Project: Service Mesh & DevSecOps

Liên kết mã nguồn
https://github.com/spring-petclinic/spring-petclinic-microservices

Mục tiêu
- Thực hành cấu hình Service Mesh (mTLS, chính sách kết nối) trên K8S cho ứng dụng microservices.

- Triển khai các công cụ DevSecOps trong CI/CD để quét mã, dependency và thực hiện DAST/secret scanning.

- Viết tài liệu hướng dẫn, test plan, và demo khả năng bảo mật & vận hành.

Yêu cầu môi trường
- K8S cluster.

- kubectl, helm, istioctl (nếu dùng Istio), hoặc lựa chọn service-mesh khác.

- CI/CD (Jenkins), SonarQube server, Snyk account/CLI, ZAP (OWASP ZAP).

- Truy cập internet để pull image và cài add-on.

Phần 1 — Service Mesh
Tóm tắt nhiệm vụ:

1. Enable TLS (mTLS) giữa các service deploy trên K8S cho ứng dụng spring-petclinic-microservices.

2. Vẽ flow chart/Topology của các service (sử dụng Kiali để quan sát).

3. Chuẩn bị kịch bản test:

- retryable: nếu service trả lỗi 500 thì retry tự động (định nghĩa retry policy trong service mesh).

- Setup policy: chỉ những service server nào được phép giao tiếp với nhau mới connect được (authorization policy).

- Test: vào pod khác trong cluster, thực hiện curl tới service để kiểm tra xem policy cho phép hay chặn kết nối.

Gợi ý triển khai (Service Mesh)
- Option phổ biến: Istio (cài trên K8S) + Kiali để visualize.

- Bật mTLS toàn mesh hoặc cho từng namespace bằng PeerAuthentication/DestinationRule.

- Dùng AuthorizationPolicy / RequestAuthentication (Istio) để giới hạn service-to-service access.

- Cấu hình retry bằng VirtualService (policy retry, timeout).

- Lệnh kiểm tra mẫu: kubectl exec -n <ns> <pod> -- curl -v http://<service>.<ns>:<port>/

Deliverables (Service Mesh)
- YAML manifest cấu hình mTLS và authorization policy.

- Screenshot Kiali topology và giải thích flow.

- Test plan + logs (kết quả curl, retry evidence).

- README hướng dẫn cách triển khai từng bước.

Phần 2 — DevSecOps
Nhiệm vụ:

1. Triển khai SonarQube trong pipeline CI/CD để quét code (SAST).

2. Sử dụng Snyk để quét dependency (vulnerabilities & license issues).

3. Thực hiện DAST bằng OWASP ZAP (trong pipeline hoặc standalone) để quét ứng dụng đang chạy.

4. Thêm Git hook (pre-commit hoặc server-side) để chạy gitleaks khi commit nhằm phát hiện secret leak.

Gợi ý triển khai (DevSecOps)
- SonarQube: chạy scanner trong CI (maven/gradle scanner) và fail build nếu quality gate không đạt.

- Snyk: kết hợp Snyk CLI hoặc Snyk GitHub/GitLab integration để quét dependency tự động.

- ZAP: dùng ZAP baseline/active scan trong CI để quét endpoints của ứng dụng đã deploy; lưu report HTML/JSON.

- Git hooks: sử dụng pre-commit framework hoặc server-side hooks + gitleaks; đồng thời có policy reject push nếu detect secret.

Deliverables (DevSecOps)
- CI pipeline (Jenkins) có bước SonarQube, Snyk, ZAP.

- Ví dụ cấu hình SonarQube properties và script gọi scanner.

- Report Snyk & ZAP (HTML/JSON) và log.

- Hook script gitleaks + hướng dẫn cài đặt.

Test Plan & Acceptance Criteria

- Service Mesh:

* mTLS: không thể kết nối plaintext giữa sidecar-enabled pod; kết nối thành công khi TLS được bật.

* Authorization: chỉ service A -> B khi policy cho phép; các request khác bị từ chối (HTTP 403/connection refused).

* Retry: khi backend trả 500, gateway hoặc client sidecar tự retry theo policy; cung cấp logs chứng minh.

- DevSecOps:

* Sonar: quality gate phải PASS hoặc build sẽ fail.

* Snyk: báo cáo các dependency có lỗ hổng, ít nhất demo một fix.

* ZAP: không có high-severity open vulns (một số medium có thể chấp nhận tuỳ rubric).

* Gitleaks hook: hook phát hiện secret mẫu và chặn commit (show demo).