Report for Pwn Finance

September 2022

Version 1.3

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 19th and 23rd August 2022.
Overall the code quality is high, after discussion with the developers, a posssible issue was found to not be applicable. The remaining issues are minor optimisations.
After retests and discussion with the developement team, the remaining issues have been resolved.

1.1 Technical Summary

The informational issues presented are small optimisations that could reduce gas needed during deployment and runtime.

Scope

The following contracts were regarded as in scope for this audit.

`TokenBundler.sol`
`ITokenBundler.sol`

The retest code was taken at commit `d4ca081d4e585149c6f21556fd07f5fa683ffa68`

Issues count summary

Stage	Critical	High	Medium	Low	Info	Total
Initial 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

Disclaimer

Client Confidentiality

Proprietary Information

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.

String variable casting not needed for abi encoded params

Risk	Informational
Affects	`TokenBundler`

`string(abi.encodePacked(_uriBase, block.chainid.toString(), "/", address(this).toHexString(), "/{id}/metadata"))`

and

`abi.encodePacked(_uriBase, block.chainid.toString(), "/", address(this).toHexString(), "/{id}/metadata")`

produce the same variable of type string.

`block.chainid.toString()` should also not be needed as `abi.encodePacked()` will cast uint to string automatically, the same is true for `address(this).toHexString()` as the address is already in hex format.

Retest 26/08/22

Status	Fixed

Fixed in commit 423013dac7102b19515c5af42dbc973fbf528948:

```
constructor(string memory _uri) ERC1155(_uri)
```

For reference, the setter function:

```
function setUri(string memory _newUri) external onlyOwner {
    _setURI(_newUri);
}
```

Duplicated code

Risk	Informational
Affects	`TokenBundler`

In Multitoken.sol `transferAsset` and `transferAssetFrom` have mostly the same function body, the repeated code could be moved to a private function in order to reduce the bytesize of the contract:

```
} else if (_asset.category == Category.ERC721) {
    IERC721 token = IERC721(_asset.assetAddress);
    token.transferFrom(_source, _dest, _asset.id);

} else if (_asset.category == Category.ERC1155) {
    IERC1155 token = IERC1155(_asset.assetAddress);
    if (_asset.amount == 0) {
        _asset.amount = 1;
    }
    token.safeTransferFrom(_source, _dest, _asset.id, _asset.amount, "");

} else {
    revert("MultiToken: Unsupported category");
}
```

To put this differently, the only part of the function body is different between the two functions is:

```
if (_asset.category == Category.ERC20) {
        IERC20 token = IERC20(_asset.assetAddress);
        token.transfer(_dest, _asset.amount);
```

One could simply keep the `transferAsset` for transfering erc20 tokens only using `.transfer` method (i.e. remove the if/else blocks) since erc721 and erc1155 don't have a `transfer` method at all, and call `transferAssetFrom` for any other transfer, including erc20 but with `.transferFrom` method.

Retest 06/09/22

Status	Fixed

The hook functions now revert when the operator is not the bundler address.

Centralized storage

Risk	Informational
Affects	`TokenBundler`

```
ERC1155(string(abi.encodePacked(_uriBase, block.chainid.toString(), "/",
address(this).toHexString(), "/{id}/metadata"))
```

Clearly cannot be ipfs since the URI is formed by this contract's address rather than an IPFS CID or similar.

Retest 26/08/22

Status	Fixed

The commit 423013dac7102b19515c5af42dbc973fbf528948 fixes the issue in the sense that there is no way to determine what the base URI will be, we simply recommend to use a decentralized storage solution such as IPFS so that token metadata cannot be tampered with.

Unnecessary "on-*-Received" tranfer hooks

Risk	Informational
Affects	`TokenBundler`

TokenBundler does not have any admin function in order to withdraw or send any erc721 it owns, therefore **`onERC721Received`** may have undesired effects, like signaling other contracts that it can handle receiving erc721 when in reality it cannot handle them unless they are sent to the contract via the `transferFrom` method by calling the `create` function.

`safeTransferFrom` is not being used by `MultiToken` and therefore there is no need to implement `onERC721Received`, unlike erc1155 which always checks the response of `try IERC1155Receiver(to).onERC1155Received()`, `try IERC721Receiver(to).onERC721Received(_msgSender()` is never called and therefore is not needed by TokenBundler.

The same applies to **`onERC1155BatchReceived`** which is not needed as the contract does not have any functions for handling erc1155 batch transfers.

Remember to remove `interfaceId == type(IERC721Receiver).interfaceId ||` from the `supportsInterface` function as well.

Retest 26/08/22

Status	Fixed

Fixed in commits 423013dac7102b19515c5af42dbc973fbf528948

As per the following comment:

> I didn't remove the onERC1155BatchReceived as EIP1155 stated that to implement IERC1155Receiver contract has to implement both functions. I realise that it’s not used and transferring assets directly to the bundler will lead to loss of funds, so I would like to ask for your opinion on this topic. Is it common to implement only those hooks, that are used? I see the benefit in protecting user who accidentally transfer funds in batch.

The EIP-1155 mentions the following under the section "onERC1155BatchReceived rules":

> The recipient contract MAY reject an increase of its balance by calling revert.
> If the recipient contract throws/reverts the transaction MUST be reverted.

In other words, we recommend that you simply add a `revert()` statement inside of the `onERC1155BatchReceived` function.

Optimisation - duplicated access of length property

Risk	Informational
Affects	`TokenBundler`

The property length of the `_assets` array is accessed twice.

Recommendation

Move line 70 `uint256 length = _assets.length;` to above the require statments, and replace the first require paramenter.

Retest 26/08/22

Status	Fixed

Fixed in commits 423013dac7102b19515c5af42dbc973fbf528948 and fd203ecaaa532bc2a12d9e654672e83613a16534