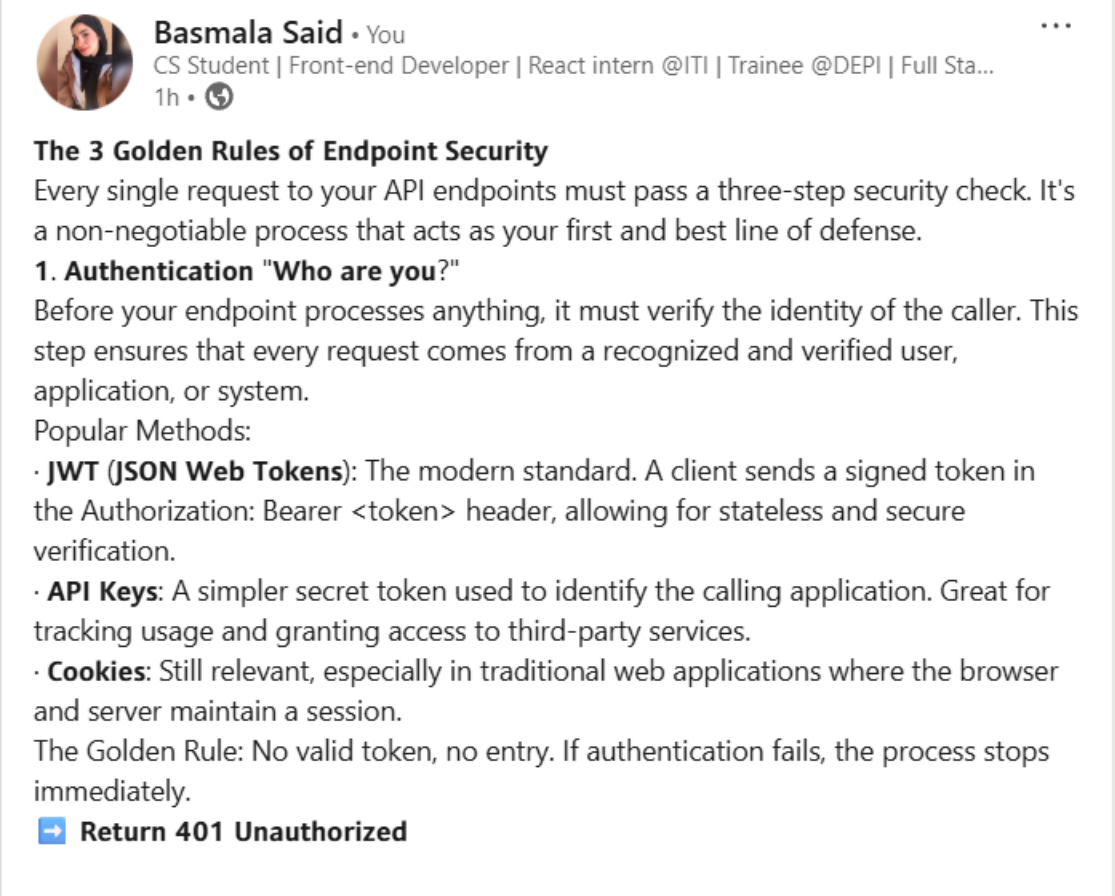
DAY02(API)

Linked-In article about 3 main ways to secure your endpoints



Basmala Said • You
CS Student | Front-end Developer | React intern @ITI | Trainee @DEPI | Full Sta...
1h •

The 3 Golden Rules of Endpoint Security
Every single request to your API endpoints must pass a three-step security check. It's a non-negotiable process that acts as your first and best line of defense.
1. **Authentication "Who are you?"**
Before your endpoint processes anything, it must verify the identity of the caller. This step ensures that every request comes from a recognized and verified user, application, or system.
Popular Methods:
· **JWT (JSON Web Tokens)**: The modern standard. A client sends a signed token in the Authorization: Bearer <token> header, allowing for stateless and secure verification.
· **API Keys**: A simpler secret token used to identify the calling application. Great for tracking usage and granting access to third-party services.
· **Cookies**: Still relevant, especially in traditional web applications where the browser and server maintain a session.
The Golden Rule: No valid token, no entry. If authentication fails, the process stops immediately.
➡️ **Return 401 Unauthorized**

2. Authorization "What are you allowed to do?"
Identity confirmed. Now what? Just because we know who the caller is doesn't mean they can do everything. Authorization is the critical step of checking permissions for the specific action requested.
Common Strategies:
· **Role-Based Access Control (RBAC)**: Is the caller an Admin, Editor, or Viewer? Each role has a different set of permissions.
· **Claims-Based Access**: The token itself can contain fine-grained permissions (claims) like can: delete_post or scope:read_profile. This allows for much more granular control.
The Golden Rule: Respect the boundaries. If the authenticated user does not have the required role or claim for the action.
➡️ **Return 403 Forbidden**

3. Input Validation "Is the data you sent clean and correct?"
This is your final guardian. Even an authenticated and authorized user can send malicious or malformed data. Input validation is the process of rigorously sanitizing and checking every piece of data before your business logic touches it.
You must validate everything:
· **Route Parameters**: Is the {id} in /orders/{id} a valid integer, not a malicious script?
· **Query Strings**: In ?page=1&limit=50, are the values numbers and within an acceptable range?
· **Request Body**: Does the JSON payload match your expected schema? Are all required fields present? Are there unexpected fields that could cause issues? Are data types correct?
The Golden Rule: Trust nothing. If any part of the incoming data is invalid, malformed, or suspicious.
➡️ **Return 400 Bad Request**