Poshmark API Login Flow: A Detailed Analysis

This report breaks down the Poshmark iOS application's login process, as demonstrated by the Python test script. The flow is not a simple submission of a username and password; instead, it employs a robust, multi-step **challenge-response mechanism** designed to prevent automated access and ensure requests come from a legitimate device.

Step 1: The Challenge Request

Before any credentials are sent, the application must first obtain a unique, single-use "challenge" from the Poshmark servers.

- **Endpoint**: `POST /api/devices/{device_id}/challenges`
- **Purpose**: To initiate the login sequence and receive a temporary token that must be cryptographically signed.
- **Process**: The script sends a request containing static device and visitor IDs. The server validates these identifiers and, if they are acceptable, generates a unique challenge string.
- **Server Response**: A successful response contains the challenge string, for example:
  `{`
-   `"data": "cad8e535bcd5d725136fa8891a2ac999"`
- `}`
-
-

This challenge is only valid for a single login attempt and a short period.

Step 2: The Login Request & Integrity Verification

This is the core of the login process, where the user's credentials are submitted along with critical security tokens that prove the integrity of the device and application.

- **Endpoint**: `POST /api/auth/users/access_token`
- **Purpose**: To exchange the user's credentials and the signed challenge for a valid session token.

The Critical Security Payload

The request body contains the username and password, but more importantly, it includes two complex security parameters that were sniffed during the initial analysis:

1. `device_integrity_hash`: This is the most crucial security element. In the real application, the challenge string from Step 1 is passed to Apple's **App Attest service**. This service uses the iPhone's secure hardware to create a cryptographic signature of the challenge. This signature proves to Poshmark's server that the request is coming from their legitimate, unmodified app on a genuine Apple device.

2. `iobb`: This is a large, obfuscated data blob. It likely serves as a secondary device fingerprint, containing a wealth of information about the device's state, configuration, and environment. The server can analyze this blob to detect signs of emulators, jailbroken devices, or other anomalies associated with fraudulent activity.