

Audit Report for The Smuggleverse LLC

July 2022

Version 1.2

Prepared by

Extropy.IO

Email	Telephone	
info@extropy.io	+44 1865261424	

1. Executive Summary

Extropy was contracted to conduct a code review and vulnerability assessment of the project. The review was carried out between 15th and 21st June 2022.

A retest was done on 01/07/22, and a further retest on 08/07/22.

This report is of the second retest.

All of the major and low risk issues have now been closed

Assessment Summary

All of the major issues have now been closed..

Issues count summary

Stage	Critical	High	Medium	Low	Info	Total
Initial Report	0	1	3	10	18	32

Retest 01/07/2022 Issues count summary

Stage	Critical	High	Medium	Low	Info	Total
Retest Report	0	0	1	0	0	1

Retest 08/07/2022 Issues count summary

Stage	Critical	High	Medium	Low	Info	Total
Retest Report	0	0	0	0	0	0

Using This Report

To facilitate the dissemination of the information within this report throughout your organisation, this document has been divided into the following clearly marked and separable sections.

- Executive Summary
- Management level, strategic overview of the assessment and the risks posed to the business
- Technical Summary
- An overview of the assessment from a more technical perspective, including a defined scope and any caveats which may apply
- Technical Findings
- Detailed discussion (including evidence and recommendations) for each individual security issue which was identified
- Methodologies
- Audit process and tools used

Disclaimer

The audit makes no statements or warranty about utility of the code, safety of the code, suitability of the business model, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only

Client Confidentiality

This document contains Client Confidential information and may not be copied without written permission.

Proprietary Information

The content of this document should be considered proprietary information.

Extropy gives permission to copy this report for the purposes of disseminating information within your organisation or any regulatory agency.

Technical Findings

The remainder of this document is technical in nature and provides additional detail about the items already discussed, for the purposes of remediation and risk assessment.

Access Control Missing

Risk	High
Affects	NFT721
	Cutting

There is no access control on the `burn` function, meaning that anyone can burn any token.

In functions `cut` in Cutting contract, there is no check against `msg.sender`, yet this calls burn and mint, it is unclear from the code whether this is the intended behaviour.

In UpgradeNFT, the function `upgrade` has no access control, and can therefore allow anyone to upgrade any NFT.

Recommendation

Restrict access using a modifier or for finer granularity use Open Zeppelin's AccessControl contract.

References

<https://docs.openzeppelin.com/contracts/2.x/access-control>

Retest 01/07/2022

Status	Closed
--------	--------

Added access control for burning functionality

The added access control requires that the caller is the owner or approved address:

```
function burn(uint256 tokenId) external {
    require(_isApprovedOrOwner(_msgSender(), tokenId), "NFT721: caller is
not owner nor approved");
```

```
_burn(tokenId);  
...
```

Reentrancy

Risk	Medium
Affects	Marketplace.sol

A reentrancy attack could cause `safeTransferFrom` to be called repeatedly. This occurs multiple times in this contract.

Recommendation

Follow the 'check effects interaction' [pattern](#) when interacting with other contracts, or use a re entrancy guard.

Place value transfer functions after updating storage.

References

<https://docs.soliditylang.org/en/v0.4.21/security-considerations.html#re-entrancy>
https://fravoll.github.io/solidity-patterns/checks_effects_interactions.html

Retest 01/07/2022

Status	Open
--------	------

Fixed in two functions, `buyToken` and `cancelSellToken`

Transferring an NFT or ERC20 is not considered reentrancy because their addresses are assigned in the constructor, i.e. it is not making an external call to an user provided address.

The code change in `buyToken()` has now introduced a new issue, the data is deleted from storage before being used, and it is not copied in memory:

```
_asksMap.remove(_tokenId);  
_userSellingTokens[_tokenSellers[_tokenId]].remove(_tokenId);  
delete _tokenSellers[_tokenId];  
  
quoteErc20.safeTransferFrom(msg.sender, _tokenSellers[_tokenId],  
netSaleValue);  
nft.safeTransferFrom(address(this), msg.sender, _tokenId);  
quoteErc20.safeTransferFrom(msg.sender, royaltiesReceiver, royaltiesAmount);
```

```
emit Trade(_tokenSellers[_tokenId], msg.sender, _tokenId, price);
```

The result is that `quoteErc20` tokens will be sent to the `0x0` address, which will be emitted by the event. To fix this, store the variables needed to call `safeTransferFrom` in a temporary variable before deleting them from storage.

Retest 08/07/2022

Status	Closed
--------	--------

Failing test cases: owner query for nonexistent token

Risk	Medium
Affects	NFT721

The test `should cut the nft without joint` fails with: `reverted with reason string 'ERC721: owner query for nonexistent token'.`

The test `should upgrade` fails with: `reverted with reason string 'not in list'.`

The test `should cut the nft without joint` fails with: `reverted with reason string 'not in list'.`

Failing test case: Token is not approved to stack

Risk	Medium
Affects	Stacking

The test `should be a valid stacking` fails with: `reverted with reason string 'Token is not approved to stack'.`

The test `should revert with weight mismatch` fails with: `reverted with reason string 'Token is not approved to stack'.`

The test `should stack the nft` fails with: `reverted with reason string 'Token is not approved to stack'.`

Recommendation

Correct contract or test case.

Unbound Loop

Risk	Low
Affects	Marketplace

The number of loops inside `getAsks()` and `getAsksByUser()` can grow indefinitely, eventually reverting with an "out of gas" exception. Each new order creates a new key/value pair in the mapping, this is expected to grow. Furthermore, filled and deleted orders are deleted from the mappings, which would return an empty object in the array.

Recommendation

Ideally, orders should be accessed using the events storage, however if a read function is needed, it would be best to keep a public "counter" variable (already available via function `getAskLength()`) that keeps track of the number of items in `_userSellingTokens` and `_asksMap`, so that the loop isn't done in Solidity but by external code, in which case the getter function would simply return a single object given a specific numerical id.

Retest 01/07/2022

Status	Closed
--------	--------

Removed these two functions from the contract and added the `getAsk` function with index value.

A getter function was created and the two affected functions were removed.

For reference:

```
function getAsk(uint256 index) external view returns (uint256 tokenId, uint256 price){
    (tokenId, price) = _asksMap.at(index);
}
```

Irrelevant Guard Condition

Risk	Low
Affects	Marketplace

The follownig `require()` condition inside the `buyToken()` function is completely irrelevant:

```
require(msg.sender != address(0) && msg.sender != address(this), 'Wrong msg sender');
```

`msg.sender` can never be `0x0` because nobody owns it, and it can never be the own contract's address either for the same reason.

Another irrelevant condition can be found in `readyToSellToken()` function:

```
require(msg.sender == nft.ownerOf(_tokenId), 'Only Token Owner can sell token');
```

The reason why it is superfluous is because the NFT needs to be transferred to the marketplace contract, which means that the `safeTransferFrom()` function will revert if the caller is not the legitimate owner.

Recommendation

Remove the affected line of code.

Retest 01/07/2022

Status	Closed
--------	--------

Removed the conditions mentioned.

Code removed as per recommendations.

Centralized Code

Risk	Low
Affects	UpgradeNFT

It appears as if `UpgradeNFT` superseeds the token metadata from `NFT721`, although this is not enforced at the contract level, that seems to be its purpose. Perhaps a frontend reads any modified metadata directly from `UpgradeNFT` rather than `NFT721`.

Functions `setUpgrade()` and `updateUpgrade()` are overwriting the metadata and adding it to the `nftMeta` mapping.

Recommendation

Remove any points of centralization.

Retest 01/07/2022

Status	Closed
--------	--------

There is no centralized code. The functionality is like this. Admin can set a new upgrade (using function setUpgrade) indication in contract for a particular nft. There is nothing we update with the actual nft. Just to know the update there for the user who holds the nft. Then the holding user can upgrade the nft by using the upgrade function. Also just removing the updateUpgrade function, there is no longer needed.

- Function `updateUpgrade` was removed.

No further changes needed.

Poor naming convention and NatSpec comments

Risk	Low
Affects	UpgradeNFT

Functions `updateUpgrade`, `setUpgrade` and `upgrade` have very similar names and the NatSpec comments are not very informative.

Recommendation

Rename functions and variables to avoid ambiguity.

Retest 01/07/2022

Status	Closed
--------	--------

Fixed

Function `updateUpgrade` was removed.

Incorrect error message returned

Risk	Low
Affects	NFT721.sol

The error message 'weight exceed' doesn't align with expected `strain weight balance exceed` in the test.

Recommendation

Align the error messages.

Retest 01/07/2022

Status	Closed
--------	--------

Fixed

Missing `mintMusic` and `mintSeed` functions

Risk	Low
Affects	NFT721.sol

The test calls on two functions that don't exist: `mintMusic` and `mintSeed`.

Recommendation

Remove tests or implement functions.

Retest 01/07/2022

Status	Closed
--------	--------

Fixed

The function calls where removed from the test files

Missing BUSD smart contract

Risk	Low
Affects	Multiple

The BUSD contract is missing so tests cannot be run, affecting `2_Marketplace.js` and `6_auction.js`

Recommendation

Include contract.

Retest 01/07/2022

Status	Closed
--------	--------

Added this contract. Please note this contract is only for testing purposes.

Missing zero-check on variable

Risk	Low
Affects	NFT721
	Cutting

In NFT721 `_royaltiesReceipient` can be a zero address causing the royalties to potentially be lost.

In Cutting `_adminWallet` can be a zero address causing the royalties to potentially be lost.

Recommendation

Check for a valid address.

Retest 01/07/2022

Status	Closed
--------	--------

Fixed

Added in `NFT721`:

```
require(_royaltiesReceipient != address(0) && _royaltiesReceipient !=  
address(this), "royalty address not valid");
```

Added in `Cutting`:

```
require(_adminWallet != address(0) && _adminWallet != address(this), "admin  
address not valid");
```

Unnecessary function

Risk	Low
Affects	MarketPlaceAuction

Line 56:

`addAuction()` only calls `_makeOrder()`

Recomendation

Remove `addAuction()` and make `_makeOrder` public.

```
function _makeOrder(  
    uint256 _tokenId,  
    uint256 _startPrice,  
    uint256 _endTime  
) public {
```

Retest 01/07/2022

Status	Closed
--------	--------

Renamed function name to addAuction

function `_makeOrder` was renamed to `addAuction` and function `addAuction` was removed.

Unnecessary usage of storage

Risk	Low
Affects	MarketPlaceAuction

This applies to `claim()` on Line 113 and `bid()` on Line 85.

In `bid` can instead load it to memory.

```
Order memory o = orderInfo[_order];
```

This can be operated on memory then only written to storage on line 102.

```
orderInfo[_order].lastBidder. = msg.sender  
orderInfo[_order].lastBidPrice. = _price
```

Retest 01/07/2022

Status	Closed
--------	--------

Fixed

Duplicated error message

Risk	Informational
Affects	Cutting
	MarketPlace

Line 76/77 cutting.sol

```
require(msg.sender == cutOwner, "Cutter is not an owner");  
require(msg.sender == feeOwner, "Cutter is not an owner");
```

Line 39/40 in MarketPlace.sol

```
require(_nftAddress != address(0) && _nftAddress != address(this), "address  
not valid");  
require(_quoteErc20Address != address(0) && _quoteErc20Address !=  
address(this), "address not valid");
```

Can not tell what caused a given revert.

Recomendation

Combine both `require()` into a single statment:

```
require(msg.sender == cutOwner && msg.sender == feeOwner, "Cutter is not an  
owner");
```

or create an explicit error messages for each revert:

```
require(msg.sender == cutOwner, "Caller is not an owner");  
require(msg.sender == feeOwner, "Caller is not a cutter");
```

Retest 01/07/2022

Status	Closed
--------	--------

Changed the messages

Use constants or immutable where possible

Risk	Informational
Affects	Stacking
	NFT721
	Cutting

`weightOfFee` in Stacking could be constant as it is never updated (In this case `weightOfFee` could be removed as it isn't being used.)

In NFT721 the following variables should be set as constants, this also assures that the contract can be upgraded:

```
uint256 public jointWeight = 1;
uint256 public maxMusic = 100000;
uint256 public maxSeed = 34;
```

Recommendation

Make such variables `constant`, like so:

```
uint256 public constant weightOfFee = 1;
```

```
uint256 public constant jointWeight = 1;
uint256 public constant maxMusic = 100000;
uint256 public constant maxSeed = 34;
```

In Cutting, declare `adminWallet` immutable since it is setup in the constructor.

Retest 01/07/2022

Status	Closed
--------	--------

Fixed

Variables were declared `constant` as recommended.

Unnecessary intialisation

Risk	Informational
Affects	Cutting
	Stacking

Line 92 cutting:

```
uint256 sum = 0;
```

Line 54 Stacking.sol:

```
uint256 sum = 0;
```

Retest 01/07/2022

Status	Closed
--------	--------

Fixed

Assignment to 0 removed.

Recomendation

Don't initialise both `sums`.

```
uint256 sum;
```

Unnecessary safemath

Risk	Informational
Affects	MarketPlace

Line 80:

```
_userSellingTokens[msg.sender].add(_tokenId);
```

Recomendation

Safe math is implicit with solidity > 0.8.0.

```
_userSellingTokens[msg.sender] + _tokenId;
```

would be sufficient.

Retest 01/07/2022

Status	Closed
--------	--------

This is not related to safemath. This is a function used ad library.
EnumerableSet.UintSet

Issue was a false positive.

Index events appropriately

Risk	Informational
Affects	Various

For example Cutting

```
event Cut(uint256 tokenId);  
event CutWithFee(uint256 tokenId, uint256 feeTokenId);
```

The events could be indexed on tokenId and possibly feeTokenId to allow more efficient searching for events.

Retest 01/07/2022

Status	Closed
--------	--------

Added event index for possible events

Hardcode values where possible

Risk	Informational
Affects	NFT721

Inside the `burn()` function, an if statement decrements `maxWeight` if `tokenDetails[tokenId].weight == jointWeight`:

```
if(tokenDetails[tokenId].weight == jointWeight)  
{  
    maxWeight[tokenDetails[tokenId].strain] -= tokenDetails[tokenId].weight;  
}
```

Since `jointWeight = 1` is a constant, there is no need to read `tokenDetails[tokenId].weight` from storage.

Recommendation:

Replace:

```
maxWeight[tokenDetails[tokenId].strain] -= tokenDetails[tokenId].weight;
```

With:

```
maxWeight[tokenDetails[tokenId].strain] -= jointWeight;
```

Retest 01/07/2022

Status	Closed
--------	--------

Fixed

Fixed as recommended:

```
if(_tW == jointWeight) {  
    maxWeight[tokenDetails[tokenId].strain] -= jointWeight;  
}
```

Avoid public functions for gas optimisation

Risk	Informational
Affects	Various

`public` functions that are never called internally should be declared `external` to save gas

Recommendation

Use the `external` visibility modifier for functions whenever possible.

References

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

Retest 01/07/2022

Status	Closed
--------	--------

Fixed where possible

virtual function modifier can be removed

Risk	Informational
Affects	NFT721

The virtual keyword is only needed if the contract function needs to be overridden, which is not the case of `NFT721`.

Recommendation

Delete all instances of the `virtual` keyword.

Retest 01/07/2022

Status	Closed
--------	--------

Removed all virtual modifiers

approvedQty should all be powers of 2

Risk	Informational
Affects	NFT721

Approved quantities are set at construction time as follows:

```
approvedQty[564] = true;  
approvedQty[256] = true;  
approvedQty[128] = true;  
approvedQty[64] = true;  
approvedQty[16] = true;  
approvedQty[8] = true;  
approvedQty[4] = true;  
approvedQty[2] = true;  
approvedQty[1] = true;
```

The quantity approved is checked at the time of minting as follows (simplified code):

```
require(approvedQty[_weight]);
```

I.e. the quantity minted must be one out of the values in `approvedQty`. Also the total amount cannot exceed `totalWeight[_strain] + _weight <= maxWeight[_strain]`

Therefore if `maxWeight` was a power of 2, it would be much easier to divide evenly by the `approvedQty`, yet `approvedQty[564]` is not a power of 2.

Recommendation

```
approvedQty[564] = true;
```

should be:

```
approvedQty[512] = true;
```

Retest 01/07/2022

Status	Closed
--------	--------

Logically this should be 564. We can't follow the powers of 2.

Issue is a false positive, no further action needed.

No setter functions for `maxWeight` and `approvedQty`

Risk	Informational
Affects	NFT721

The `maxWeight` and `approvedQty` mappings cannot be changed once set.

Recommendation

If required, add a setter function for admin only in order to update the `maxWeight` and `approvedQty` properties.

Retest 01/07/2022

Status	Closed
--------	--------

No setter needed for `maxWeight`. Added one setter for `approvedQty` (`setQuantity`).

Added new setter function for `approvedQty` (`setQuantity`).

Excessive error message length

Risk	Informational
------	---------------

Risk	Informational
Affects	Cutting

Line 71:

```
require(nft.isApproved(address(this), feeTokenId), "Token is not approved to transfer");
```

Messages from `require()` are best contained within 32 bytes as otherwise it will be distributed among two slots.

Recommendation

To stay below the 32 byte limit reduce message to:

```
require(nft.isApproved(address(this), feeTokenId), "Token not approved to transfer");
```

Retest 01/07/2022

Status	Closed
--------	--------

Fixed

```
require(nft.isApproved(address(this), tokenId), "not approved to cut");
require(nft.isApproved(address(this), feeTokenId), "not approved to transfer");
```

Dead Code

Risk	Informational
Affects	NFT721
	LibPart.sol
	AbstractRoyalties

NFT721 Internal function `_approve()` is never used.

NFT721 variable `_tokenURIs` is not used.

Function `hash()` and `TYPE_HASH` constant in LibPart is never used.

Functions `_onRoyaltiesSet()` and `_updateAccount()` in AbstractRoyalties are never used and should be removed.

Recommendations

Remove the unused functions and constant.

Retest 01/07/2022

Status	Closed
--------	--------

- NFT721 Internal function `_approve()` is never used.
 - The public function was overridden instead of using a separate internal function
- NFT721 variable `_tokenURIs` is not used.
 - Removed
- Function `hash()` and `TYPE_HASH` constant in LibPart is never used.
 - Unchanged as it is an external library
- Functions `_onRoyaltiesSet()` and `_updateAccount()` in AbstractRoyalties are never used and should be removed.
 - Unchanged as it is an external library

Floating Pragma

Risk	Informational
Affects	various

All libraries contracts use a floating `^0.8.0` pragma:

- EnumerableMap.sol
- RoyaltiesV2Impl.sol
- AbstractRoyalties.sol
- RoyaltiesV2.sol
- LibRoyaltiesV2.sol
- LibPart.sol

Recommendation

Since the contracts are in the local `contracts` folder and not in the `node_modules` they should be modified to match the fixed pragma of other contracts, 0.8.4 at the time of testing.

Retest 01/07/2022

Status	Closed
--------	--------

No further action needed, developers provided the following explanation:

Not changing anything inside the library.

Use a later Solidity Version

Risk	Informational
Affects	Various

Recommendation

Upgrade to Solidity compiler version > 0.8.8 for improved efficiency.

Retest 01/07/2022

Status	Closed
--------	--------

No further action needed, developers provided the following explanation:

Please keep the version the code is being created. 0.8.4 is used.

Unused-state

Risk	Informational
Affects	NFT721

`_tokenURIs` is never used in NFT721.

Recommendation

Remove `_tokenURIs` if not used, or create a getter function that returns the tokenURIs.

Retest 01/07/2022

Status	Closed
--------	--------

No further action needed, developers provided the following explanation:

This is already fixed with Dead Code (25th).

Remove >= and <=

Risk	Informational
Affects	Marketplace

The solidity compiler does not have an opcode for `<=`, `>=` or `!=`, therefore it is cheaper to get rid of them if possible from the line shown below:

```
require(_royaltiesPercentage >= 100 && _royaltiesPercentage <= 10000, "not valid");
```

Recommendation

```
require(_royaltiesPercentage >= 100 && _royaltiesPercentage <= 10000, "not valid");
```

Becomes:

```
require(_royaltiesPercentage > 99 && _royaltiesPercentage < 10001, "not valid");
```

Retest 01/07/2022

Status	Closed
--------	--------

Just removing the minimum check here. Only keep the greater check like `require(_royaltiesPercentage < 10001, "not valid")`.

Changed as recommended:

```
require(_royaltiesPercentage < 10001, "not valid")
```

Appendix

Static Analysis ERC Check

ERC721	Metadata extension (optional)	Enumerable extension (optional)
 Does contract support ERC165 Gas used: 133047	 Does contract support ERC721Metadata Gas used: 133027	 Does contract support ERC721Enumerable Gas used: undefined Check interface ERC721Enumerable.
 Does contract support ERC721. Gas used: 132972	 Contract name Gas used: 136048	 Total supply of the contract Gas used: undefined totalSupply should be greater than 0.
 Balance of zero address Gas used: undefined	 Contract symbol Gas used: 136050	 Get zero indexed token. Gas used: undefined tokenByIndex(0) should not throw.

Check functions

- [✓] balanceOf(address) is present
- [✓] balanceOf(address) → () (correct return value)
- [✓] balanceOf(address) is view
- [] transfer(address,uint256) is missing
- [✓] transferFrom(address,address,uint256) is present
- [] transferFrom(address,address,uint256) → () should return bool
- [✓] Transfer(address,address,uint256) is emitted
- [✓] approve(address,uint256) is present
- [] approve(address,uint256) → () should return bool
- [✓] Approval(address,address,uint256) is emitted
- [] allowance(address,address) is missing
- [✓] name() is present
- [✓] name() → () (correct return value)
- [✓] name() is view
- [✓] symbol() is present
- [✓] symbol() → () (correct return value)
- [✓] symbol() is view
- [] decimals() is missing (optional)

Check events

- [✓] Transfer(address,address,uint256) is present
- [✓] parameter 0 is indexed
- [✓] parameter 1 is indexed
- [✓] Approval(address,address,uint256) is present
- [✓] parameter 0 is indexed
- [✓] parameter 1 is indexed

