**Đề tài: API Gateway Security Service**

**1. Nghiên cứu và đánh giá thực trạng hiện nay**

**1.1 Thực trạng API trong hệ thống hiện nay**

* **API-first** đang là xu hướng trong phát triển ứng dụng web, mobile, SaaS, fintech…
* API thường được công khai (public hoặc partner API) → trở thành mục tiêu hàng đầu của tin tặc.
* Các API nếu không có cơ chế bảo mật chuẩn sẽ dễ bị:
  + **Credential stuffing / brute force**: thử mật khẩu hàng loạt, tấn công đăng nhập.
  + **Bot/automation abuse**: tạo tài khoản ảo, spam giao dịch.
  + **Injection / schema tampering**: gửi payload sai, gây lỗi logic hoặc crash service.
  + **Broken Object Level Authorization (BOLA)**: thay đổi ID trong request để truy cập dữ liệu của người khác.

**1.2 Số liệu và báo cáo thực tế**

* Theo báo cáo [**OWASP Top 10 API Security Risks – 2023 - OWASP API Security Top 10**](https://owasp.org/API-Security/editions/2023/en/0x11-t10/), trên **70% sự cố bảo mật web** đến từ API.
* Báo cáo [**Salt Security’s 2024 State of API Security Report**](https://salt.security/blog/increasing-api-traffic-proliferating-attack-activity-and-lack-of-maturity-key-findings-from-salt-securitys-2024-state-of-api-security-report) (2023) tổng hợp từ Salt Security năm 2024 cho thấy: **37% tổ chức gặp sự cố bảo mật API** trong 12 tháng qua, **chỉ 7.5% có chương trình kiểm thử/threat modeling chuyên biệt**. Thích hợp để chứng minh vấn đề thực tế đang ngày càng nghiêm trọng.
* Theo báo cáo của Kaspersky: [Việt Nam đứng đầu Đông Nam Á về số vụ tấn công 'vét cạn' - Báo VnExpress](https://vnexpress.net/viet-nam-dung-dau-dong-nam-a-ve-so-vu-tan-cong-vet-can-4887320.html), với **hơn 19,8 triệu vụ tấn công trong năm 2024** — chiếm **37% tổng số vụ trong khu vực**. Đây là con số rất đáng chú ý, minh chứng cho tần suất và mức độ nghiêm trọng của các cuộc tấn công brute-force hướng tới hệ thống đăng nhập (có thể là API login, login endpoint của các e-commerce app, v.v.) tại Việt Nam.

**1.3 Đánh giá hạn chế hiện tại**

* **Backend service** thường chỉ có **xác thực đơn giản** (username/password), không chuẩn hóa JWT/OAuth2.
* **Không có rate limiting**: API login/signup dễ bị spam.
* **Payload validation sơ sài**: ít khi kiểm tra body/params theo schema chuẩn.
* **Thiếu logging tập trung**: khó phát hiện tấn công, khó điều tra.

**=> Thực trạng chung: API là xương sống hệ thống, nhưng lớp bảo vệ hiện nay chưa đủ, dễ bị abuse.**

**2. Giải pháp đề xuất: API Gateway Security Service**

**2.1 Mục tiêu**

* Tạo một lớp **bảo vệ trung tâm** cho tất cả API.
* Giải quyết các vấn đề:
  1. **Xác thực & ủy quyền chuẩn hóa** bằng JWT/OIDC.
  2. **Chống abuse/bot** bằng rate limiting & throttling.
  3. **Ngăn injection/sai lệch** bằng OpenAPI schema validation.
  4. **Audit tập trung**: log toàn bộ request/response quan trọng để dễ giám sát.

**2.2 Cách tiếp cận**

* Tích hợp **API Gateway** làm lớp trung gian giữa client và service backend.
* Gateway chịu trách nhiệm:
  + Verify JWT (qua JWKS/IdP).
  + Thực thi rate limiting trên route nhạy cảm (login, signup, OTP).
  + Validate request theo file OpenAPI spec.
  + Xuất log/audit ra hệ thống giám sát (ELK, Grafana).

**2.3 Lợi ích chính**

* **Giảm rủi ro API abuse**: bot, brute force, credential stuffing.
* **Chuẩn hóa bảo mật**: các microservice không cần viết lại logic bảo mật.
* **Cải thiện tuân thủ**: phù hợp ISO 27001, SOC 2, PCI DSS.
* **Dễ demo và thực thi**: có thể triển khai bằng Kong/NGINX + Keycloak trong Docker.

**3. Công nghệ đề xuất**

**3.1 Thành phần chính**

1. **API Gateway**:
   * **Kong Gateway** (mã nguồn mở, hỗ trợ plugin bảo mật phong phú).
   * Hoặc **NGINX + Lua plugin** (nếu muốn nhẹ hơn).
2. **Identity Provider (IdP)**:
   * **Keycloak** (mã nguồn mở, hỗ trợ OIDC/OAuth2/SAML).
   * Có thể thay thế bằng Auth0/Okta nếu dùng dịch vụ cloud.
3. **Backend Service mẫu**:
   * **Node.js (NestJS)** – dễ viết demo, có route login & user.
   * Hoặc Java Spring Boot nếu muốn gắn với hệ thống lớn.
4. **Logging & Monitoring**:
   * **Fluent Bit → Elasticsearch/Kibana** để quan sát request/response.
   * Hoặc Grafana Loki/Prometheus nếu quen DevOps.

**3.2 Mô hình kiến trúc**

* Client (Web/Mobile) → API Gateway → IdP (Keycloak) → Backend service (NestJS) → Log/Audit (Kibana).

**3.3 Các tính năng bảo mật cụ thể**

* **JWT/OIDC verify**: xác minh token từ Keycloak.
* **Rate limiting**: giới hạn số request/ phút trên IP hoặc user.
* **Schema validation**: so khớp payload với OpenAPI spec.
* **Audit log**: lưu user, route, quyết định, latency.

**3.4 Công cụ hỗ trợ demo**

* **Docker Compose** để dựng môi trường.
* **Postman/Insomnia** để test API.
* **Mermaid diagram** để vẽ sơ đồ trong báo cáo.

**4. Kết luận phần nghiên cứu và giải pháp**

* **Thực trạng**: API ngày càng phổ biến nhưng bị lạm dụng nhiều, thiếu lớp bảo vệ chuẩn.
* **Giải pháp**: Xây dựng **API Gateway Security Service** – lớp bảo mật trung tâm, ngăn abuse, chuẩn hóa xác thực, enforce schema, logging audit.
* **Công nghệ**: Dùng **Kong + Keycloak + NestJS + Fluent Bit/ELK**, phù hợp để demo và triển khai thực tế.

**PHỤ LỤC: GIẢI THÍCH MỘT SỐ CÁC THUẬT NGỮ CHUYÊN NGÀNH**

**1. API-first**

Phương pháp phát triển ứng dụng ưu tiên thiết kế API trước khi xây dựng giao diện người dùng (web, mobile). Giúp hệ thống dễ mở rộng và tích hợp.

**2. Public API / Partner API**

* **Public API**: API công khai, bất kỳ ai cũng có thể truy cập.
* **Partner API**: API dành riêng cho đối tác đã được cấp quyền.

**3. Credential stuffing / Brute force**

* **Credential stuffing**: Tấn công bằng cách sử dụng danh sách tài khoản/mật khẩu bị rò rỉ để thử đăng nhập vào hệ thống.
* **Brute force**: Thử nhiều mật khẩu ngẫu nhiên cho tới khi tìm được đúng mật khẩu.

**4. Bot/automation abuse**

Sử dụng phần mềm tự động (bot) để spam, tạo tài khoản ảo hoặc thực hiện các hành vi gian lận qua API.

**5. Injection / Schema tampering**

* **Injection**: Chèn dữ liệu độc hại vào API (ví dụ: SQL Injection) nhằm khai thác lỗ hổng bảo mật.
* **Schema tampering**: Gửi dữ liệu sai định dạng, thay đổi cấu trúc dữ liệu nhằm gây lỗi cho dịch vụ.

**6. Broken Object Level Authorization (BOLA)**

Lỗi phân quyền cho phép người dùng truy cập dữ liệu/tài nguyên của người khác chỉ bằng cách thay đổi ID trong request.

**7. JWT / OAuth2 / OIDC**

* **JWT (JSON Web Token)**: Chuẩn token chứa thông tin xác thực, trao đổi giữa client và server.
* **OAuth2**: Giao thức xác thực/ủy quyền hiện đại, hỗ trợ “đăng nhập bằng Google, Facebook”…
* **OIDC (OpenID Connect)**: Lớp mở rộng của OAuth2, hỗ trợ xác thực người dùng mạnh mẽ hơn.

**8. Rate limiting / Throttling**

* **Rate limiting**: Giới hạn số lượng request tới API trong một khoảng thời gian (ví dụ: 10 lần/phút).
* **Throttling**: Làm chậm hoặc từ chối các request vượt quá giới hạn.

**9. OpenAPI schema validation**

Kiểm tra dữ liệu gửi lên API có đúng định dạng, kiểu dữ liệu và các trường thông tin quy định dựa trên file mô tả OpenAPI.

**10. Audit log / Centralized logging**

* **Audit log**: Nhật ký ghi lại các hoạt động quan trọng trên hệ thống API.
* **Centralized logging**: Tập trung tất cả log về một nơi để giám sát, điều tra.

**11. API Gateway**

Thành phần trung gian giữa client và backend, giúp kiểm soát, bảo vệ, chuyển tiếp các request đến service phía sau.

**12. Identity Provider (IdP)**

Hệ thống cung cấp dịch vụ xác thực/ủy quyền cho người dùng (ví dụ: Keycloak, Auth0, Okta).

**13. Kong Gateway / NGINX + Lua plugin**

* **Kong Gateway**: API Gateway mã nguồn mở, nhiều tính năng bảo mật.
* **NGINX + Lua plugin**: Sử dụng NGINX làm Gateway, mở rộng tính năng bằng script Lua.

**14. Fluent Bit, ELK, Grafana, Prometheus, Loki**

* **Fluent Bit**: Công cụ thu thập log nhẹ.
* **ELK**: Bộ công cụ Elasticsearch, Logstash, Kibana dùng cho lưu trữ, phân tích, trực quan hóa log.
* **Grafana, Prometheus, Loki**: Bộ công cụ giám sát, phân tích log và chỉ số trong DevOps.

**15. Docker Compose**

Công cụ giúp dựng nhanh nhiều dịch vụ (API Gateway, Keycloak, backend, log…) chỉ với một file cấu hình duy nhất.

**16. Postman / Insomnia**

Phần mềm hỗ trợ kiểm thử API (gửi request, kiểm tra response).

**17. Mermaid diagram**

Công cụ vẽ sơ đồ (flowchart, sequence diagram…) bằng mã code đơn giản.

**18. Enforce schema**

Áp đặt, kiểm soát chặt chẽ định dạng dữ liệu gửi lên API để tránh lỗi, lỗ hổng.

**19. Compliance: ISO 27001, SOC 2, PCI DSS**

Các tiêu chuẩn/bộ quy tắc về an toàn thông tin, bảo mật dữ liệu mà doanh nghiệp cần tuân thủ.