SHERLOCK

Security Review For Fluent

Collaborative Audit Prepared For:	Fluent
Lead Security Expert(s):	TessKimy
	vinica_boy
Date Audited:	December 10 - December 12, 2025

Introduction

Fluent is the first blended execution network where EVM, WASM, and SVM contracts talk to each other like they're written in the same language. No bridges. No friction. Just pure expressivity.

Scope

Repository: fluentlabs-xyz/fluent-nft-sale

Audited Commit: 51bde412b20d754421068b7b77d66b31b3bb8fee

Final Commit: 3025dd04ec8b5a7b967c9c5f26d9a90e2ee5a97c

Files:

- src/FluentNFTSale.sol
- src/interface/IFluentNFTSale.sol

Final Commit Hash

3025dd04ec8b5a7b967c9c5f26d9a90e2ee5a97c

Findings

Each issue has an assigned severity:

- High issues are directly exploitable security vulnerabilities that need to be fixed.
- Medium issues are security vulnerabilities that may not be directly exploitable or may require certain conditions in order to be exploited. All major issues should be addressed.
- Low/Info issues are non-exploitable, informational findings that do not pose a security risk or impact the system's integrity. These issues are typically cosmetic or related to compliance requirements, and are not considered a priority for remediation.

Issues Found

High	Medium	Low/Info
0	0	4

Issues Not Fixed and Not Acknowledged

High	Medium	Low/Info
0	0	0

Issue L-1: `getMintLimit` return values can be optimized for edgecases [RESOLVED]

Source: https://github.com/sherlock-audit/2025-12-fluent-dec-10th/issues/3

Summary

`getMintLimit` function returns 0 value for unlimited case and it always returns tier mint limits in other cases. These return values don't account MAX_SUPPLY. It can be optimized with max supply value because it cannot pass that limit anyway.

Code Snippet

https://github.com/sherlock-audit/2025-12-fluent-dec-10th/blob/e5a4c53522ec915a32b9787fe02fac96c556b272/fluent-nft-sale/src/FluentNFTSale.sol#L567-L580

Tool Used

Manual Review

Recommendation

Consider returning max supply minus total minted value for some cases:

```
function getMintLimit(uint8 tier) external view returns (uint8) {
    require(tier > 0 && tier <= NUM_TIERS, InvalidTier());
    MainStorage storage $ = _getStorage();

    uint8 maxActive = getMaxActiveStage();
    if (maxActive == 0) return 0;

    StageType stageType = $.stages[maxActive].stageType;
    if (stageType == StageType.Public || tier > MAX_PRESALE_TIER) {
        return MAX_SUPPLY - $.totalMinted; // limited by max supply cap
    }
    return $.presaleTierConfigs[tier].mintLimit > MAX_SUPPLY - $.totalMinted ?
     ↪  MAX_SUPPLY - $.totalMinted : $.presaleTierConfigs[tier].mintLimit;
}
```

Discussion

d1r1

Fixed. Return type changed to uint64 to support values up to MAX_SUPPLY (5000).

Issue L-2: # Redundant handling of mint recipient being EOA/contract [RESOLVED]

Source: https://github.com/sherlock-audit/2025-12-fluent-dec-10th/issues/4

Summary

The logic for minting in `FluentNFTSale::_executeMint()` explicitly handles whether to use `mint()` or `safeMint()` for the NFT:

```
// Mint tokens (use _mint for EOA, _safeMint for contracts)
if (recipient.code.length == 0) {
    for (uint256 id = startTokenId; id < nextId;) {
        _mint(recipient, id);
        unchecked {
            ++id;
        }
    }
} else {
    for (uint256 id = startTokenId; id < nextId;) {
        _safeMint(recipient, id);
        unchecked {
            ++id;
        }
    }
}
```

There is no need to do that and `safeMint()` can always be used as it internally does the check and execute `onERC721Received` check only for contracts (addresses with code.lenght > 0).

Recommendation

Consider always using `safeMint()`.

Discussion

d1r1

fixed

Issue L-3: Redundant two loops can be combined in a single one [RESOLVED]

Source: https://github.com/sherlock-audit/2025-12-fluent-dec-10th/issues/5

Summary

The logic for minting in `FluentNFTSale::_executeMint()` have two loops to handle the accounting for token tiers and actual minting.

```
for (uint8 i = 0; i < count;) {
            $.tokenTiers[nextId] = tier;
            unchecked {
                ++nextId;
                ++i;
            }
        }
        $.nextTokenId = nextId;

        // Mint tokens (use _mint for EOA, _safeMint for contracts)
        if (recipient.code.length == 0) {
            for (uint256 id = startTokenId; id < nextId;) {
                _mint(recipient, id);
                unchecked {
                    ++id;
                }
            }
        } else {
            for (uint256 id = startTokenId; id < nextId;) {
                _safeMint(recipient, id);
                unchecked {
                    ++id;
                }
            }
        }
```

Loops can be combined in a single one which does both - save the token tier and mint in the same traversal.

Recommendation

Consider refactoring to:

```
for (uint8 i = 0; i < count;) {
    $.tokenTiers[nextId] = tier;
    _safeMint(recipient, nextId);
    unchecked {
        ++nextId;
        ++i;
    }
}
$.nextTokenId = nextId;
```

Discussion

d1r1

fixed

Issue L-4: Pausing public sale via pauseStage would allow users to mint NFTs with prices based on pre-sale stages [RESOLVED]

Source: https://github.com/sherlock-audit/2025-12-fluent-dec-10th/issues/6

Summary

Pausing stage 4 (Public) via `pauseStage()` causes `getMaxActiveStage()` to return stage 3, allowing users to mint NFTs at cheaper presale prices instead of public prices.

Vulnerability Detail

The `getMaxActiveStage()` function loops from stage 4 down to 1 and returns the first active stage. When stage 4 is paused, it returns stage 3 (Presale), and pricing logic uses this stage's type to determine prices.

Impact

Users can mint NFTs at presale prices (e.g., 0.2-0.4 ETH for tiers 1-3) instead of paying the higher public price (0.5 ETH).

Code Snippet

https://github.com/sherlock-audit/2025-12-fluent-dec-10th/blob/e5a4c53522ec915a32b9787fe02fac96c556b272/fluent-nft-sale/src/FluentNFTSale.sol#L224-L228

https://github.com/sherlock-audit/2025-12-fluent-dec-10th/blob/e5a4c53522ec915a32b9787fe02fac96c556b272/fluent-nft-sale/src/FluentNFTSale.sol#L513-L521

Tool Used

Manual Review

Recommendation

Consider using `pauseSale()` in this case and document the behavior.

Discussion

d1r1

fixed

Disclaimers