code4rena

Playfi

Smart Contract Security Assessment

Version 2.0

Audit dates: May 02 — May 02, 2024

Audited by: 10hash
zzykxx

Contents

1. Introduction

1.1 About Zenith

Zenith is an offering by Code4rena that provides consultative audits from the very best security researchers in the space. We focus on crafting a tailored security team specifically for the needs of your codebase.

Learn more about us at https://code4rena.com/zenith.

1.2 Disclaimer

This report reflects an analysis conducted within a defined scope and time frame, based on provided materials and documentation. It does not encompass all possible vulnerabilities and should not be considered exhaustive.

The review and accompanying report are presented on an "as-is" and "as-available" basis, without any express or implied warranties.

Furthermore, this report neither endorses any specific project or team nor assures the complete security of the project.

1.3 Risk Classification

SEVERITY LEVEL	IMPACT: HIGH	IMPACT: MEDIUM	IMPACT: LOW
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

2. Executive Summary

2.1 About Playfi

PlayFi turns content into real-time data using powerful AI Agents, enabling web3 features for games, streams, sports and more.By building a layer that allows real time data to be extracted from live events in a verifiable way, PlayFi unlocks markets worth hundreds of billions of dollars with new methods of audience participation.

2.2 Scope

Repository	PlayFi-Labs/node-license-sale-contracts/tree/develop/contracts
Commit Hash	219c9ac0c963074cdd032f1188d0b5b7f6e3ccbb
Mitigation Hash	e01669a37ff9d8a533c4cecb005f8c3d0e4e30c6

2.3 Audit Timeline

DATE	EVENT
May 02, 2024	Audit start
May 02, 2024	Audit end
Nov 14, 2024	Report published

2.4 Issues Found

SEVERITY	COUNT
Critical Risk	0
High Risk	3
Medium Risk	1
Low Risk	1
Informational	1
Total Issues	6

3. Findings Summary

ID	DESCRIPTION	STATUS
H-1	Excess payment is not reimbursed causing user's to spend more than necessary for the license	Resolved

H-2	Users can bypass limits and purchase more licenses than allowed by re-entering functions	Resolved
H-3	The functions `claimLicensePartner()` and `claimLicensePublic()` don't update state variables allowing users to purchase more licenses than expected	Resolved
M-1	Reorg's may cause licenses to be sold at 0 price	Resolved
L-1	Reorg's can cause referral code and associated fees to be assigned to another user	Resolved
I-1	`getTier` handles `isWhitelist` incorrectly	Resolved

4. Findings

4.1 High Risk

A total of 3 high risk findings were identified.

[H-1] Excess payment is not reimbursed causing user's to spend more than necessary for the license

Severity: High	Status: Resolved

Context:

- PlayFiLicenseSale.sol#L210
- PlayFiLicenseSale.sol#L261

Description: When the payment made by user's is in excess, the remaining ETH is not returned

```
    function claimLicensePublic(uint256 amount, uint256 tier, string
memory referral) public payable {

        ......

        (uint256 toPay, uint256 commission,) =
paymentDetailsForReferral(amount, tier, referral, false);
        if(msg.value < toPay) revert InsufficientPayment();
```

Since the payment amount is dependent on the referral discount which in turn is dependent on the claimCount, the actual price at the time of execution can end up being lower than at the start of the block. Hence a user can end up making excess payment which will not be refunded

```
    function paymentDetailsForReferral(uint256 amount, uint256 tier,
string memory referral, bool isWhitelist) public view returns (uint256
toPay, uint256 commission, uint256 discount) {

        .......

        if(referrals[referral].receiver != address(0)) {
=>       uint256 totalClaims = referrals[referral].totalClaims;
```

```
            if(totalClaims < 20) {
                commission = fullPrice * 10 / 100;
            } else if (totalClaims < 40) {
                commission = fullPrice * 11 / 100;
```

Recommendation: Refund excess payment

Playfi: Fixed with the following @e01669a37ff9.. and this one @3cd84feb51780..

[H-2] Users can bypass limits and purchase more licenses than allowed by re-entering functions

Severity: High	Status: Resolved

Context:

- PlayFiLicenseSale.sol#L181
- PlayFiLicenseSale.sol#L187
- PlayFiLicenseSale.sol#L212
- PlayFiLicenseSale.sol#L242

Description: The `PlayFiLicenseSale` contract sends a commission in ETH to a `referreal` address if a referral code is used during license purchases. This is generally done via a low-level call performed before state variables are updated, which allows for re-entrancy in order to bypass limits and purchase more licenses than allowed.

We will take the claimLicensePublic() function as an example:

```
function claimLicensePublic(uint256 amount, uint256 tier, string memory
referral) public payable {
    if(!publicSaleActive) revert PublicSaleNotActive();
    if(tiers[tier].totalClaimed + amount > tiers[tier].totalCap) revert
TotalTierCapExceeded();
    if(claimsPerTierPerAddress[tier][msg.sender] + amount >
tiers[tier].individualCap) revert IndividualTierCapExceeded();
    (uint256 toPay, uint256 commission,) =
paymentDetailsForReferral(amount, tier, referral, false);
    if(msg.value < toPay) revert InsufficientPayment();
    if(commission > 0) {
        (bool sent, ) = payable(referrals[referral].receiver).call{
value: commission }("");
        if (!sent) revert CommissionPayoutFailed();
        emit CommissionPaid(referral, referrals[referral].receiver,
commission);
    }
    tiers[tier].totalClaimed += amount;
    publicClaimsPerAddress[msg.sender] += amount;
    totalLicenses += amount;
    referrals[referral].totalClaims += amount;
    emit PublicLicensesClaimed(msg.sender, amount, tier, toPay,
referral);
}
```

Alice wants to purchase 4 licenses, but only 2 are left. As a workaround, she can:

1. Create and deploy malicious contract `MAL.sol` that triggers a call to claimLicensePublic() to purchase licenses whenever `ETH` are received.
2. Call the function setReferral() from `MAL.sol` in order to create a referral code `MAL` where the receiver is `MAL.sol`.
3. Call the function claimLicensePublic() by using `MAL` as referral code and purchasing the `2` licenses left.
4. `MAL.sol` will receive `ETH` for the commission and will internally call the function again claimLicensePublic() by purchasing `2` extra linceses. This is possible because the state variables have not been updated yet.

Recommendation: Either:

- Use nonReentrant modifiers on all impacted functions
- Transfer the referreal/partners commissions at the end of each function, after the state variables have been updated

Playfi: Fixed with the following commit

Zenith: Verified.

[H-3] The functions `claimLicensePartner()` and `claimLicensePublic()` don't update state variables allowing users to purchase more licenses than expected

Severity: High	Status: Resolved

Context:

- PlayFiLicenseSale.sol#L173
- PlayFiLicenseSale.sol#L205

Description: The functions claimLicensePartner() and claimLicensePublic() use the variables `partnerClaimsPerTierPerAddress` and `claimsPerTierPerAddress` respectively to ensure that individual caps are respected, reverting if an user attempts to purchase more licenses than allowed.

Both variables will always `0` because their values are never updated by the new purchased licenses. This makes it possible for users to purchase more licenses than they should

Recommendation:

- In claimLicensePartner() increase the variable `partnerClaimsPerTierPerAddress[][][]` by the `amount` of licenses purchased
- In claimLicensePublic() increase the variable `claimsPerTierPerAddress[][]` by the `amount` of licenses purchased

Playfi: Fixed with the following commit

Zenith: Verified.

4.2 Medium Risk

A total of 1 medium risk findings were identified.

[M-1] Reorg's may cause licenses to be sold at 0 price

Severity: Medium	Status: Resolved

Context:

- PlayFiLicenseSale.sol#L143
- PlayFiLicenseSale.sol#L162

Description: The payment amount for `claimLicenseEarlyAccess` and `claimLicenseFriendsFamily` is determined by `tiers[1].price` while the ability to start claiming is dependent on other variables like `earlyAccessSaleActive` and `earlyAccessMerkleRoot` and these are set by different entities

```
    function claimLicenseEarlyAccess(uint256 amount, bytes calldata data,
bytes32[] calldata merkleProof) public payable {
        if(!earlyAccessSaleActive) revert EarlyAccessSaleNotActive();
        (uint256 index, uint256 claimCap) = abi.decode(data,
(uint256,uint256));
        uint256 claimedLicenses =
earlyAccessClaimsPerAddress[msg.sender];
        if(amount + claimedLicenses > claimCap) revert
IndividualClaimCapExceeded();
        bytes32 node = keccak256(abi.encodePacked(index, msg.sender,
claimCap));
        if (!MerkleProof.verify(merkleProof, earlyAccessMerkleRoot,
node)) revert InvalidProof();
        uint256 toPay = tiers[1].price * amount;
```

The code is also planned to be deployed on Polygon where reorgs are common. This creates a possible scenario where the tier price update transaction gets pushed to a later block and a user claim transaction occurs earlier. This will cause the license to be sold at 0 price

Recommendation: Ensure sufficient time interval (atleast >1min) between the tier setting tx and the status update transaction / check if tier[1].price is zero and revert

Playfi: Fixed with the following commit

Zenith: Verified

4.3 Low Risk

A total of 1 low risk findings were identified.

[L-1] Reorg's can cause referral code and associated fees to be assigned to another user

Severity: Low	Status: Resolved

Context:

- PlayFiLicenseSale.sol#L205
- PlayFiLicenseSale.sol#L349-L350

Description: The receiver of the referral commission is solely determined the referral code which is served on a first come basis

```
    function setReferral(string memory code) public {
        _setReferral(code, msg.sender);
    }
```

The project also plans to deploy on Polygon where reorgs are common. This allows for a scenario where the referral code is obtained by another user and the commission payments still happen

```
    function claimLicensePublic(uint256 amount, uint256 tier, string
memory referral) public payable {
```

Reorg scenario:

```
User A claims referral code A
User B also makes a tx to obtain referral code A which fails initially
Associated user's with A claims license with referral code A
Due to reorg, user B's tx occur earlier than user A's tx
The commission payments from the user's go to user B instead of A
```

Recommendation Ensure sufficient time interval (atleast >1min) between the user's claim tx's and the referral code setting

Playfi: Won't fix

4.4 Informational

A total of 1 informational findings were identified.

[I-1] `getTier` handles `isWhitelist` incorrectly

Severity: Informational	Status: Resolved

Context:

- PlayFiLicenseSale.sol#L325

Description: The `getTier` function doesn't handle the `isWhitelist` variable correctly and returns the `whitelistTier` instead of normal `tier` when `isWhitelist` is false and vice-versa

```
    function getTier(uint256 id, bool isWhitelist) public view
returns(Tier memory tier) {
        if(isWhitelist) {
            tier = tiers[id];
        } else {
            tier = whitelistTiers[id];
        }
    }
```

Recommendation: Apply the following change

```
     function getTier(uint256 id, bool isWhitelist) public view
returns(Tier memory tier) {
-        if(isWhitelist) {
+        if(!isWhitelist) {
            tier = tiers[id];
        } else {
            tier = whitelistTiers[id];
```

Playfi: Fixed with the following commit

Zenith: Verified