Trove Marketplace Contract Review #74

Open **fulldecent** opened this issue on Jun 11, 2022 · 2 comments

fulldecent commented on Jun 11, 2022 Contributor

Please find complete review below. I'm not quite sure if this branch is intended to merge into the `master` branch here or to a forked repository. I can turn these into bite-sized issues in this or the other repo as appropriate.

Trove Marketplace Contract Review
William Entriken
Delivered 2022-06-11

Thank you for the opportunity to participate in this code review!

Review scope

This is a review of Trove Marketplace, pre-release version, "vinny/collection-level-currency" branch. Code was published at https://github.com/TreasureProject/treasure-marketplace-contracts/tree/b92618e0f760ab7aa5d5450dceea0dd576f5c4b8 with SHA-1 hash `b92618e0f760ab7aa5d5450dceea0dd576f5c4b8`.

The scope of review included code-level review (human reading source code), automated code review, (computer reading source code) and unit testing (computer running small-scope tests). Due to limitations of the engagement, the review does not include functional testing (human using compiled program).

The review was performed for the purpose of comparing Trove Marketplace against the plain English meaning of comments and function names, the provided documentation in the README, documentation posted in the Trove Marketplace Whitepaper (version 01.2022) (SHA-256 hash: 2788fdd66f241ac672e6d04e73b416ecff234b174841c7cbe764fc55ba129b8b09ca09d2fa53d042b7fd1617bf38c9c2) and a general understanding of NFT marketplaces. Any deviation from this standard, or deficiency against relevant best programming practices would be an in-scope review finding.

Sensitivity was set as "anything we should fix or that you wouldn't want a million people to read".

OpenZeppelin Contacts Upgradable (version 4.5.2) were specifically excluded from the scope. Although certain notes about the contracts used are provided below.

Certain comments in the Trove Whitepaper discussing "money", "currency", "pricing", "capital", coordinated seller behavior ("temperature") and "[signaling]", forward looking statements regarding revenue expectations, and similar are excluded from this review. This review does not consider the applicability of this project to any specific jurisdiction's laws. Although certain of these concepts are discussed below where there is a risk of customer confusion or harm.

Quality and correctness of automated testing approaches and deployment strategy is excluded from the scope.

Findings

Each finding is listed and is organized by topic. Critical findings that are not resolved are marked with the bomb symbol (💣) and other unresolved material findings are marked with the warning symbol (⚠️). An informational note is marked with an information symbol (ℹ️). Resolved and other less-than-material findings have no emoji marking.

💣 Incorrect note about listing "blocking"

A comment in the code advertises a condition where listings could become invalidated.

> Ensure the buyer, the seller, and the collection all agree on the token to be used for the purchase. If the token used for buying/selling has changed since the listing was created, this effectively blocks all the old listings with the old payment tokens from being bought.

This statement is incorrect because the listing could become "revalidated" if the token is changed back, whereas the wording "invalidated" implies that the process is permanent.

A specification scenario could result in customer harm:

1. A seller creates a listing
2. The admin changes the token
3. The seller believes their listing is dead
4. The value of the token increases significantly
5. The admin changes the token back
6. Somebody buys at the old price which is below market value

Recommendation: clearly explain the circumstances where a listing could or could not be applicable.

💣 Reference is made to "backwards compatibility"

> **but we ultimately decided to open a second marketplace**

Recommendation: remove any backwards compatibility shims, or if this is indeed a proxy upgrade of an existing contract, clearly document this and tell me right away!

💣 Make sure that ownership of deployment wallet has been renounced

A note in the README states:

> Make sure that ownership of deployment wallet has been renounced

This is not implemented in the deployment script.

Furthermore, if a wallet is renounced, this may prevent further contract upgrades, obviating the need to use upgradeable contracts in the first place.

Recommendation: determine the original intention of this statement, and a) remove contract upgradeability, b) carefully implement renouncing, or c) remove the comment.

⚠️ Owner can rugpull with upgrade

This contract is using upgradable logic. This means that the owner can deploy a new contract at any time and take every NFT and all authorized payment tokens owned by everybody that has listed NFTs for sale on the marketplace.

This may be an economically feasible rugpull strategy. And the feasibility of this rugpull increases as time goes on.

This risk is disclosed in the contract source code! (Good!)

But it contradicts information in the Trove Marketplace Whitepaper. Specifically the whitepaper claims to put forth a "decentralized marketplace."

Such a risk may not be obvious to participants of the smart contract. And so for them this is an unknown risk.

Recommendation: use appropriate outreach/disclaimers so that individuals who read that whitepaper can be aware of this change.

⚠️ Project is not (only) denominated in Ether

This contract allows multiple payment tokens. And it appears that the intention is to support multiple tokens other than WETH.

This contradicts information in the Trove Marketplace Whitepaper. Specifically the whitepaper claims to be "denominated in ETH".

⚠️ Graveyard mechanism not implemented

The Whitepaper introduces:

> Rather than incorporating a hide mechanism for your unwanted/0 value NFTs there will be a graveyard mechanic where you can exchange 5 of those NFT's for a chance at summoning some sort of productive asset from the Treasure ecosystem.

This feature is explained to be a part of this project. However this is not implemented in the smart contract.

Recommendation: use appropriate outreach/disclaimers so that individuals who read that whitepaper can be aware of this change.

Stale reference to `collectionToTokenAddress`

A comment refers to `collectionToTokenAddress` where such a thing does not exist.

Recommendation: update reference to `collectionToPaymentToken` or as appropriate.

Interchangeable use of "token address" and "payment token"

The terms "token address" and "payment token" appear to represent the same thing and the words are used interchangeably.

Using different wordings to refer to the same thing can reduce clarity of text.

Recommendation: if these indeed refer to the same thing, pick one wording and use it consistently.

NatSpec documentation is missing

NatSpec documentation allows wallets and other Web3 applications to inform a human interacting with the contract what they are about to do. They may see "are you sure you want to call `cancelListing` with `_nftAddress` 0x232398473... and `tokenID` 11" rather than something like "do you want to call `0x5cd346fa` with message data `0x0273eecfb480dcfbab423bd15c71e7ad70c7a9504689ff56edb137a5d37146d6` ".

Best practice is to include documentation for all public interfaces and this is recommended by the Solidity project:

> It is recommended that Solidity contracts are fully annotated using **NatSpec** for all public interfaces (everything in the ABI).
>
> https://docs.soliditylang.org/en/v0.8.6/style-guide.html?highlight=natspec

Missing package-lock.json

The file package-lock.json (yarn.lock?) should be checked into the repository to allow reproducible builds.

The openzeppelin/contracts-upgradeable dependency is version pinned so this only affects test running.

Recommendation: remove package-lock.json and yarn.lock from .gitignore and commit this file to the repository.

Initialization function should handle setting WETH

WETH is a singleton contract on each Ethereum chain.

This smart contract allows to set the address of the WETH contract only once.

Set-something-once functionality belongs in the contract initializer if possible.

By separating the logic into its own function, this creates a risk that it is not performed properly or that it is accidently skipped during deployment.

And specifically for this contract a separate check is made in the code which will perform differently if WETH is set to the (default) zero address. Namely, `_paymentTokenForListing == address(weth)` .

Recommendation: remove the `setWeth` function and move this feature into `initialize` .

Cancelling bid does not delete all associated storage

When cancelling bids, not all storage is freed, only the quantity field is freed.

```
collectionBids[_cancelBidParam.nftAddress][_msgSender()].quantity = 0;


tokenBids[_cancelBidParam.nftAddress][_cancelBidParam.tokenId][_msgSender()].quantity = 0;
```

Recommendation: delete the whole struct

Contract is named treasure marketplace but this is trove marketplace

This appears to be the Trove marketplace, which is distinct from the Treasure marketplace.

Per whitepaper:

Recommendation: update naming to Trove.

Implement type safety for storage variables/function parameters

Several instances in the code are using casting for `account payable` or `IERC20Upgradeable`. These indicate places where too generic of a type is being used.

Using overly generic types creates a risk of human errors when maintaining the product in the future.

Recommendation: use more specific types, `address => IERC20Upgradeable`, `address => address payable` where appropriate.

Bid and bidstorage variable names are used interchangeably to refer to a storage location

The naming `bid` and `bidStorage` are used interchangeably to refer to a storage location for a bid.

Using different wordings to refer to the same thing can reduce clarity of text.

Recommendation: if these indeed refer to the same thing, pick one wording and use it consistently, also apply this decision more broadly as applicable.

Solidity version is unsupported

Solidity recommends that only the latest released version (currently 0.8.14) should be used:

> When deploying contracts, you should use the latest released version of Solidity.
>
> https://docs.soliditylang.org/en/v0.8.14/#solidity

This smart contract currently allows to compile with another version.

Recommendation: explicitly require the currently supported compiler (i.e. `pragma solidity 0.8.14;`).

ℹ️ Almost attack with reentrancy-before-read

The product is NOT vulnerable to this attack. But I wanted to document this anyway as it could be actionable if one of the smart contract features was changed in the future.

- The buy and sell functions are not symmetric, one uses memory variables and the other uses storage.
- Loading the fee amount is done after the transfer is validated.
- The fee amount (for collection) can be changed using a separate function call.

Recommendation: if the smart contract is changed in the future, confirm that the current mitigation approach is still in place to prevent this attack.

lessthanjake commented on Jun 12, 2022

@fulldecent The whitepaper mentioned here (Trove Marketplace Whitepaper (version 01.2022)) is out of date as key fundamentals with regards to the Trove marketplace have changed. More recent docs related to Trove can be found here: https://docs.treasuretrove.lol/about-trove/introduction

- Trove site will not be decentralized (same as the current Treasure marketplace)
- Graveyard or Burn/exchange mechanism will be implemented at a future date: https://docs.treasuretrove.lol/about-trove/trove-mechanics/graveyardhttps://docs.treasuretrove.lol/about-trove/trove-mechanics/graveyard
- In addition to $magic and eth denominated collections, other stablecoin denominated collections may be made available: https://docs.treasuretrove.lol/about-trove/usdmagic-and-eth-denominations
- Trove will replace Treasure Marketplace (hence the upgrade contract). But at launch both marketplace sites will exist and run in parallel. Once sufficient parity and reliability is established on the Trove site, the Treasure marketplace site will be shut down.

fulldecent commented on Jun 13, 2022 Contributor Author

Thank you for the documentation.

1. Not decentralized, got it.
2. Graveyard appears that it will be implemented separately from the smart contract here we are discussing. So we can ignore it for here.
3. Got it, thank you.
4. The word "upgradeable" has a special meaning in smart contracts. That is referring to a deployed smart contract at a specific address that changes its behavior while keeping all connected data. What you are referring to here appears to be not that, you are referring to a NEW smart contract deployed to a NEW address and then for business reasons, eventually removing mentions of the old contract. If I misunderstood this, let's please review in detail as this is an important distinction.

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

2 participants