In order to secure the endpoints in our current Koa-based API, I propose we consider the following steps:

1. **User Authentication:** Implementing user authentication can restrict who has access to our endpoints. This can be achieved by using JSON Web Tokens (JWTs) or a similar authentication scheme. Upon verifying their credentials, users would be provided with a token, which they must supply with each request to gain access to secured endpoints.
2. **Rate Limiting:** This is a critical measure to prevent abuse and potential denial-of-service attacks. By restricting the number of requests a client or an IP address can make within a specific time frame, we can protect our server resources and maintain availability for all users. Libraries like koa2-ratelimiter can help with this.
3. **Input Validation:** Server-side input validation is an essential step to prevent threats like SQL injection or cross-site scripting (XSS). We can utilize packages like **joi** to validate the incoming data before processing it.
4. **HTTPS:** Serving the API over HTTPS is vital to protect the data while it's in transit between the client and server. This measure encrypts the data, thereby safeguarding it from potential eavesdropping or tampering.
5. **OAuth:** The coding challenge suggested using Spotify API and OAuth "Client Credentials Flow" for authorization, and we've implemented this. This OAuth strategy provides a way for our API to access and interact with the data from Spotify without exposing our credentials.