**Adding EIP-4337 to the dynamic NFT of Boonty**

**Author**: Hortense Vallat

**Created**: 2024-03-10

**Abstract**

This document delves into the implementation of EIP-4337 within Boonty’s dynamic NFT, emphasizing the benefits that such a functionality brings to the user experience.

**Motivation**

In most Web3 applications or applications like Boonty, users are often required to understand the intricacies of digital wallets, including the necessity for wallet addresses and sufficient funds. This requirement poses a barrier to a significant part of the population unfamiliar with these technologies. Account abstraction addresses these challenges, making NFT interactions effortless and user-friendly.

**Benefits of Account Abstraction**

Traditionally, users seeking to mint an NFT have to interact with a contract account and incur associated gas fees, potentially introducing security risks such as private key exposure (for instance, people make mistakes and loose access to their private keys). Account abstraction provides several notable advantages:

* **Simplified user interaction** (see picture below): Eliminates the need for user to create and fund wallet accounts.
* **Security enhancement**: Mitigates potential loss of access due to user mistake or mishandling of private keys introducing social recovery features or milti-sig. Moreover, it is more difficult for attackers to steal from a specific account because the user’s private key is not used to directly control the assets contained in the account but only to authorize the execution of a smart contract.
* **Expanded functionalities:** Introduces permission controls, enabling multi-signature requirements for transaction approvals.
* **Gas abstraction (paymaster functionality):** Allows third party to pay for the gas fees
* **Enhanced user transparency**: Provides users with increased visibility and understanding of transaction processes, fostering trust and transparency.

Thus, introducing EIP-4337 to Boonty’s application not only addresses the usability challenges but also elevates security and overall transparency for a more inclusive and user-centric experience.

Une image contenant texte, capture d’écran, multimédia, logiciel

Description générée automatiquement

**Specification: implement in Boonty Application**

To implement EIP-4337 in Boonty, one should follow these steps:

**1. UserOperation Object Creation:**

* Users initiate the EIP-4337 process by creating a UserOperation object.
* UserOperation is a pseudo-transaction object representing a user’s transaction intent. It serves as a replacement for a conventional transaction, with fields tailored to the account contract rather than the EOA.
* Configure the sender as the contract account, responsible for verifying the signature, receiving, and executing the calldata.

**2. Bundling and Execution of UserOperations:**

* UserOperation objects are transmitted to a dedicated mempool, where bundlers handle them and include them in a unified transaction to the entry point contract.
* Bundlers conduct sanity checks to ensure the UserOperation is appropriately formed with valid inputs for its fields.
* A simulation is run by the bundler using the **EntryPoint.simulateValidation()** function to assess the UserOperation's validity and capability to cover execution costs.
* The simulation includes a call to **sender.validateUserOp()**, which validates the UserOperation's signature and overall integrity.
* If all checks pass, the UserOperation is deemed valid, added to the mempool, and becomes eligible for bundling and transaction inclusion.

**3. Handling UserOperations at Contract Level:**

* The account typically incorporates an **execute()** function to interpret the calldata, which may represent a series of calls for the contract account to execute.

The architectural diagram below shows how EIP-4337 works in Ethereum.

**Une image contenant texte, capture d’écran

Description générée automatiquement**

Source : https://web3edge.io/research/what-is-eip-4337-and-account-abstraction/