address(0)) (YASUKE.sol#223)

    - store.setHighestBid(tokenId,auctionId,0) (YASUKE.sol#224)

External calls sending eth:

    - _withdrawOwner(tokenId,auctionId) (YASUKE.sol#210)

        - (sent,None) = address(t.getIssuer()).call{value: issuerFees}() (YASUKE.sol#258)
```

```
- (sent,None) = address(store.getXendFeesAddress()).call{value: xendFees}()
(YASUKE.sol#263)

- (sent,None) = address(owner).call{value: withdrawalAmount}() (YASUKE.sol#267)

- _withdrawOwner(tokenId,auctionId) (YASUKE.sol#217)

- (sent,None) = address(t.getIssuer()).call{value: issuerFees}() (YASUKE.sol#258)

- (sent,None) = address(store.getXendFeesAddress()).call{value: xendFees}()
(YASUKE.sol#263)

- (sent,None) = address(owner).call{value: withdrawalAmount}() (YASUKE.sol#267)

Event emitted after the call(s):

- LogWithdrawal(msg.sender,tokenId,auctionId) (YASUKE.sol#227)

Reentrancy in Yasuke.buyNow(uint256) (YASUKE.sol#93-124):

External calls:

- store.setInSale(tokenId,false) (YASUKE.sol#110)

- store.setNoBiddingPrice(tokenId,0) (YASUKE.sol#111)

- require(bool,string)(t.changeOwnership(tokenId,owner,buyer),CNCO) (YASUKE.sol#113)

- result = IERC20(legalTender).transferFrom(msg.sender,burnAddress,noBiddingPrice)
(YASUKE.sol#116)

- (sent) = address(msg.sender).call{value: noBiddingPrice}() (YASUKE.sol#119)

External calls sending eth:

- (sent) = address(msg.sender).call{value: noBiddingPrice}() (YASUKE.sol#119)

Event emitted after the call(s):

- Sold(owner,msg.sender,tokenId,noBiddingPrice) (YASUKE.sol#123)
```

```
Reentrancy in Yasuke.cancelAuction(uint256,uint256) (YASUKE.sol#276-281):

External calls:

- store.setCancelled(tokenId,auctionId,true) (YASUKE.sol#279)

Event emitted after the call(s):

- LogCanceled() (YASUKE.sol#280)

Reentrancy in Token.changeOwnership(uint256,address,address) (Token.sol#34-40):

External calls:

- safeTransferFrom(from,to,tokenId) (Token.sol#36)

Event emitted after the call(s):

- allowSpending(tokenId) (Token.sol#37)

Reentrancy in Yasuke.placeBid(uint256,uint256) (YASUKE.sol#141-189):

External calls:

- store.setEndBlock(tokenId,auctionId,block.number - 1) (YASUKE.sol#152)

- (sent) = address(msg.sender).call{value: difference}() (YASUKE.sol#157)

- store.setSellNowTriggered(tokenId,auctionId,true) (YASUKE.sol#163)

- (sent) = address(highestBidder).call{value: highestBid}() (YASUKE.sol#175)

- store.setHighestBidder(tokenId,auctionId,msg.sender) (YASUKE.sol#179)

- store.setHighestBid(tokenId,auctionId,newBid) (YASUKE.sol#180)

- store.addBidder(tokenId,auctionId,msg.sender) (YASUKE.sol#181)

- store.addBid(tokenId,auctionId,newBid) (YASUKE.sol#182)
```

```
External calls sending eth:
- (sent) = address(msg.sender).call{value: difference}() (YASUKE.sol#157)
- (sent) = address(highestBidder).call{value: highestBid}() (YASUKE.sol#175)
Event emitted after the call(s):
- LogBid(msg.sender,newBid) (YASUKE.sol#184)
Reentrancy in Yasuke.placeBid(uint256,uint256) (YASUKE.sol#141-189):
External calls:
- store.setEndBlock(tokenId,auctionId,block.number - 1) (YASUKE.sol#152)
- (sent) = address(msg.sender).call{value: difference}() (YASUKE.sol#157)
- store.setSellNowTriggered(tokenId,auctionId,true) (YASUKE.sol#163)
- (sent) = address(highestBidder).call{value: highestBid}() (YASUKE.sol#175)
- store.setHighestBidder(tokenId,auctionId,msg.sender) (YASUKE.sol#179)
- store.setHighestBid(tokenId,auctionId,newBid) (YASUKE.sol#180)
- store.addBidder(tokenId,auctionId,msg.sender) (YASUKE.sol#181)
- store.addBid(tokenId,auctionId,newBid) (YASUKE.sol#182)
- _withdrawal(tokenId,auctionId,false) (YASUKE.sol#187)
- store.setHighestBid(tokenId,auctionId,0) (YASUKE.sol#239)
- require(bool,string)(t.changeOwnership(tokenId,owner,highestBidder),CNCO)
(YASUKE.sol#213)
- store.setInAuction(tokenId,false) (YASUKE.sol#219)
- store.setOwner(tokenId,highestBidder) (YASUKE.sol#220)
```

```
    - (sent,None) = address(t.getIssuer()).call{value: issuerFees}() (YASUKE.sol#258)

    - store.setFinished(tokenId,auctionId,true) (YASUKE.sol#221)

    - store.setStarted(tokenId,auctionId,false) (YASUKE.sol#222)

    - store.setHighestBidder(tokenId,auctionId,address(0)) (YASUKE.sol#223)

    - store.setHighestBid(tokenId,auctionId,0) (YASUKE.sol#224)

    - (sent,None) = address(store.getXendFeesAddress()).call{value: xendFees}()
(YASUKE.sol#263)

    - (sent,None) = address(owner).call{value: withdrawalAmount}() (YASUKE.sol#267)

External calls sending eth:

 - (sent) = address(msg.sender).call{value: difference}() (YASUKE.sol#157)

 - (sent) = address(highestBidder).call{value: highestBid}() (YASUKE.sol#175)

 - _withdrawal(tokenId,auctionId,false) (YASUKE.sol#187)

    - (sent,None) = address(t.getIssuer()).call{value: issuerFees}() (YASUKE.sol#258)

    - (sent,None) = address(store.getXendFeesAddress()).call{value: xendFees}()
(YASUKE.sol#263)

    - (sent,None) = address(owner).call{value: withdrawalAmount}() (YASUKE.sol#267)

Event emitted after the call(s):

 - LogWithdrawal(msg.sender,tokenId,auctionId) (YASUKE.sol#227)

    - _withdrawal(tokenId,auctionId,false) (YASUKE.sol#187)

Reentrancy in Yasuke.sellNow(uint256,uint256,bool) (YASUKE.sol#126-139):

External calls:

 - store.startSale(tokenId,price,withToken) (YASUKE.sol#136)
```

```
Event emitted after the call(s):

 - OnSale(msg.sender,tokenId) (YASUKE.sol#138)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Different versions of Solidity are used:

 - Version used: ['>=0.4.22<0.9.0', '>=0.7.0<0.9.0', '^0.8.0', '^0.8.1']

 - >=0.7.0<0.9.0 (Storage.sol#2)

 - ABIEncoderV2 (Storage.sol#3)

 - >=0.7.0<0.9.0 (Token.sol#2)

 - ABIEncoderV2 (Token.sol#3)

 - >=0.7.0<0.9.0 (YASUKE.sol#2)

 - ABIEncoderV2 (YASUKE.sol#3)

 - >=0.7.0<0.9.0 (interfaces/StorageInterface.sol#2)

 - ABIEncoderV2 (interfaces/StorageInterface.sol#3)

 - >=0.7.0<0.9.0 (interfaces/YasukeInterface.sol#2)

 - ABIEncoderV2 (interfaces/YasukeInterface.sol#3)

 - >=0.7.0<0.9.0 (library/models.sol#2)

 - ABIEncoderV2 (library/models.sol#3)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Pragma version>=0.7.0<0.9.0 (Storage.sol#2) is too complex
```

```
Pragma version>=0.7.0<0.9.0 (Token.sol#2) is too complex

Pragma version>=0.7.0<0.9.0 (YASUKE.sol#2) is too complex

Pragma version>=0.7.0<0.9.0 (interfaces/StorageInterface.sol#2) is too complex

Pragma version>=0.7.0<0.9.0 (interfaces/YasukeInterface.sol#2) is too complex

Pragma version>=0.7.0<0.9.0 (library/models.sol#2) is too complex

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-
versions-of-solidity

Low level call in Yasuke.buyNow(uint256) (YASUKE.sol#93-124):

- (sent) = address(msg.sender).call{value: noBiddingPrice}() (YASUKE.sol#119)

Low level call in Yasuke.placeBid(uint256,uint256) (YASUKE.sol#141-189):

- (sent) = address(msg.sender).call{value: difference}() (YASUKE.sol#157)

- (sent) = address(highestBidder).call{value: highestBid}() (YASUKE.sol#175)

Low level call in Yasuke._withdrawOwner(uint256,uint256) (YASUKE.sol#230-269):

- (sent,None) = address(t.getIssuer()).call{value: issuerFees}() (YASUKE.sol#258)

- (sent,None) = address(store.getXendFeesAddress()).call{value: xendFees}()
(YASUKE.sol#263)

- (sent,None) = address(owner).call{value: withdrawalAmount}() (YASUKE.sol#267)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-
calls

Parameter Storage.setXendFeesPercentage(uint256)._percentage (Storage.sol#39) is not in
mixedCase

Parameter Storage.setIssuerFeesPercentage(uint256)._percentage (Storage.sol#48) is not
in mixedCase
```

```
Parameter Storage.setXendFeesAddress(address)._feesAddress (Storage.sol#57) is not in
mixedCase

Parameter Storage.setAdmin(address,address)._admin (Storage.sol#74) is not in mixedCase

Parameter Storage.setAdmin(address,address)._parent (Storage.sol#74) is not in
mixedCase

Parameter Storage.setEndBlock(uint256,uint256,uint256)._endBlock (Storage.sol#195) is
not in mixedCase

Parameter Storage.setOwner(uint256,address)._owner (Storage.sol#243) is not in
mixedCase

Parameter Storage.setCancelled(uint256,uint256,bool)._cancelled (Storage.sol#255) is
not in mixedCase

Parameter Storage.setStarted(uint256,uint256,bool)._started (Storage.sol#277) is not in
mixedCase

Parameter Storage.setSellNowTriggered(uint256,uint256,bool)._sellNowTriggered
(Storage.sol#287) is not in mixedCase

Parameter Storage.setFinished(uint256,uint256,bool)._finished (Storage.sol#297) is not
in mixedCase

Parameter Storage.setInAuction(uint256,bool)._inAuction (Storage.sol#304) is not in
mixedCase

Parameter Storage.setInSale(uint256,bool)._inSale (Storage.sol#309) is not in mixedCase

Parameter Yasuke.issueToken(uint256,address,string,string,string)._uri (YASUKE.sol#77)
is not in mixedCase

Parameter Yasuke.issueToken(uint256,address,string,string,string)._name (YASUKE.sol#78)
is not in mixedCase

Parameter Yasuke.issueToken(uint256,address,string,string,string)._symbol
(YASUKE.sol#79) is not in mixedCase

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-
```

```
solidity-naming-conventions

Yasuke.slitherConstructorVariables() (YASUKE.sol#14-315) uses literals with too many
digits:

- burnAddress = 0x000000000000000000000000000000000000dEaD (YASUKE.sol#21)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-
digits

Yasuke.burnAddress (YASUKE.sol#21) should be constant

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-
variables-that-could-be-declared-constant

setXendFeesPercentage(uint256) should be declared external:

- Storage.setXendFeesPercentage(uint256) (Storage.sol#39-42)

getXendFeesPercentage() should be declared external:

- Storage.getXendFeesPercentage() (Storage.sol#44-46)

setIssuerFeesPercentage(uint256) should be declared external:

- Storage.setIssuerFeesPercentage(uint256) (Storage.sol#48-51)

getIssuerFeesPercentage() should be declared external:

- Storage.getIssuerFeesPercentage() (Storage.sol#53-55)

setXendFeesAddress(address) should be declared external:

- Storage.setXendFeesAddress(address) (Storage.sol#57-60)

getXendFeesAddress() should be declared external:

- Storage.getXendFeesAddress() (Storage.sol#62-64)

getParent() should be declared external:
```

```
- Storage.getParent() (Storage.sol#66-68)

getAdmin() should be declared external:

- Storage.getAdmin() (Storage.sol#70-72)

setAdmin(address,address) should be declared external:

- Storage.setAdmin(address,address) (Storage.sol#74-83)

addToken(uint256,Token) should be declared external:

- Storage.addToken(uint256,Token) (Storage.sol#85-88)

getToken(uint256) should be declared external:

- Storage.getToken(uint256) (Storage.sol#90-92)

startAuction(Models.AuctionInfo) should be declared external:

- Storage.startAuction(Models.AuctionInfo) (Storage.sol#94-111)

startSale(uint256,uint256,bool) should be declared external:

- Storage.startSale(uint256,uint256,bool) (Storage.sol#113-119)

getNoBiddingPrice(uint256) should be declared external:

- Storage.getNoBiddingPrice(uint256) (Storage.sol#121-123)

getBuyWithToken(uint256) should be declared external:

- Storage.getBuyWithToken(uint256) (Storage.sol#125-127)

setBuyWithToken(uint256,bool) should be declared external:

- Storage.setBuyWithToken(uint256,bool) (Storage.sol#129-132)

getAuction(uint256,uint256) should be declared external:

- Storage.getAuction(uint256,uint256) (Storage.sol#134-156)
```

```
getSellNowPrice(uint256,uint256) should be declared external:

- Storage.getSellNowPrice(uint256,uint256) (Storage.sol#158-160)

getHighestBid(uint256,uint256) should be declared external:

- Storage.getHighestBid(uint256,uint256) (Storage.sol#162-164)

getMinimumBid(uint256,uint256) should be declared external:

- Storage.getMinimumBid(uint256,uint256) (Storage.sol#166-168)

setHighestBid(uint256,uint256,uint256) should be declared external:

- Storage.setHighestBid(uint256,uint256,uint256) (Storage.sol#170-177)

setHighestBidder(uint256,uint256,address) should be declared external:

- Storage.setHighestBidder(uint256,uint256,address) (Storage.sol#179-186)

getHighestBidder(uint256,uint256) should be declared external:

- Storage.getHighestBidder(uint256,uint256) (Storage.sol#188-190)

setEndBlock(uint256,uint256,uint256) should be declared external:

- Storage.setEndBlock(uint256,uint256,uint256) (Storage.sol#192-199)

getEndBlock(uint256,uint256) should be declared external:

- Storage.getEndBlock(uint256,uint256) (Storage.sol#201-203)

getStartBlock(uint256,uint256) should be declared external:

- Storage.getStartBlock(uint256,uint256) (Storage.sol#205-207)

getCurrentBlock(uint256,uint256) should be declared external:

- Storage.getCurrentBlock(uint256,uint256) (Storage.sol#209-211)
```

```
addBidder(uint256,uint256,address) should be declared external:

- Storage.addBidder(uint256,uint256,address) (Storage.sol#213-220)

getBidders(uint256,uint256) should be declared external:

- Storage.getBidders(uint256,uint256) (Storage.sol#222-224)

addBid(uint256,uint256,uint256) should be declared external:

- Storage.addBid(uint256,uint256,uint256) (Storage.sol#226-233)

getBids(uint256,uint256) should be declared external:

- Storage.getBids(uint256,uint256) (Storage.sol#235-237)

getOwner(uint256) should be declared external:

- Storage.getOwner(uint256) (Storage.sol#239-241)

setOwner(uint256,address) should be declared external:

- Storage.setOwner(uint256,address) (Storage.sol#243-246)

isCancelled(uint256,uint256) should be declared external:

- Storage.isCancelled(uint256,uint256) (Storage.sol#248-250)

setCancelled(uint256,uint256,bool) should be declared external:

- Storage.setCancelled(uint256,uint256,bool) (Storage.sol#252-260)

isStarted(uint256,uint256) should be declared external:

- Storage.isStarted(uint256,uint256) (Storage.sol#262-264)

isFinished(uint256,uint256) should be declared external:

- Storage.isFinished(uint256,uint256) (Storage.sol#266-268)

isSellNowTriggered(uint256,uint256) should be declared external:
```

```
- Storage.isSellNowTriggered(uint256,uint256) (Storage.sol#270-272)

setStarted(uint256,uint256,bool) should be declared external:

- Storage.setStarted(uint256,uint256,bool) (Storage.sol#274-282)

setSellNowTriggered(uint256,uint256,bool) should be declared external:

- Storage.setSellNowTriggered(uint256,uint256,bool) (Storage.sol#284-292)

setFinished(uint256,uint256,bool) should be declared external:

- Storage.setFinished(uint256,uint256,bool) (Storage.sol#294-302)

setInAuction(uint256,bool) should be declared external:

- Storage.setInAuction(uint256,bool) (Storage.sol#304-307)

setInSale(uint256,bool) should be declared external:

- Storage.setInSale(uint256,bool) (Storage.sol#309-312)

setNoBiddingPrice(uint256,uint256) should be declared external:

- Storage.setNoBiddingPrice(uint256,uint256) (Storage.sol#314-317)

isInAuction(uint256) should be declared external:

- Storage.isInAuction(uint256) (Storage.sol#319-321)

isInSale(uint256) should be declared external:

- Storage.isInSale(uint256) (Storage.sol#323-325)

echo() should be declared external:

- Storage.echo() (Storage.sol#327-331)

mint(uint256) should be declared external:
```

```
- Token.mint(uint256) (Token.sol#23-28)

changeOwnership(uint256,address,address) should be declared external:

- Token.changeOwnership(uint256,address,address) (Token.sol#34-40)

getIssuer() should be declared external:

- Token.getIssuer() (Token.sol#47-49)

upgrade(address) should be declared external:

- Yasuke.upgrade(address) (YASUKE.sol#30-33)

testUpgrade() should be declared external:

- Yasuke.testUpgrade() (YASUKE.sol#35-38)

startAuction(uint256,uint256,uint256,uint256,uint256,uint256,uint256) should be
declared external:

- Yasuke.startAuction(uint256,uint256,uint256,uint256,uint256,uint256,uint256)
(YASUKE.sol#40-72)

issueToken(uint256,address,string,string,string) should be declared external:

- Yasuke.issueToken(uint256,address,string,string,string) (YASUKE.sol#74-85)

endBid(uint256,uint256) should be declared external:

- Yasuke.endBid(uint256,uint256) (YASUKE.sol#87-91)

buyNow(uint256) should be declared external:

- Yasuke.buyNow(uint256) (YASUKE.sol#93-124)

sellNow(uint256,uint256,bool) should be declared external:

- Yasuke.sellNow(uint256,uint256,bool) (YASUKE.sol#126-139)

placeBid(uint256,uint256) should be declared external:
```

```
- Yasuke.placeBid(uint256,uint256) (YASUKE.sol#141-189)

withdraw(uint256,uint256) should be declared external:

- Yasuke.withdraw(uint256,uint256) (YASUKE.sol#271-273)

cancelAuction(uint256,uint256) should be declared external:

- Yasuke.cancelAuction(uint256,uint256) (YASUKE.sol#276-281)

getTokenInfo(uint256) should be declared external:

- Yasuke.getTokenInfo(uint256) (YASUKE.sol#283-301)

getAuctionInfo(uint256,uint256) should be declared external:

- Yasuke.getAuctionInfo(uint256,uint256) (YASUKE.sol#303-306)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-
function-that-could-be-declared-external
```

0x Guard