STAYSAFU
AUDIT

MAY 26TH, 2022

DEFI DINOS

TABLE OF CONTENTS

AUDIT SUMMARY

This report was written for DeFi Dinos in order to find flaws and vulnerabilities in the DeFi Dinos project's source code, as well as any contract dependencies that weren't part of an officially recognized library.

A comprehensive examination has been performed, utilizing Static Analysis, Manual Review, and Defi Dinos Deployment techniques. The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors
- Assessing the codebase to ensure compliance with current best practices and industry standards
- Ensuring contract logic meets the specifications and intentions of the client
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders
- Through line-by-line manual review of the entire codebase by industry expert

AUDIT OVERVIEW

PROJECT SUMMARY

Project name	DEFI DINOS
Description	The DeFi Dinos project is built around a 4,999 supply NFT about dinosaurs. The audit concerns Fightclub, the new PVP and P2E game of the project.
Platform	C-CHAIN
Language	Solidity
Codebase	https://pastebin.com/hVtZdKJ6

FINDINGS SUMMARY

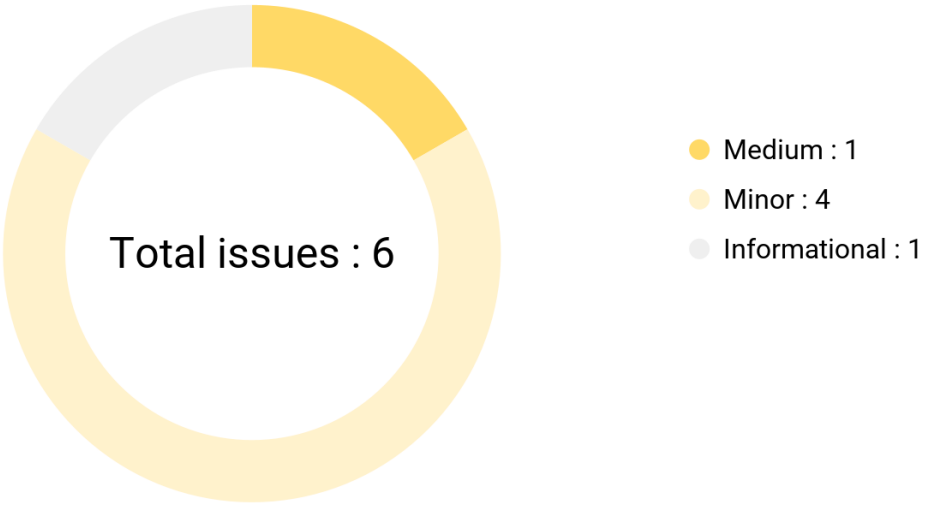
Vulnerability	Total
Critical	0
Major	0
Medium	1
Minor	4
Informational	1

EXECUTIVE SUMMARY

DeFi Dinos was one of the first NFT projects on Avalanche, DeFi Dinos launched on the C-Chain in October 2021. Starting as a simple collection of 4,999 unique dinosaur NFTs, the project has evolved to introduce additional utility for the DeFi Dinos community. In particular, Dinos Fightclub is a PVP and PVE game in which dinosaur NFTs fight each other. To start a game, players go to the Lobby page and select a fighter. Only fighters that are fully healed (100 HP) are eligible. When another player joins the lobby, they are automatically matched with the first player. The gameplay is based on the classic Rock, Paper, Scissors game and there are three basic moves (Bite, Claw, and Whip-tail).

There have been no major or critical issues related to the codebase and all findings listed here are minor or informational. The major security problem is the centralization of privileges.

AUDIT FINDINGS



Code	Title	Severity
CENT-1	Centralization of major privileges	Medium
BLOC-1	Usage of block.timestamp	Minor
THRE-1a	Missing threshold for minor func	Minor
THRE-1b	Missing threshold for minor func	Minor
COMP-1	Unfixed version of compiler	Minor
UINT-1	Unclear error message	Informational

UINT-1 | Wrong uint size

Description

Some variables in the contract are of type uint, but not of the right size.

In order to optimize gas costs when deploying and using the contract, we invite you to assign the right size uint to each variable.

11 errors of this type have been found in the smart contract.

Recommendation

We recommend changing these uint sizes :

```
//Edited code containing appropriate uint sizes
//We didn't noticed it here but notice that you can use lower uint system for token id and winner/loser id if you only have 5000 token identified by numbers from 0 to 4999 and you don't add any other token
//l 47 (since max level is 5)
mapping (uint256 => uint8) public level;
//l 56 (since max health is 100 000, health per second is under 100 000)
uint32 public healthPerSecond;
//l 57 (since duration cannot be that high. You can set it as uint 16, which is already a 18 hours game fight)
uint16 public duration;
//l 148 (since max token level is 5)
uint8 tokenLevel = getLevel(tokenId);
//l 152
uint8 levelTo = (tokenLevel + 1);
//l 161
function setLevel(uint256[] calldata tokenIds, uint8
```

```
newLevel) external onlyOwner {
//l 185
function setHealthPerSecond(uint16 newHPS) external
onlyOwner {
//l 207
function getHealth(uint256 tokenId) public view returns
(uint32) {
//l 245
function calculateWonDXP(uint8 _level, uint8 damage)
public pure returns (uint256 amountDXP) {
//l 261
function attackDamage(uint256 tokenId, uint256 damage)
internal returns (uint32 newHealth) {
//l 270
uint32 healthWinner = getHealth(winner);
```

COMP-1 | Unfixed version of compiler

Description

Dinos Fightclub's contract does not have locked compiler versions, meaning a range of compiler versions can be used. This can lead to differing bytecodes being produced depending on the compiler version, which can create confusion when debugging as bugs may be specific to a specific compiler version(s).

To rectify this, we recommend setting the compiler to a single version, the lowest version tested to be compatible with the code, an example of this change can be seen below.

Recommendation

We recommend fixing the compiler version to the most recent one :

```
//Edited code containing fixed version of compiler
//l 2
pragma solidity 0.8.0;
```

BLOC-1 | Using block.timestamp

Description

The use of block.timestamp can be problematic. The timestamp can be partially manipulated by the miner (see https://cryptomarketpool.com /block-timestamp-manipulation-attack). In this smart contract this is not critical as in the worst case an attacker could manipulate the fight timings.

Recommendation

We fully understand the smart contract's logic of the Dinos Fightclub. Nevertheless, it is still useful to point out this kind of potential security problem.

THRE-1a | Missing threshold for duration setting

Description

The setDuration function for buying does not have a safety threshold. Even though this function is protected by the onlyOwner modifier, it is important to add a threshold to prevent an attacker from setting duration to 0.

2 errors of this type have been found in the smart contract.

Recommendation

We recommend adding these thresholds to concerned functions :

```
//Edited code containing threshold mechanism. Of course,we leave it to you to decide which threshold is the most adapted to the logic of the project
//l 180
function setDuration(uint256 newDuration) external onlyOwner {
    require(newDuration > X, "Duration cannot be under X")
    duration = newDuration;
}
```

THRE-1b | Missing threshold for HPS setting

Description

The setHealthPerSecond function for buying does not have a safety threshold. Even though this function is protected by the onlyOwner modifier, it is important to add a threshold to prevent an attacker from setting health per second to 0 or even 1000000.
2 errors of this type have been found in the smart contract.

Recommendation

We recommend adding these thresholds to concerned functions :

```
//Edited code containing thresholds mechanism. Of course,we leave it to you to decide which threshold is the most adapted to the logic of the project
//l 185
function setHealthPerSecond(uint256 newHPS) external onlyOwner {
    require(newHPS < X && newHPS > Y, "HPS cannot be over X or under Y")
    healthPerSecond = newHPS;
}
```

CENT-1 | Centralization of major privileges

Description

The onlyOwner modifier of the smart contract gives major privileges over it (owner can stop every game, and set any custom level for any player)*. This can be a problem, in the case of a hack, an attacker who has taken possession of these privileged accounts could damage the project and the investors.

*This list is not exhaustive but presents the most sensitive points

Recommendation

We recommend at least to use a multi-sig wallet as owner address, and at best to establish a community governance protocol to avoid such centralization. For more information, see https://solidity-by-example.org/app/multi-sig-wallet/

Global security warnings

These are safety issues for the whole project. They are not necessarily critical problems but they are inherent in the structure of the project itself. Potential attack vectors for these security problems should be monitored.

CENT-1 | Global SPOF (Single Point Of Failure)

The project's smart contracts often have a problem of centralized privileges. The onlyOwner system in particular can be subject to attack. To address this security issue we recommend using a multi-sig wallet, establishing secure project administration protocols and strengthening the security of project administrators.

DISCLAIMER

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement.

This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement.

This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without StaySAFU's prior written consent.This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts StaySAFU to perform a security assessment.

This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance. This report should not be used in any way

to make decisions around investment or involvement with any particular project.

This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk.

StaySAFU's position is that each company and individual are responsible for their own due diligence and continuous security. StaySAFU's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or fun.