Security Best Practices in Django REST Framework (DRF)

1. Authentication
2. Use proper authentication mechanisms:

    - SessionAuthentication
    - TokenAuthentication
    - JWT Authentication (recommended for APIs)

3. Permissions
4. Control who can access your endpoints:

    - IsAuthenticated → only logged-in users
    - IsAdminUser → only admin users
    - Avoid AllowAny on sensitive endpoints

5. Throttling / Rate Limiting
6. Limit the number of requests per user/IP:
    - UserRateThrottle
    - AnonRateThrottle
7. Prevent abuse and DDoS attacks
8. Input Validation
9. Always validate incoming data using Serializers
10. Never trust raw user input
11. HTTPS / SSL
12. Encrypt all data in transit with HTTPS
13. CORS (Cross-Origin Resource Sharing)
14. Control which domains can access your API
15. Use django-cors-headers for configuration
16. CSRF Protection
17. Protect session-based endpoints from Cross-Site Request Forgery attacks
18. Sensitive Data Handling

19. Don’t expose secrets (passwords, API keys) in code
20. Use environment variables for credentials
21. Logging and Monitoring
22. Keep logs of login attempts, failed requests, and webhook calls
23. Monitor suspicious activity
24. Webhooks Security
    - Verify incoming webhook requests (use signatures or tokens)
    - Only accept requests from trusted sources
25. Anti-Brute Force / Account Protection
    - Implement login throttling
    - Limit repeated login attempts
26. Error Handling / Information Exposure
    - Don’t expose stack traces or internal errors to users
    - Return generic error messages
27. Data Encryption at Rest
    - Encrypt sensitive data in the database if needed