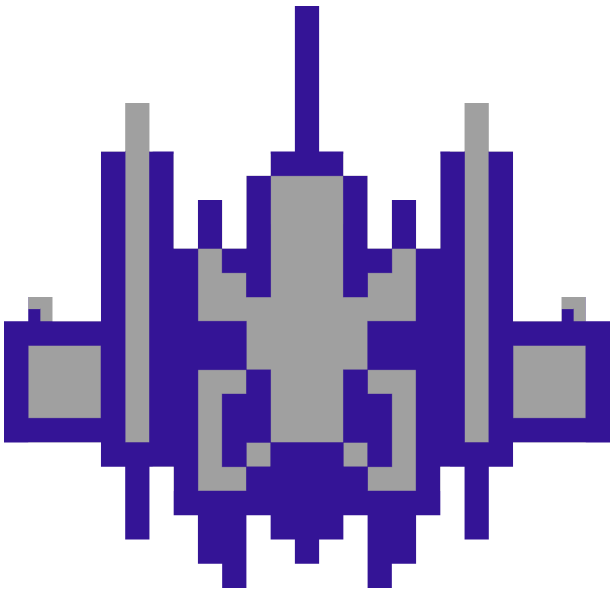

Horse Store Audit Report

Version 1.0

BigBagBoogy.io

January 16, 2024

Horse Store Audit Report

BigBagBoogy

January 16, 2024

Prepared by: BigBagBoogy Lead Auditors: BigBagBoogy -

Table of Contents

- Table of Contents
- Protocol Summary
- Disclaimer
- Risk Classification
- Audit Details
  - Scope
  - Roles
- Executive Summary
  - Issues found
- Findings
- High
  - [H-1] Arithmetic error breaks functionality of `isHappyHorse` within the first 24h after feeding (initializing)
  - [H-2] In Huff, only 1 horse token can be minted due to horseId not updating.
  - *While this does fix `TOTAL_SUPPLY` not updating, it still not simultaneously updates `horseId`*

Protocol Summary

Protocol allows anyone to mint their own horse NFT. feedHorse: Allow anyone to feed a horse NFT. This will make the horse happy for 24h. keep track of happiness: Allow anyone to see if a horse is happy.

Disclaimer

The BigBagBoogy team makes all effort to find as many vulnerabilities in the code in the given time period, but holds no responsibilities for the findings provided in this document. A security audit by the team is not an endorsement of the underlying business or product. The audit was time-boxed and the review of the code was solely on the security aspects of the Solidity implementation of the contracts.

Risk Classification

		Impact		
		High	Medium	Low
	High	H	H/M	M
Likelihood	Medium	H/M	M	M/L
	Low	M	M/L	L

We use the CodeHawks severity matrix to determine severity. See the documentation for more details.

Audit Details

solidity rendition that has been re-written in huff.

Scope

all contracts in /src

1. mintHorse: Allow anyone to mint their own horse NFT.

2. feedHorse: Allow anyone to feed a horse NFT. This will add a block.timestamp to a mapping tracking when the horse was last fed.
3. isHappyHorse: Allow anyone to see if a horse is happy. A horse is happy if and only if they have been fed within the past 24 hours.

Roles

None

Executive Summary

After extensive research I regrettibly was not able to write a functional model where the updated `TOTAL_SUPPLY` transfers it's value to `horseId`.

therefore: I'm looking forward to a deepdive for this codebase in part 2 of the sc security audit course!

Issues found

Severity	Number of issues found
High	2
Medium	0
Low	0
Info	0
Total	2

Findings

High

[H-1] Arithmetic error breaks functionality of `isHappyHorse` within the first 24h after feeding (initializing)

Description: In Solidity, `block.timestamp` is a `uint256`. If a horse is checked to be happy within the first 24 hours of having been minted, the calculation `block.timestamp` -

`HORSE_HAPPY_IF_FED_WITHIN` will result in an arithmetic error. Since the outcome of this calculation would be negative, but `uint256` can't be negative, it wraps around, causing the error as per the 0.8.0 Solidity over/underflow prevention update.

Impact: The current implementation of the `isHappyHorse` function may lead to unexpected behavior and incorrect results when checking the happiness of a horse within the first 24 hours of its creation or feeding due to the arithmetic error.

Proof of Code: Please paste this test at the bottom of Base_Test.t.sol, and run `forge test --mt test_calling_IsHappyHorse_within24hFails -vvvvv`

```
1   function test_calling_IsHappyHorse_within24hFails() public {
2         uint256 horseId = horseStore.totalSupply();
3         vm.warp(horseStore.HORSE_HAPPY_IF_FED_WITHIN() - 1 hours);
4         vm.roll(horseStore.HORSE_HAPPY_IF_FED_WITHIN() - 1 hours);
5         vm.prank(user);
6         horseStore.mintHorse();
7         horseStore.feedHorse(horseId);
8         vm.expectRevert(); // this does not work with huff. comment out
              to see errors in stacktrace for both .sol and .huff
9         horseStore.isHappyHorse(horseId);
10     }
```

Recommended Mitigation: Update the comparison condition in the `isHappyHorse` function to ensure that the subtraction won't result in a negative value.

```
1 function isHappyHorse(uint256 horseId) external view returns (bool) {
2 -     if (horseIdToFedTimeStamp[horseId] <= block.timestamp -
      HORSE_HAPPY_IF_FED_WITHIN) {
3 +     if (block.timestamp >= HORSE_HAPPY_IF_FED_WITHIN +
      horseIdToFedTimeStamp[horseId]) {
4           return false;
5       }
6       return true;
7 }
```

This modification ensures that the subtraction won't result in a negative value, preventing potential arithmetic errors, while still maintaining it's intendid functionality.

[H-2] In Huff, only 1 horse token can be minted due to horseId not updating.

Description: The Huff code is not correctly updating the `TOTAL_SUPPLY` and `horseId` after a horse token is minted, leading to the `ALREADY_MINTED` error when trying to mint the second horse. When minting, the `horseId` param may have been intended to be derived from the `TOTAL_SUPPLY`. Since `TOTAL_SUPPLY` does not update `horseId`, the mint function reverts because the Huff code checks for `horseId`- uniquiness.

Impact: This completely breaks the functionality of the protocol. Multiple users should be able to mint horse tokens.

Proof of Concept: please place this code at the bottom of `Base_Test.t.sol` and run `forge test --mt testTwoUsersCanMintAHorse -vvvvv`. please also add `consol2` to the imports like so: `import {Test, console2, StdInvariant} from "forge-std/Test.sol";`

```
    function testTwoUsersCanMintAHorse() public {
        console2.log("totalSupply", horseStore.totalSupply());
        vm.prank(alice); // user does next line
        horseStore.mintHorse(); // this is the first horse an has
            tokenId 0
        console2.log("totalSupply", horseStore.totalSupply());
        vm.prank(bob);
        horseStore.mintHorse(); // this should be tokenId 1

        horseStore.balanceOf(alice);
        horseStore.balanceOf(bob);
        console2.log("who owns horse 0?", horseStore.ownerOf(0));
        console2.log("who owns horse 1?", horseStore.ownerOf(1));
    }
```

Recommended Mitigation: `TOTAL_SUPPLY` not updating can be fixed by adding the following code:

```
// Increment totalSupply
    [TOTAL_SUPPLY] sload                                  // [
        currentTotalSupply]
    0x01 add                                             // [newTotalSupply]
    [TOTAL_SUPPLY] sstore                                // []
```

This code should be added after the transfer event has been emitted. in the `_MINT` function.

While this does fix `TOTAL_SUPPLY` not updating, it still not simultaneously updates `horseId`

After extensive research I regrettably was not able to write a functional model where the updated `TOTAL_SUPPLY` transfers it's value to `horseId`.

therefore: I'm looking forward to a deepdive for this codebase in part 2 of the sc security audit course!