FAIVR Audit Notes for D23E

From: Old School GmbH (Technical Team) **Date:** February 19, 2026 **Re:** Smart Contract Audit · Areas of Focus

Hi Arthur,

Thanks for the thorough quote · we're going ahead with the smart contract audit (5,800 CHF). Below are areas we'd like you to pay particular attention to, based on our internal review.

· High Priority

1. FeeModule · ETH transfer gas limit

`_sendETH` uses `call{value: amount, gas: 10000}`. If the recipient is a smart contract wallet (e.g., Gnosis Safe, smart account), 10k gas may be insufficient. The fallback to `_pendingWithdrawals` is meant to handle this, but please verify:

- Can the pull-withdrawal path ever leave funds stuck?
- Is the `_pendingWithdrawals` accounting correct if `_sendETH` fails for multiple recipients in a single `settleTask` call?
- Can `withdrawPending` itself be griefed?

2. FeeModule · Low-level staticcall for agent owner lookup

`settleTask` and `settleTaskFor` use a raw `staticcall` with `abi.encodeWithSignature("ownerOf(uint256)", ...)` instead of a typed interface. Please assess:

- What happens if the identity registry is upgraded and `ownerOf` behavior changes?
- Is the `data.length >= 32` check sufficient, or could a malicious/broken registry return data that decodes to an attacker-controlled address?

3. FeeModule · settleTask / settleTaskFor code duplication

These two functions contain nearly identical business logic. Please flag if there are any subtle differences that could lead to inconsistent behavior, especially after future upgrades where one gets patched but not the other.

4. Genesis Program · Client vs Agent semantics

`_genesisAgents` maps **client addresses**, not agent IDs. This means a genesis client gets fee exemptions on tasks for *any* agent, not just their own. Please confirm whether this is a security concern or just a design choice we should document.

5. Router · ERC20 approval target

`Router.registerAndFund()` calls `feeModule.fundTaskFor()`, which does `safeTransferFrom(client, address(feeModule), amount)`. The client must have approved the **FeeModule** contract, not the Router. Please verify there's no path where a user approves the Router expecting it to work, loses the tx, and their approval is left dangling.

· Medium Priority

6. ReputationRegistry · Duplicate event parameter

In `giveFeedback`, the `NewFeedback` event emission appears to pass `tag1` twice:

```
emit NewFeedback(agentId, msg.sender, feedbackIndex, value,
valueDecimals, tag1, tag1, tag2, ...);
```

Likely should be `tag1, tag2` · please confirm.

7. VerificationRegistry · tokenURI linear scan

The `tokenURI` function scans from `1` to `_nextTokenId + 100` to find the agentId for a given tokenId. The `+ 100` is arbitrary. Please assess if this could miss valid tokens or cause gas issues at scale.

8. VerificationRegistry · Soulbound NFT ownership after agent transfer

When an agent NFT is transferred, the soulbound verification NFT remains with the original owner. The verification *record* still points to the agentId, but the NFT holder is now stale. Is there a scenario where the old owner could exploit this?

9. Storage gap verification

We added genesis program storage (3 slots) to FeeModule and adjusted the gap to 46. Please verify all storage gap arithmetic across all 6 contracts is correct for safe UUPS upgrades.

10. No deadline bounds on fundTask

`fundTask` accepts any `deadline` value. A user could:

- Set `deadline = 1` · fund and immediately reclaim in the same block
- Set `deadline = type(uint256).max` · funds locked essentially forever

Please assess if either edge case creates issues.

· Things we believe are solid

- ReentrancyGuard on all state-changing FeeModule functions
- Checks-effects-interactions in Identity's `_registerInternal` (state set before `_safeMint`)
- EIP-712 + nonce-based replay protection on `setAgentWallet`
- Pausable on escrow operations
- SafeERC20 for all ERC-20 transfers
- Access control on all admin/upgrade functions

Let us know if you need any additional context or access. Looking forward to working together!

Best, Old School GmbH