

# Kong & API Security

---

## 1. Mục tiêu

Tập trung xây dựng một **ứng dụng chấm công đơn giản** (check-in / check-out) với tài liệu **chuyên sâu** về:

- Cấu hình Kong làm API Gateway (JWT validation, rate limiting, IP restriction)
- Thiết kế token (access + refresh), expiry, rotation, revocation
- Cách tích hợp backend (ví dụ Node.js/Express) để cấp và verify token
- Các best practices vận hành và giám sát

Tài liệu loại bỏ hoàn toàn phần face-scan, camera, hoặc các cơ chế xác thực phức tạp khác.

---

## 2. Kiến trúc tổng quan (gọn)

```
[Mobile App]
  | Authorization: Bearer <access_token>
  v
[Kong API Gateway]
  | JWT plugin, rate-limiting, IP restrictions
  v
[Attendance Service (Node.js/Express)]
  | business logic: checkin/checkout
  v
[Postgres / Redis (revocation)]
```

---

## 3. API Design (Minimal)

### 3.1. Endpoints

- POST /auth/login -> trả về { accessToken, refreshToken, expiresIn }
- POST /auth/refresh -> dùng refresh token lấy access token mới
- POST /attendance/checkin -> protected
- POST /attendance/checkout -> protected
- POST /auth/logout -> thu hồi refresh token

### 3.2. Request/Response ví dụ

#### **POST /auth/login**

```
{ "username": "240202005", "password": "..." }
```

## Response

```
{  
  "accessToken": "eyJhbGci... ",  
  "refreshToken": "rJk23... ",  
  "expiresIn": 900  
}
```

(access token expiresIn: 900s = 15 phút)

---

## 4. Thiết kế Token (Access + Refresh)

### 4.1. Access Token (JWT)

- Ngắn hạn (5–30 phút). Khuyến nghị 10–15 phút cho mobile app.
- Chứa các claim tối thiểu: `sub` (user id), `iat`, `exp`, `iss`, `aud`, `scope`.
- **Không** chứa thông tin nhạy cảm (password, PII đầy đủ).

#### Sample payload

```
{  
  "sub": "240202005",  
  "iss": "attendance-auth",  
  "aud": "attendance-api",  
  "iat": 1700000000,  
  "exp": 1700009000,  
  "scope": "attendance:write"  
}
```

### 4.2. Refresh Token

- Dài hạn hơn (7–30 ngày tuỳ chính sách).
- Có thể là opaque token (random string) lưu trên DB/Redis kèm `userId`, `deviceId`, `expiresAt`.
- Khi refresh token bị leak, attacker có thể lấy access token mới — vì vậy:
  - Gắn nhiều cơ chế: rotate-on-use, store last-used-ip/device, và hỗ trợ revoke.

### 4.3. Rotation & Revoke

- **Rotate-on-use:** khi dùng refresh token để lấy access token mới → cấp refresh token mới, thu hồi token cũ.
  - **Revocation list:** lưu token bị thu hồi (access token không thể trực tiếp revoke nếu stateless JWT) — giải pháp: rút ngắn expiry hoặc sử dụng reference token / introspection.
-

## 5. Cài đặt Kong — PoC Local (Code)

### 5.1. Docker Compose (Kong DB-less)

```
version: '3.8'
services:
  kong:
    image: ghcr.io/kong/kong:3.6
    environment:
      KONG_DATABASE: 'off'
      KONG_DECLARATIVE_CONFIG: /kong/kong.yml
      KONG_PROXY_ACCESS_LOG: /dev/stdout
      KONG_ADMIN_ACCESS_LOG: /dev/stdout
    volumes:
      - ./kong.yml:/kong/kong.yml
    ports:
      - '8000:8000'
      - '8001:8001'
```

### 5.2. Định nghĩa service & route (Admin API)

```
# tạo service
curl -i -X POST http://localhost:8001/services \
  --data name=attendance-service \
  --data url='http://attendance-api:8080'

# tạo route
curl -i -X POST http://localhost:8001/services/attendance-service/routes \
  --data 'paths[]=/attendance' --data name=attendance-route
```

### 5.3. Kích hoạt JWT plugin

```
curl -i -X POST http://localhost:8001/services/attendance-service/plugins \
  --data 'name=jwt' \
  --data 'config.claims_to_verify=exp'
```

Kong sẽ validate signature & **exp** claim.

### 5.4. Thêm rate-limiting theo credential (token)

```
curl -i -X POST http://localhost:8001/services/attendance-service/plugins \
  --data 'name=rate-limiting' \
  --data 'config.minute=20' \
  --data 'config.limit_by=credential'
```

## 5.5. IP restriction (tùy chọn)

```
curl -i -X POST http://localhost:8001/services/attendance-service/plugins \
--data 'name=ip-restriction' --data
'config.whitelist=10.0.0.0/8,192.168.0.0/16'
```

## 6. Backend: Issue & Verify JWT (Node.js Example)

### 6.1. Cài đặt nhanh

```
npm i express jsonwebtoken bcryptjs uuid redis ioredis
```

### 6.2. Tạo access + refresh (ví dụ)

```
const jwt = require('jsonwebtoken');
const { v4: uuidv4 } = require('uuid');

const ACCESS_EXPIRES = 15 * 60; // 15 minutes
const REFRESH_EXPIRES = 30 * 24 * 3600; // 30 days

function signAccessToken(userId) {
  const payload = { sub: userId, iss: 'attendance-auth', aud: 'attendance-api' };
  return jwt.sign(payload, process.env.JWT_SECRET, { expiresIn: ACCESS_EXPIRES });
}

async function issueRefreshToken(userId, deviceId) {
  const token = uuidv4();
  const key = `refresh:${token}`;
  await redis.set(key, JSON.stringify({ userId, deviceId }), 'EX',
  REFRESH_EXPIRES);
  return token;
}
```

### 6.3. Refresh flow + rotate-on-use

```
app.post('/auth/refresh', async (req, res) =>{
  const { refreshToken, deviceId } = req.body;
  const data = await redis.get(`refresh:${refreshToken}`);
  if(!data) return res.status(401).send({error: 'INVALID_REFRESH'});

  const { userId, deviceId: stored } = JSON.parse(data);
  if(stored !== deviceId) return
```

```

res.status(403).send({error: 'INVALID_DEVICE'});

// rotate: revoke old token + issue new
await redis.del(`refresh:${refreshToken}`);
const newRefresh = await issueRefreshToken(userId, deviceId);
const access = signAccessToken(userId);
res.send({ accessToken: access, refreshToken: newRefresh, expiresIn: ACCESS_EXPIRES });
});

```

## 6.4. Logout (revoke refresh)

```

app.post('/auth/logout', async (req, res) => {
  const { refreshToken } = req.body;
  await redis.del(`refresh:${refreshToken}`);
  res.send({ ok: true });
});

```

---

## 7. Token Revocation & JWT Introspection Options

- **Short JWT expiry + refresh tokens** là giải pháp đơn giản, an toàn.
- Nếu cần revoke access token ngay lập tức:
  - 1. store revoked **jti** in Redis and check on each request via Kong custom plugin or via backend introspection; hoặc
  - 2. use opaque access tokens and Kong introspection (Enterprise) — Kong Enterprise hỗ trợ OAuth2 Introspection.

---

## 8. Test Cases & Kịch bản tấn công (và phòng)

1. **Access token expired** → Kong trả 401; client phải dùng refresh token để lấy token mới.
2. **Refresh token replayed** → rotate-on-use sẽ detect vì token đã bị xóa.
3. **Refresh token leaked + device mismatch** → block (store deviceld at issue time).
4. **High-frequency requests** → rate-limiting 429.
5. **Token manipulated (invalid signature)** → Kong 401.

---

## 9. Best Practices

- Không để JWT secret trong mã nguồn; dùng secret manager (HashiCorp Vault / AWS Secrets Manager).
- Log mọi hành vi auth: issue, refresh, revoke, failed attempts.
- Giới hạn scope minimal trong token.
- Dùng TLS everywhere (HTTPS + HSTS).

- Thường xuyên rotate signing keys và hỗ trợ key id (`kid`).
  - Giới hạn thời gian refresh token lifetime theo sensitivity (mặc định 30d có thể giảm với risk high).
- 

## 10. Giám sát & Alerting

- Metrics: `token_issued`, `token_refreshed`, `token_revoked`, `jwt_failures`, `rate_limit_hits`.
  - Alerts: spike trong `jwt_failures` (tấn công brute-force), nhiều refresh từ 1 token (replay), rate-limit tripped.
  - Dùng Prometheus exporter từ Kong + Grafana dashboards.
- 

## 11. Checklist Triển Khai (DevOps)

- Kong config quản lý bằng IaC (declarative YAML hoặc Terraform)
  - Backup Kong config & signing keys
  - Secret management cho JWT keys
  - Enable Kong Admin access control (RBAC)
  - Test chaos: expire keys, revoke refresh, simulate rate-limit
- 

## 12. Tóm tắt

Phiên bản "đơn giản nhưng chuyên sâu" này loại bỏ mọi xác thực bằng camera, OCR, hay face-match.

Thay vào đó tập trung vào:

- Cấu hình Kong để validate JWT và áp giới hạn truy cập
- Thiết kế access/refresh token an toàn: expiry ngắn, rotate-on-use, revocation
- Kiểm soát, logging, và giám sát để phát hiện abuse

Nếu bạn muốn, tôi sẽ:

- Sinh file `kong.yml` declarative sẵn sàng deploy
  - Viết sample backend full (Express) với auth + Redis
  - Viết minimal React Native client demo (login, refresh, checkin)
- Chọn 1 trong 3 và tôi sẽ tiếp tục ngay.