EMERALD SHIELD

Smart Contract Audit Report

Prepared for

YDY

v1 - October 18th 2022

Disclaimer

The audit reports delivered by Emerald City Labs Inc represents an extensive auditing process intended to help our clients increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology. The Reports do not provide any warranty or representation to any Third-Party in any respect, including regarding the bug-free nature of code, the business model or proprietors of any such business model and the legal compliance of such business model. The Reports should not be used as an endorsement or indictment of the project or team, and does not guarantee the security of the project.

No Third-Party should rely on the Reports in any way, including for the purpose of making any decisions to buy or sell any token, product or service. The Reports do not consider, and should not be interpreted as considering or having any bearing on the potential economics of a token, token sale, or any other product, service or asset.

Specifically, for the avoidance of doubt, the Reports delivered by Emerald City Labs Inc., should not be represented to any Third-Party as investment advice, nor as an endorsement of the project, team or to the absolute security of the project.

About Us

Emerald City Labs Inc is a private company with its address at 905, 447 Broadway, 2nd Floor, New York, NY, US, 10013

www.ecdao.org

v1 - October 18th 2022

About YDY

YDY is a Sweat to Earn platform that rewards fitness activity. Get started sweating with our 100% free instructor-led fitness classes.

Audit Outline

Scope of Audit	5 Contracts & Associated Transactions
Performed by	Jacob Tucker
Git Repo	https://github.com/StudioEngineering/YDY-testnet-contract
Git Branch	main
Commit Hash	f7d700dc36da929a8598175ab44633a3be378aa6

Audit Process

During October 2022, Metaverse Football League (MFL) engaged Emerald City Labs Inc. to conduct an audit of 6 smart contracts that relate to their recently launched dApp. The engagement was technical in nature and the audit was focussed on ensuring the security & efficiency of their Cadence codebase. After an introductory call, MFL provided Emerald City Labs Inc to access to their Github repositories and whitepaper.

This document will be updated to reflect any changes implemented by the team at MFL and when the audit is deemed complete & satisfactory by Emerald Labs Inc. the final version will be published publicly.

Audit Structure

The findings in this document have been structured into the following sections:

- Critical Bugs
- Minor Bugs
- Informational
- Recommendations
- Other Comments & Notes

Critical Bugs

A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.

1. No critical bugs were found.

Minor Bugs

A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.

2. One minor bug was found.

YDYHeartNFT Contract

Item #	CODE REFERENCE	RECOMMENDATION
2.1	Line 305 & 345	I recommend never using &AnyResource if you don’t have to. Ensure this is the correct type by using &Collection instead.

Informational

An observation or noted vulnerability that is informational in character but is not effecting any of the code.

3. The following informational areas were discovered

YDYHeartNFT Contract

Item #	CODE REFERENCE	RECOMMENDATION
3.1	Line 102	Note that your NFT ids will begin at 1. It is usually best practice to increment from 0, but if you intend to start at 1, this is totally fine.
3.2	Line 139	I'm not sure if this is what you're looking for, but, you might want to get rid of the pre-condition and just cap the stamina at 100 if stamina + points > 100. This way, the transaction won't abort, and you'll leave the user with 100 stamina points.
3.3	Line 145	On the contrary to Item #3.2, it might make sense to add a pre-condition here to make sure stamina doesn't go below 0, but this is up to you.
3.4	Line 169	You should probably support ExternalURL view as well to be go into the NFT Catalog.
3.5	Line 277	Your return type is an optional type, but the code inside the function will never return an optional. It is up to you to decide what you want to do, but I'd recommend following through on the optional return type and changing the force unwrap inside the function.

YDYMarketplace Contract

Item #	CODE REFERENCE	RECOMMENDATION
3.6	Line 11	You don’t seem to use SaleItem anywhere. Is this intentional?
3.7	Line 88	While this isn’t necessarily a bug, I’d recommend passing in a: Capability<&YDYHeartNFT.Collection{YDYHeartNFT.YDYHeartNFTCollectionPrivate, YDYHeartNFT.YDYHeartNFTCollectionPublic}> type instead of a: Capability<&YDYHeartNFT.Collection> type. It won’t effect your code at all, but makes it more clear to a caller what type they need to pass in.

Recommended Changes

Recommendations to improve efficiency, effectiveness, clarify, maintainability, security, and control based on established best practices.

4. The following recommendations were made

YDYToken Contract

Item #	CODE REFERENCE	RECOMMENDATION
4.1	Line 60	It’s inefficient to have a burnTokens function just to emit an event. You should emit `TokensBurned` inside of the destroy function of the Vault resource, and then delete the `burnTokens` function.
4.2	Line 90	I believe the Burner resource is not needed. Its only function is to burnTokens, which only has the purpose of emitting a TokensBurned event. Follow Item 4.1.

FTWToken Contract

Item #	CODE REFERENCE	RECOMMENDATION
4.3	Line 56	Same as 4.1
4.4	Line 87	Same as 4.2

YDYHeartNFT Contract

Item #	CODE REFERENCE	RECOMMENDATION
4.5	Line 262	It’s more efficient to use the force-move operator `<-!` instead of double moving. Meaning:

		`self.ownedNFTs[id] <-! token` This way you do not have to destroy an old resource.
4.6	Line 292	It seems inefficient to me to store a buy function inside of every user’s collection if only the Admin should be able to mint it. I would recommend putting the buy function inside some Admin type of resource, like NFTMinter in your case.
4.7	Line 336	The getTotalSupply function is not needed, and although minimal, will waste contract storage.
4.8	Line 415	Because `price` is a variable with a set price, you may want to add a `changePrice` function inside an Admin-like resource to change this price in the future.
4.9	Line 130	Instead of setting royalties as a variable in the init function, it might make more sense to simply return a royalty array inside the MetadataViews.Royalty on line 215. This is because right now, if you ever wanted to update the royalty, you wouldn’t be able to on pre-existing NFTs since they are already set as variables. But in a function call, you can change this whenever you want.

END

v1 - October 18th 2022