SOLIDIFIED

Audit Report for Frameit - December 16, 2022

DRAFT - DO NOT PUBLISH

## Summary

Audit Report prepared by Solidified covering the Frameit creator platform.

## Process and Delivery

Three (3) independent Solidified experts performed an unbiased and isolated audit of the code below. The final debrief took place on December 16, 2022, and the results are presented here.

## Audited Files

The source code has been supplied in the following source code repository:

Repo: https://gitlab.com/rungie/frameit-contracts/
Commit hash: `10c78a09621338b53e0a606f6a313988ea95ed40`

The following contracts were in scope:

```
.
├── DEX
│   └── contracts
│       └── UniswapV3Oracle.sol
└── smartcontracts
    └── contracts
        ├── album
        │   └── FrameItAlbum.sol
        ├── factory
        │   └── FrameItFactory.sol
        ├── marketplace
        │   ├── FrameItMarketplace.sol
        │   └── FrameItPacksMarketplace.sol
        ├── moonpay
        │   └── MoonPayWallet.sol
        ├── nfts
        │   ├── FrameItLootBox.sol
        │   ├── FrameItNFT.sol
        │   └── FrameItSoulbound.sol
        ├── royalties
```

```
|   └── FrameItSalesSplitter.sol
└── token
    └── RungieToken.sol
```

## Intended Behavior

The audited contracts implement an NFT marketplace for primary (lootboxes) and secondary sales.

## Findings

Smart contract audits are an important step to improve the security of smart contracts and can find many issues. However, auditing complex codebases has its limits and a remaining risk is present (see disclaimer).

Users of a smart contract system should exercise caution. In order to help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentation**, and **test coverage**.

Note, that high complexity or lower test coverage does not necessarily equate to a higher risk, although certain bugs are more easily detected in unit testing than a security audit and vice versa.

| Criteria | Status | Comment |
|---|---|---|
| Code complexity | Low | - |
| Code readability and clarity | Medium | - |
| Level of Documentation | Low-Medium | The functions / parameters are not documented and there are only a few comments in the codebase |
| Test Coverage | Medium | - |

SOLIDIFIED

Audit Report for Frameit - December 16, 2022

DRAFT - DO NOT PUBLISH

## Issues Found

Solidified found that the Frameit contracts contain 2 critical issues, 1 major issue, 13 minor issues, and 4 informational notes.

We recommend issues are amended, while informational notes are up to the team's discretion, as they refer to best practices.

| Issue # | Description | Severity | Status |
|---|---|---|---|
| 1 | UniswapV3Oracle.sol: TWAP of pool with very low liquidity may be used | Critical | Pending |
| 2 | Lack of access control for destructive operations with albums | Critical | Pending |
| 3 | FrameItMarketplace.sol: Frontrunner can cancel offers | Major | Pending |
| 4 | Centralization risks | Minor | Pending |
| 5 | FrameItNFT and FrameItSoulbound not compliant with EIP 721 | Minor | Pending |
| 6 | FrameItSoulbound returning the same URL for all tokens | Minor | Pending |
| 7 | FrameItSoulbound: Anyone can mint | Minor | Pending |
| 8 | FrameItSalesSplitter: transfer used for sending ETH | Minor | Pending |
| 9 | FrameItSalesSplitter._withdrawETH9Payments: Failure to accept payment by one of the royalty wallets blocks other wallets | Minor | Pending |
| 10 | FrameItAlbum.populateAlbum: Completing album that was added with _startIndex > 0 not possible | Minor | Pending |
| 11 | FrameItAlbum.getUserNFTsForAlbumId can | Minor | Pending |

| | | | |
|---|---|---|---|
| | wrongly include ID 0 | | |
| 12 | FrameItPacksMarketplace.buyPack: No discount applied when sale token is rungie token | Minor | Pending |
| 13 | FrameItMarketplace.buySale: Malicious seller can cause DoS | Minor | Pending |
| 14 | FrameItMarketplace.OfferData: Data structure is inconsistent with the interface | Minor | Pending |
| 15 | FrameItMarketplace.cancelOffer: Method has redundant payable modifier | Minor | Pending |
| 16 | FrameItSalesSplitter: If the royaltyFeesInBips sums to greater than 10000 the transaction will revert | Minor | Pending |
| 17 | FrameItPacksMarketplace: Same NFT can be listed multiple times | Note | Pending |
| 18 | Gas optimizations | Note | Pending |
| 19 | FrameItMarketplace: Marking signatures as used is not recommended | Note | Pending |
| 20 | String concatenation | Note | Pending |

# Critical Issues

## 1. UniswapV3Oracle.sol: TWAP of pool with very low liquidity may be used

The UniswapV3Oracle always checks all available pools (with the different fee levels) for a given pair and uses all available pools to calculate the result. However, for certain pairs (e.g., pools with a 1% fee for stablecoins) some pools may exist, but have very little liquidity. This can make price manipulations very cheap, although a TWAP is used. Moreover, an attacker could also deploy a pool for a certain fee level (to manipulate the price) when it does not exist already.

**Recommendation**
Consider discarding pools when the liquidity differs a lot (e.g. by over 50%) from the other pools for the pair.

## 2. Lack of access control for destructive operations with albums

Anyone can use `destroyAlbum` to delete any album stored in the `fullAlbum` mapping. This is in contrast to `populateAlbum`, which can only be called by an owner.

**Recommendation**
Consider restricting access to the `owner` of the album or the NFT.

# Major Issues

## 3. FrameItMarketplace.sol: Frontrunner can cancel offers

Anyone that knows the signature of an offer can use `cancelOffer` to cancel it. This can be abused by a frontrunner that sees a call to `acceptOffer` (with the signature) in the mempool and uses `cancelOffer` to cancel it before the `acceptOffer` call.

**Recommendation**
Only allow canceling offers by the address that made the offer (i.e., `offer.from`).

# Minor Issues

## 4. Centralization risks

In `MoonPayWallet` and `FrameItSalesSplitter`, the `owner` can withdraw the whole balance (Moonpay balance for `MoonPayWallet`, native token balance for `FrameItSalesSplitter`) using the functions `companyWithdraw` and `withdrawNative`. These tokens are reserved for other receivers, so a misuse of those functions (e.g., after stolen private keys) will lead to a loss of funds for those users. Also, the `owner` of `FrameItSalesSplitter` contract can replace royalty wallets with pending payments by other wallets. The pending payments would become withdrawable only to new royalty wallets. Royalty fee structure can be altered for already pending payments as well.

**Recommendation**

Consider removing functions `companyWithdraw` and `withdrawNative` as well as making function `setWallets` either revert in case of positive balance of `FrameItSalesSplitter` or be callable only once.

## 5. FrameItNFT and FrameItSoulbound not compliant with EIP 721

EIP 721 states that the function `tokenUri` "throws if `_tokenId` is not a valid NFT". This is not the case for `FrameItNFT` and `FrameItSoulbound`. Even when the token does not exist, a token URI is returned. This can lead to problems with other smart contracts which expect that the contract adheres to EIP 721 (and therefore reverts for invalid IDs).

**Recommendation**
Revert for invalid / non-existing IDs.

## 6. FrameItSoulbound returning the same URL for all tokens

The function tokenURI of FrameItSoulbound returns the base URI for all token IDs and therefore does not allow customizing the image of an NFT.

**Recommendation**
Return a URI that includes the ID of the token.

## 7. FrameItSoulbound: Anyone can mint

The `mint` function of `FrameItSoulbound` is not protected and anyone can call it to mint new tokens. While these tokens will be minted for the creator (limiting the impact), this may still be undesirable because a malicious user could mint a lot of unneeded tokens to the creator.

**Recommendation**
Consider restricting the minting to the creator.

## 8. FrameItSalesSplitter: transfer used for sending ETH

The `withdrawNative` function of `FrameItSalesSplitter` uses `transfer` to send ETH to the owner.

As `transfer` only comes with a 2300 gas stipend, its use is discouraged. If the owner is for instance a multi-sig wallet with some complicated business logic in its `receive` function, the transaction will revert.

**Recommendation**
Use `call` for sending ETH.

## 9. FrameItSalesSplitter._withdrawETH9Payments: Failure to accept payment by one of the royalty wallets blocks other wallets

The function `_withdrawEth9Payments` reverts if any payment to a royalty wallet was unsuccessful. Although the `owner` of `FrameItSalesSplitter` contract controls which wallets are added, if one of the wallets is a smart contract it can fail to receive the payment. In this case, all of the other wallets will not be able to receive their funds.

**Recommendation**
Consider adopting a pull payments pattern.

## 10. FrameItAlbum.populateAlbum: Completing album that was added with _startIndex > 0 not possible

When an album is added with a `_startIndex > 0`, `totalNFTs` is still set to the length of the whole `_ids` array, but only parts of it are added. This means that `checkAlbumComplete` can

never return true for such an album, because it requires that a user has `totalNFTs` NFTs, but he can have `totalNFTs - _startIndex` at most.

**Recommendation**
Consider either setting `totalNFTs` to `_ids.length - _startIndex` or removing the `_startIndex` option.

## 11. `FrameItAlbum.getUserNFTsForAlbumId` can wrongly include ID 0

---

Because the array `ids` is allocated with length `nfts.length`, but a user can have less NFTs of an album, the last items may not be set and will therefore be 0. This can confuse consumers of this function, as ID 0 can also be a valid token ID.

**Recommendation**
Resize the array to length `counter` before returning it.

## 12. `FrameItPacksMarketplace.buyPack`: No discount applied when sale token is rungie token

---

The discount for buying with the rungie token is subtracted within the following branch:
`if (sale._token != _paymentToken) {`
Therefore, when `defaultSaleToken` is set to the rungie token (and sales therefore have to use this token), the discount will not be applied.

**Recommendation**
Also apply the discount when the token of the sale is the rungie token.

## 13. FrameItMarketplace.buySale: Malicious seller can cause DoS

Although access to `setSale` is restricted to the owner of the `FrameItMarketplace` contract, a malicious or malfunctioning contract could be registered as a seller of a NFT. Successive calls to `buySale` could lead to reversion of the whole batch if just one of the sellers fail to receive the payment.

**Recommendation**
Consider adopting the pull payments pattern and/or adding a non-batched version of `buySale` function.

## 14. FrameItMarketplace.OfferData: Data structure is inconsistent with the interface

The data structure `OfferData` is implemented in both the contract `FrameItMarketplace` and its interface `IFrameItMarketplace`. The version in `IFrameItMarketplace` lacks the field `offerDate` which is used for signature verification in the function `_verify`. Since third parties that integrate with the contracts likely use the interface to do so, their transactions that use `verify` can construct invalid signatures.

**Recommendation**
Consider extracting common code into a dedicated module.

## 15. FrameItMarketplace.cancelOffer: Method has redundant payable modifier

The function `cancelOffer` in the interface `IFrameItMarketplace` is only marked as `external`, but not as payable. However, in the contract `FrameItMarketplace` implementing this interface,

`cancelOffer` function is also marked as `payable`. The code of it neither works with payments nor tracks user deposits, so accidentally attaching a payment to a call of this method would result in loss of funds.

**Recommendation**
Remove the `payable` modifier. Also, in order to avoid inaccuracies of this kind, implement interfaces of any contract in the codebase explicitly using the keyword `is`.

## 16. FrameItSalesSplitter: If the royaltyFeesInBips sums to greater than 10000 the transaction will revert

---

In _withdrawTokenPayments and _withdrawETH9Payments of FrameItSalesSplitter, the royaltyWallets are iterated from front to back. If royaltyFeesInBips sums to greater than 10000, when transfers are made to the last wallets, there will not be enough balance to transfer the remaining amounts, causing a revert of all transfers.

**Recommendation**
Ensure that royaltyFeesInBips sums to less than or, ideally, equal to 10000.

## Informational Notes

---

## 17. FrameItPacksMarketplace: Same NFT can be listed multiple times

---

The owner of an NFT can use `setPackForSale` to list it for sale multiple times. After there is a bid on one auction, the other(s) will no longer work, because `_sendPacks` transfers the individual tokens in a consecutive fashing (starting with token ID 1).

**Recommendation**
Consider validating that an NFT is only listed once to avoid errors.

## 18. Gas optimizations

There are multiple gas optimizations that would result in significant savings:

- The usage of `_safeMint(address(this), id)` in `FrameItLootBox.mintLootBoxes` results in an additional, unnecessary callback to the contract for each mint. Consider using `_mint` instead.
- In `FrameItPacksMarketplace`, instead of having an array `allowedTokens` and iterating over the whole array to find out if a token is allowed, a mapping could be used.
- In FrameItAlbum, the functions `checkAlbumComplete` and `getUserAlbumIds` could use `break` instead of `continue` within the inner loop. This would be semantically equivalent (because it does not matter if `found` is set to true multiple times for the same NFT), but could save a lot of iterations.

## 19. `FrameItMarketplace`: Marking signatures as used is not recommended

Because of signature malleability (i.e., the possibility to create another signature for the same payload with one valid signature), it is generally not recommended to mark signatures (or their hash) as used, like it is done in FrameItMarketplace. Instead, the payload itself (or rather the hash of the payload) should be marked as used or nonces should be used. Before OpenZeppelin 4.7.3, the contract would have been vulnerable (as the ECDSA library had a malleability vulnerability when a single `bytes` argument was passed to them). While this is fixed and the used version is no longer vulnerable, it is still recommended to follow best practices.

**Recommendation**
Consider marking the hash of the offer payload as used or including a nonce.

## 20. String concatenation

With the release of Solidity 0.8.12 there is no need to use `abi.encodePacked` for the concatenation of strings. New method `strings.concat` makes the code easier to read and maintain.

**Recommendation**
Consider setting the compiler version to 0.8.12 or higher and adopting the `strings.concat` method.

## Disclaimer

Solidified audit is not a security warranty, investment advice, or an endorsement of DIGITAL COLLECTIVE TOKENS S.L. or its products. This audit does not provide a security or correctness guarantee of the audited smart contract. Securing smart contracts is a multistep process, therefore running a bug bounty program as a complement to this audit is strongly recommended.

The individual audit reports are anonymized and combined during a debrief process, in order to provide an unbiased delivery and protect the auditors of Solidified platform from legal and financial liability.

*Oak Security GmbH*