HoneyFun Stickers Contracts Security Review

Duration: January 20, 2025 - January 21, 2025

Date: January 23, 2025

Conducted by: **KeySecurity**
gkrastenov, Lead Security Researcher

Table of Contents

1 About KeySecurity

KeySecurity is a new, innovative Web3 security company that hires top-talented security researchers for your project. We have conducted over 30 security reviews for various projects, collectively holding over $300,000,000 in TVL. For security audit inquiries, you can reach out to us on Twitter/X or Telegram `@gkrastenov` or check our previous work `here`.

2 About HoneyFun

`Honeyfun AI` is pioneering the co-ownership framework for AI agents specifically tailored for the Berachain ecosystem, focusing on defi, gaming and entertainment. We envision AI agents as pivotal revenue-generating entities in the future, as we believe the era of Utility AI Agents is just beginning, and in the coming years, their untapped potential across every field will be revealed.

3 Disclaimer

Audits are a time, resource, and expertise bound effort where trained experts evaluate smart contracts using a combination of automated and manual techniques to identify as many vulnerabilities as possible. Audits can show the presence of vulnerabilities **but not their absence**.

4 Risk classification

Severity	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

4.1 Impact

- **High** - leads to a significant loss of assets in the protocol or significantly harms a group of users.
- **Medium** - only a small amount of funds can be lost or a functionality of the protocol is affected.
- **Low** - any kind of unexpected behaviour that's not so critical.

4.2 Likelihood

- **High** - direct attack vector; the cost is relatively low to the amount of funds that can be lost.
- **Medium** - only conditionally incentivized attack vector, but still relatively likely.
- **Low** - too many or too unlikely assumptions; provides little or no incentive.

4.3 Actions required by severity level

- **Critical** - client **must** fix the issue.
- **High** - client **must** fix the issue.
- **Medium** - client **should** fix the issue.
- **Low** - client **could** fix the issue.

5 Executive summary

Overview

Project Name	HoneyFun Stickers
Repository	https://github.com/honey-fun/honey-fun-stickers-contracts
Commit hash	d724f990dde6a83a3c64455daa4aed83b240bbcf
Review Commit hash	564572948ddbe7fbb173c2a7d2bb4f6e6a30bf4a
Documentation	https://docs.honey.fun/stickers-campaign
Methods	Manual review

Scope

HoneyFunStickerPacks.sol
HoneyFunStickerPacksMinter.sol

Timeline

January 20, 2024	Audit kick-off
January 21, 2024	Preliminary report
January 22, 2024	Mitigation review

Issues Found

Severity	**Count**
High	0
Medium	1
Low	0
Information	3
Total	**4**

6 Findings

6.1 Medium

6.1.1 User pack IDs were incorrectly added during minting

Severity: *Medium*

Context: HoneyFunStickerPacks.sol#L55

Description: When new stickers are minted, a for loop is used to set the pack type and to push all newly minted NFTs to the `userPacksIds` mapping. However, before adding the `tokenId`, it is incorrectly incremented. As a result, the `tokenId` that is minted and the one added to the user will be different.

Additionally, the function `userPacksIds` will return incorrect IDs for the user, as every ID will be bigger by 1 compared to the actual NFT IDs the user owns.

```
   for (uint256 tokenId = nextTokenId; tokenId < lastTokenId; ) {
            _packTypes[tokenId] = packType;

            unchecked {
                tokenId++;
            }

            _userPacksIds[to][packType].push(tokenId);
        }
```

PoC

```
    // forge test --match-test testCorrectlySettingUserPackToIds
    function testCorrectlySettingUserPackToIds() public {
        // Mint 1 NFT by the minter address
        vm.prank(minter);
        stickerPacks.mint(user, 1, IHoneyFunStickerPacks.PackType.BRONZE);

        // Check if the tokens were minted
        assertEq(stickerPacks.balanceOf(user), 1);
        // NFT with Id = 0 has a BRONZE pack type
        assertEq(
            uint(stickerPacks.typeOfPack(0)), // tokenId
            uint(IHoneyFunStickerPacks.PackType.BRONZE) // type
        );

        uint256[] memory ids = stickerPacks.userPacksIds(
            user,
            IHoneyFunStickerPacks.PackType.BRONZE
        );
        // In array ids should has only 1 NFT with Id = 0
        console.log(ids[0]); // print 1
        assertEq(0, ids[0]); // 0 != 1 revert
    }
```

Recommendation: Increment the tokenId after pushing it to _userPacksIds.

Resolution and Client comment: Resolved. PR: #1

6.2 Information

6.2.1 Unnecessary calling of the _setMinter function

Severity: *Information*

Context: HoneyFunStickerPacks.sol#L38

Description: In the constructor of the `HoneyFunStickerPacks` contract, the internal function `_setMinter` is called to set the minter address. At this time, the minter contract will not be deployed because, in the constructor of the minter contract, the stickers contract should be set.

```
    constructor(
        HoneyFunStickerPacks stickers_,
        address treasury_,
        address owner_,
        address freePacksSigner_,
        uint256[] memory stickerPrices_
    ) Ownable(owner_) {
        _setStickers(stickers_);
        _setTreasury(treasury_);
        _setFreePacksSigner(freePacksSigner_);
        _setStickerPrices(stickerPrices_);
    }
```

In the minter contract, the stickers contract can only be set in the constructor, meaning that the stickers contract should already be deployed. Therefore, when the constructor of the stickers contract is executed, the address of the minter contract will be `address(0)`.

Recommendation: Do not internally call the `_setMinter` function from the constructor of the stickers contract.

Resolution and Client comment: Resolved. PR: #1

6.2.2 Emit event in crucial places

Severity: *Information*

Context: HoneyFunStickerPacks.sol#L140

Description: Emit an event in crucial places, such as in the `_setMinter` function, when the minter contract address is set.

Recommendation: Use the `MinterSet` event from the `IHoneyFunStickerPacks` interface.

Resolution and Client comment: Resolved. PR: #1

6.2.3 Sticker prices can not be updated

Severity: *Information*

Context: HoneyFunStickerPacksMinter.sol#L145

Description: Currently, the sticker prices are set in the constructor of the `Minter` contract. After that, it is not possible to update the sticker prices. In the case of high or low interest in the project, the admin will not be able to change the prices to align with the current market state and maximize profit.

Recommendation: Allow the admin to update sticker prices.

Resolution and Client comment: Resolved. PR: #1