Đại học Quốc gia thành phố Hồ Chí Minh
Trường Đại học Công nghệ Thông tin
Khoa mạng máy tính và truyền thông
Bộ môn An toàn thông tin

4

Lab

Security with Snyk in DevSecOps

Thực hành môn An toàn mạng máy tính nâng cao

Tháng 3/2024

Lưu hành nội bộ

A. TỔNG QUAN

1. Mục tiêu

- Tìm hiểu sử dụng công cụ bảo mật Snyk trong DevSecOps để tăng cường bảo mật cho các bước trong quy trình phát triển và phân phối phần mềm

2. Thời gian thực hành

- Thực hành tại lớp: 5 tiết tại phòng thực hành.
- Hoàn thành báo cáo kết quả thực hành: tối đa 13 ngày.

B. CHUẨN BỊ MÔI TRƯỜNG

- Máy tính kết nối Internet
- Phần mềm: Visual Studio Code, Snyk, Git

C. THỰC HÀNH

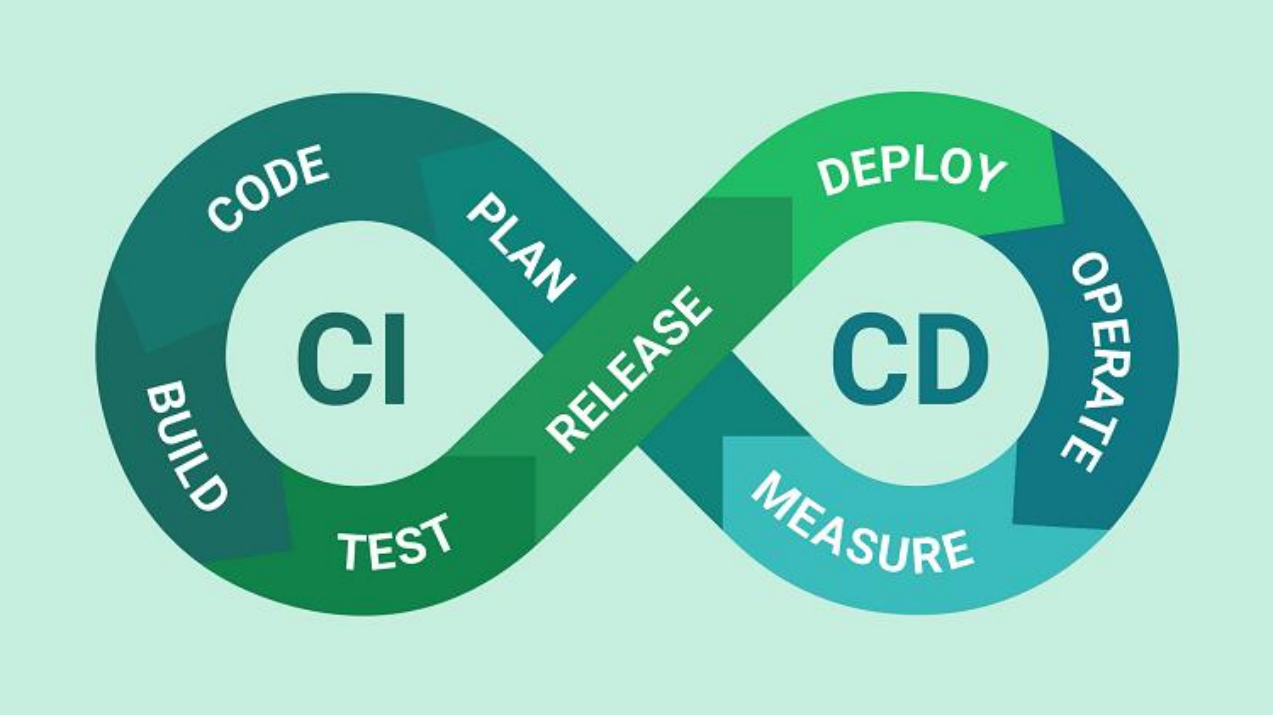
1. Giới thiệu về Snyk:

Snyk là bộ công cụ bảo mật cung cấp các giải pháp tìm và sửa các lỗ hổng trong phần mềm. Snyk kiểm tra các lỗ hổng trong mã nguồn, các phần phụ thuộc, image container, cơ sở hạ tầng dưới dạng cấu hình mã và môi trường đám mây, đồng thời đưa ra ngữ cảnh, mức độ ưu tiên và biện pháp khắc phục.
Snyk hỗ trợ nhiều ngôn ngữ khác nhau như JavaScript, Java (Gradle, Maven), .NET, Python, Golang, Swift, Objective-C (CocoaPods), Scala, Ruby, PHP, Bazel, Terraform, CloudFormation, Azure Resource Manager, Kubernetes, and Dockerfiles.
Trong bài thực hành này, chúng ta sẽ sử dụng bộ công cụ Snyk để thực hiện đánh giá bảo mật trong từng giai đoạn phát triển sản phẩm.
Để bắt đầu bài thực hành, chúng ta cần đăng ký một tài khoản Snyk (https://snyk.io/) và GitHub (https://github.com/) để lưu trữ nội dung mã nguồn.
Task: Tạo tài khoản Snyk và GitHub

2. Đánh giá bảo mật với Snyk

Snyk bao gồm nhiều công cụ khác nhau hỗ trợ cho từng giai đoạn trong quá trình phát triển và triển khai phần mềm:

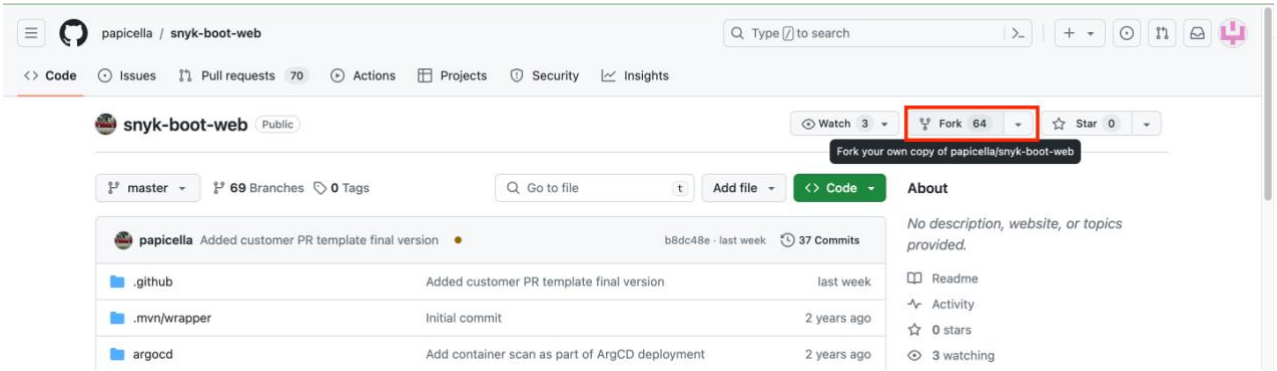
- Snyk Code (SAST), Snyk Open Source(SCA): kiểm tra code và các gói 3rd-party open source
- Snyk Container: kiểm tra cấu hình file image và lỗ hổng trên nền tảng Linux
- Snyk Infrastructure as Code: cung cấp đánh giá cho các cấu hình cở sở hạ tầng đám mây.



Question: Dựa vào thông tin về các công cụ của Snyk, hãy dự đoán các công cụ này của Snyk hỗ trợ kiểm tra, đánh giá và khắc phục các vấn đề bảo mật ở những giai đoạn nào trong quá trình phát triển phần mềm?

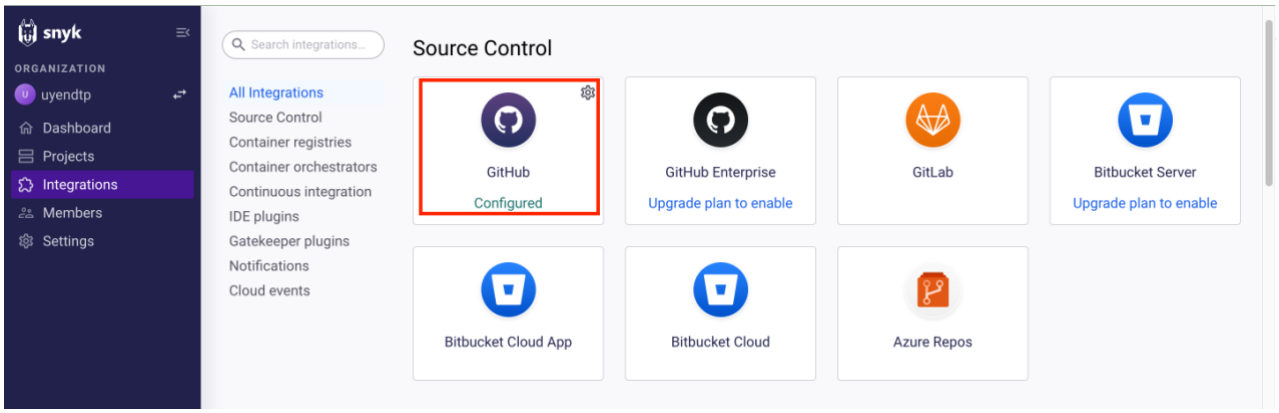
a. Fork sample webapp vào GitHub repository

- Truy cập đường dẫn của GitHub repository sau và chọn **Fork**: https://github.com/papicella/snyk-boot-web



b. Cấu hình GitHub Intergration

- Đăng nhập vàp http://app.snyk.io
- Tại trang chủ, chọn **Intergrations ➔ Source Control ➔ GitHub**

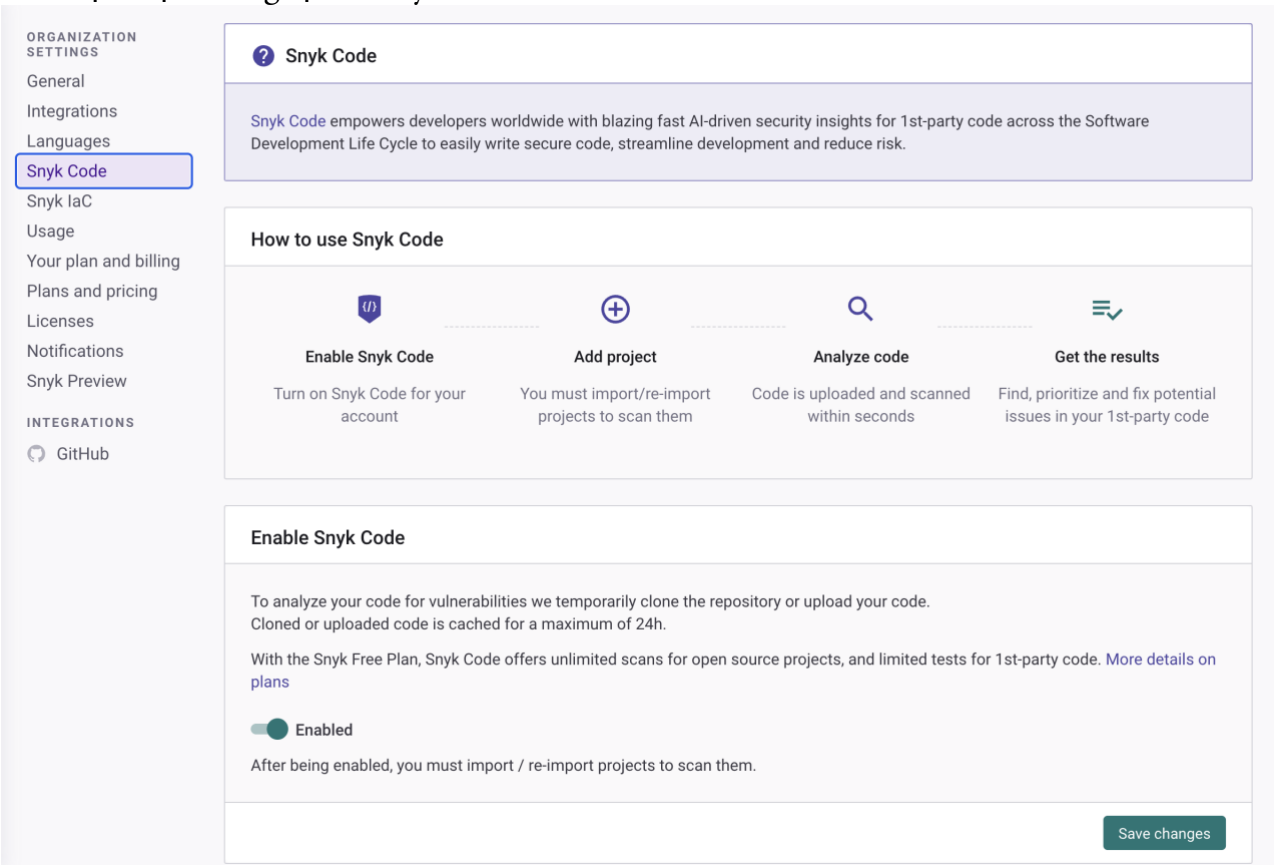


- Điền các thông tin để kết nối GitHub và Snyk

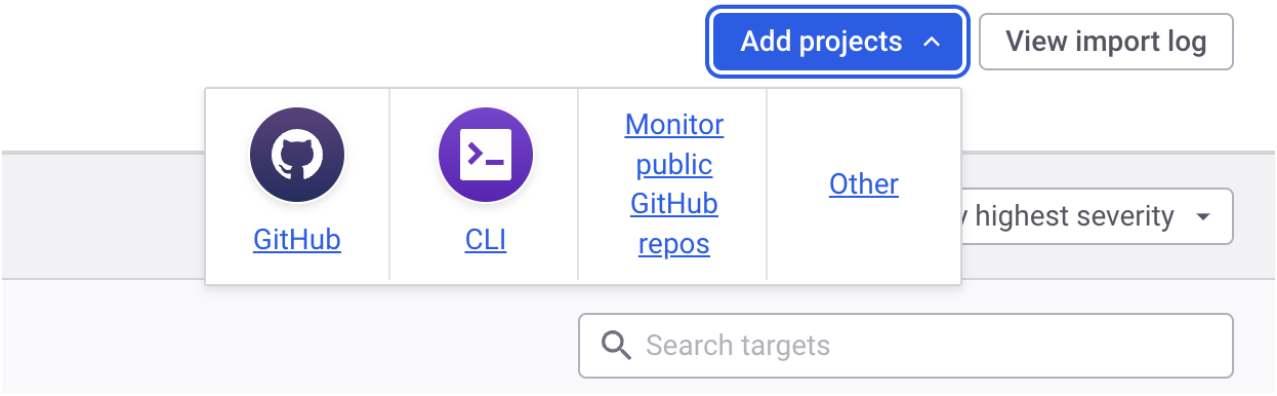
c. Import Repository và enable Snyk Code

Synk đã được kết nối với GitHub account của bạn. Tiếp theo, chúng ta sẽ tiến hành enable Snyk Code và import Repository vào Snyk.

- Kiểm tra Synk Code đã được enable chưa bằng cách truy cập vào **Settings ➔ Snyk Code**. Tiến hành **Enable** và lưu các thay đổi
- Thực hiện tương tự với Snyk IaC



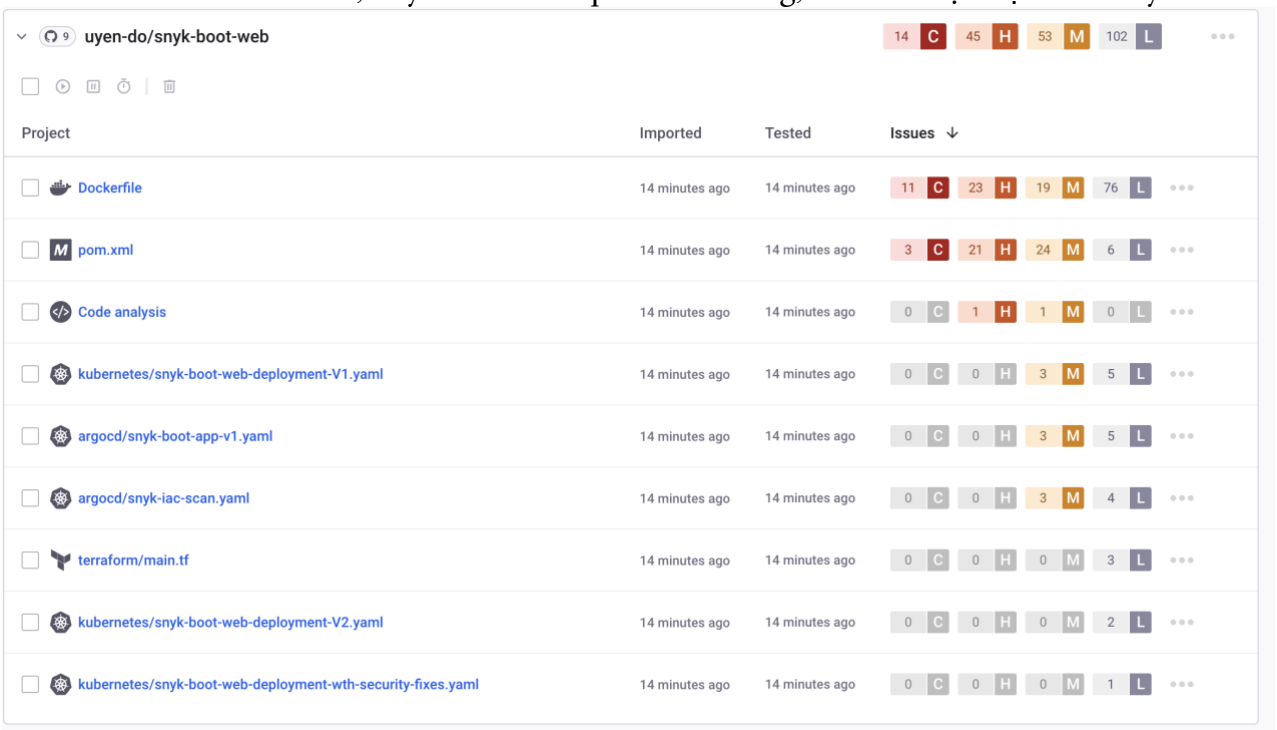
- Chọn **Project ➔ Add project ➔ GitHub**
- Chọn Repo đã được fork ở bước trước và chọn **Add selected repositories**



- Có thể mất vài phút để code được import và scan bởi Snyk

d. Phân tích kết quả của Snyk

- Sau khi tiến hành scan, Snyk trả về kết quả các lỗ hổng, mối đe doạ được tìm thấy



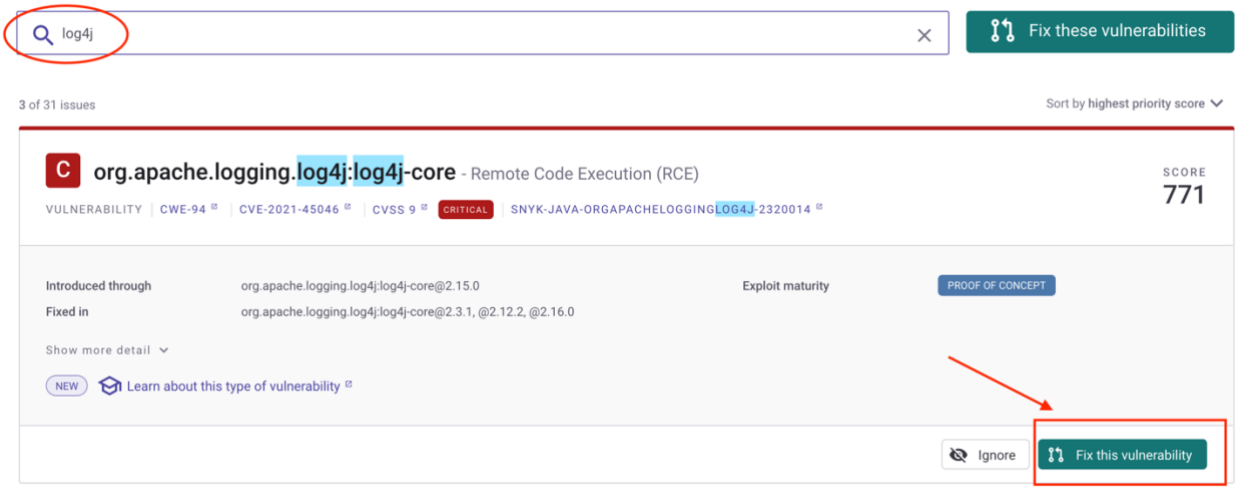
Task: Quan sát và phân tích kết quả của việc scan trên các môi trường khác nhau: code application, container, IaC.

e. Fix các lỗ hổng bảo mật bằng tính năng Snyk Pull Request

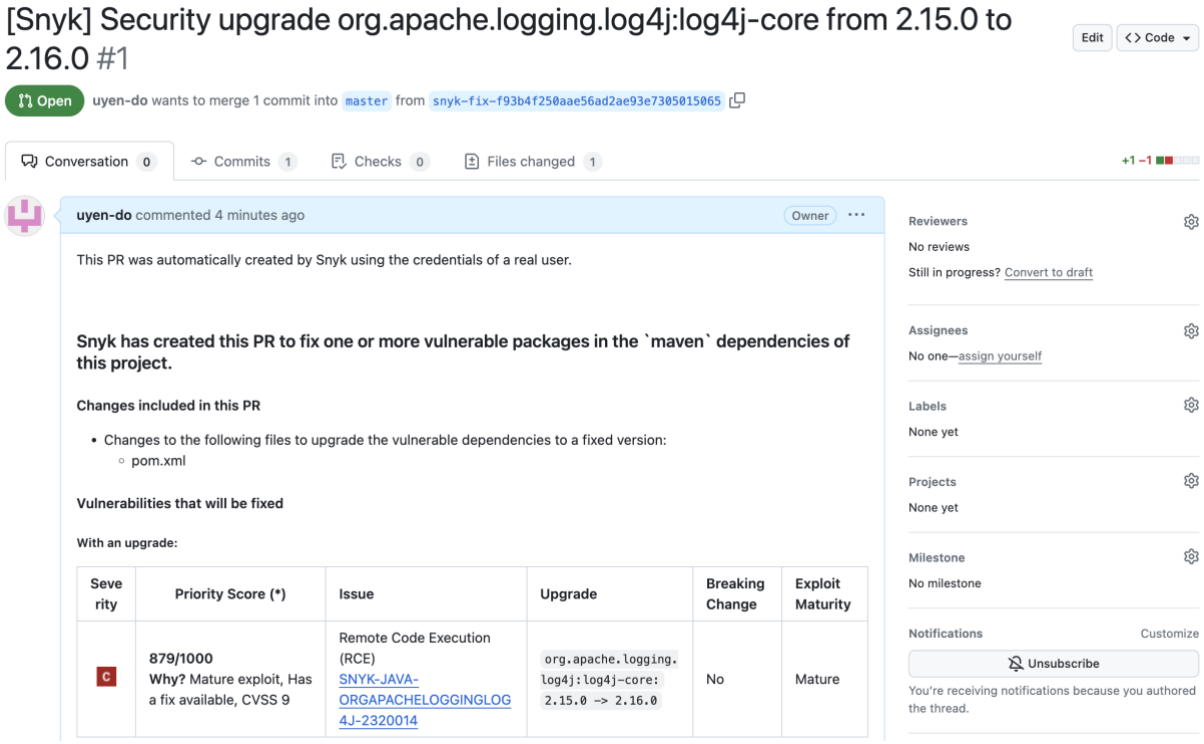
Ngoài việc cung cấp các lỗ hổng bảo mật được tìm thấy trong mã nguồn, Snyk Code còn cung cấp những gợi ý để khắc phục các lỗ hổng này.

Task: Dùng tính năng Snyk Pull Request để fix các lỗ hổng được tìm thấy

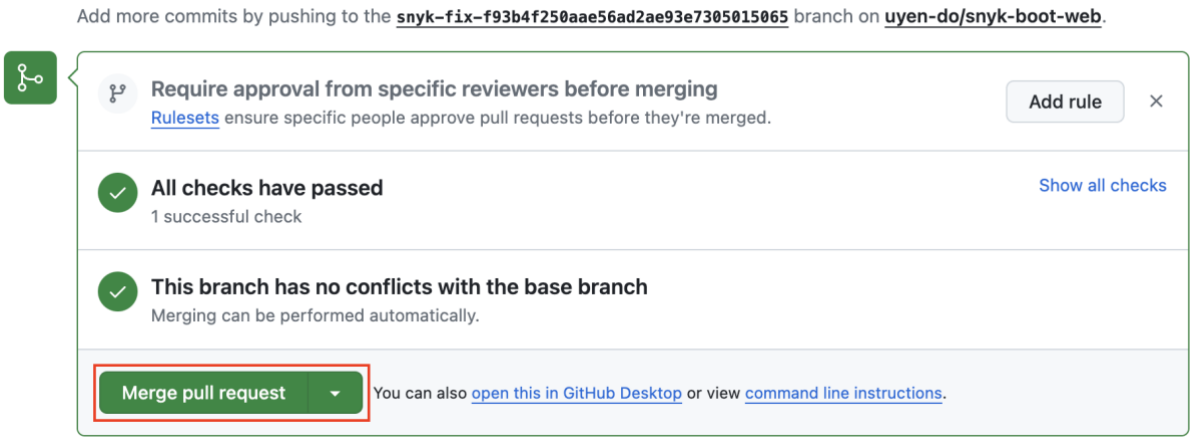
- Mở file **pom.xml** để quan sát lại các lỗ hổng bảo mật đã được tìm thấy.
- Chọn một lỗ hổng bảo mật và chọn **Fix this vulnerability**



- Chọn những lỗ hổng cần khắc phục và chọn **Open PR Fix** để tạo một pull request mới



- Lúc này, một pull request mới đã được tạo, chúng ta có thể chọn các tab Conversation, Commits, Checks, Files changed để xem thông tin chi tiết về Pull Request này.
- Sau khi kiểm tra và xác nhận không có xung đột gì, tiến hành merge pull request



- Quay lại Snyk, kiểm tra và thấy rằng số lượng cảnh báo trên tập tin **pom.xml** đã giảm.

3. Snyk CLI & Snyk IDE

Bên cạnh giao diện dễ sử dụng, Snyk cũng cung cấp một chế độ dòng lệnh (CLI) và plugin IDE linh hoạt và mạnh mẽ. Chế độ CLI và IDE thường được sử dụng trong quy trình DevOps để tự động hóa việc quét các lỗ hổng bảo mật trong mã nguồn, cũng như hạ tầng cơ sở hạ tầng, một cách linh hoạt và hiệu quả. Điều này giúp tích hợp Snyk vào các quy trình tự động hóa và công cụ hiện có của tổ chức một cách dễ dàng, giúp cải thiện quy trình phát triển và triển khai phần mềm một cách an toàn và đáng tin cậy.

a. Snyk CLI

- Cài đặt Snyk CLI theo hướng dẫn sau: https://docs.snyk.io/snyk-cli/install-or-update-the-snyk-cli
- Uỷ quyền cho Snyk CLI bằng cách chạy câu lệnh sau ở Terminal/CMD.

```
snyk auth
```

- Cài đặt Git theo hướng dẫn: https://git-scm.com/downloads
- Clone nội dung Webapp về máy

```
git clone https://github.com/papicella/snyk-boot-web
cd snyk-boot-web
```

- Sử dụng Synk Open Source để scan manifest file

```
snyk test
```

```
(base) uyendo@192 snyk-boot-web % snyk test

Testing /Users/uyendo/Downloads/snyk-boot-web...

Tested 40 dependencies for known issues, found 54 issues, 54 vulnerable paths.


Issues to fix by upgrading:

  Upgrade com.h2database:h2@1.4.200 to com.h2database:h2@2.2.220 to fix
  ✗ Information Exposure [Medium Severity][https://security.snyk.io/vuln/SNYK-JAVA-COMH2DATABASE-3146851] in com.h2database:h2@1.4.200
    introduced by com.h2database:h2@1.4.200
  ✗ Remote Code Execution (RCE) [High Severity][https://security.snyk.io/vuln/SNYK-JAVA-COMH2DATABASE-2331071] in com.h2database:h2@1.4.200
    introduced by com.h2database:h2@1.4.200
  ✗ XML External Entity (XXE) Injection [High Severity][https://security.snyk.io/vuln/SNYK-JAVA-COMH2DATABASE-1769238] in com.h2database:h2@1.4.200
    introduced by com.h2database:h2@1.4.200
  ✗ Remote Code Execution (RCE) [Critical Severity][https://security.snyk.io/vuln/SNYK-JAVA-COMH2DATABASE-2348247] in com.h2database:h2@1.4.200
    introduced by com.h2database:h2@1.4.200

  Upgrade org.apache.logging.log4j:log4j-core@2.15.0 to org.apache.logging.log4j:log4j-core@2.17.1 to fix
  ✗ Arbitrary Code Execution [Medium Severity][https://security.snyk.io/vuln/SNYK-JAVA-ORGAPACHELOGGINGLOG4J-2327339] in org.apache.logging.log4j:log4j-core@2.15.0
    introduced by org.apache.logging.log4j:log4j-core@2.15.0
  ✗ Denial of Service (DoS) [High Severity][https://security.snyk.io/vuln/SNYK-JAVA-ORGAPACHELOGGINGLOG4J-2321524] in org.apache.logging.log4j:log4j-core@2.15.0
    introduced by org.apache.logging.log4j:log4j-core@2.15.0
  ✗ Remote Code Execution (RCE) [Critical Severity][https://security.snyk.io/vuln/SNYK-JAVA-ORGAPACHELOGGINGLOG4J-2320014] in org.apache.logging.log4j:log4j-core@2.15.0
    introduced by org.apache.logging.log4j:log4j-core@2.15.0

  Upgrade org.springframework.boot:spring-boot-actuator@2.3.10.RELEASE to org.springframework.boot:spring-boot-actuator@2.7.18 to fix
  ✗ Improper Handling of Case Sensitivity [Low Severity][https://security.snyk.io/vuln/SNYK-JAVA-ORGSPRINGFRAMEWORK-2689634] in org.springframework:spring-context@5.2.14.RELEASE
    introduced by org.springframework.boot:spring-boot-actuator@2.3.10.RELEASE > org.springframework.boot:spring-boot@2.3.10.RELEASE > org.springframework:spring-context@5.2.14.RELEASE
  ✗ Denial of Service (DoS) [Medium Severity][https://security.snyk.io/vuln/SNYK-JAVA-ORGSPRINGFRAMEWORKBOOT-6226862] in org.springframework.boot:spring-boot-actuator@2.3.10.RELEASE
    introduced by org.springframework.boot:spring-boot-actuator@2.3.10.RELEASE
  ✗ Improper Input Validation [Medium Severity][https://security.snyk.io/vuln/SNYK-JAVA-ORGSPRINGFRAMEWORK-2330878] in org.springframework:spring-core@5.2.14.RELEASE
    introduced by org.springframework.boot:spring-boot-actuator@2.3.10.RELEASE > org.springframework.boot:spring-boot@2.3.10.RELEASE > org.springframework:spring-core@5.2.14.RELEASE
  ✗ Improper Output Neutralization for Logs [Medium Severity][https://security.snyk.io/vuln/SNYK-JAVA-ORGSPRINGFRAMEWORK-2329097] in org.springframework:spring-core@5.2.14.RELEASE
    introduced by org.springframework.boot:spring-boot-actuator@2.3.10.RELEASE > org.springframework.boot:spring-boot@2.3.10.RELEASE > org.springframework:spring-core@5.2.14.RELEASE
```

- Sử dụng Synk Code để scan source code

```
snyk code test
```

- Xuất kết quả thành file HTML. Để xuất được kết quả thành file HTML, cần cài đặt một plugin snyk-to-html (https://docs.snyk.io/snyk-cli/scan-and-maintain-projects-using-the-cli/cli-tools/snyk-to-html)

```
snyk test --json | snyk-to-html -o results.html
```

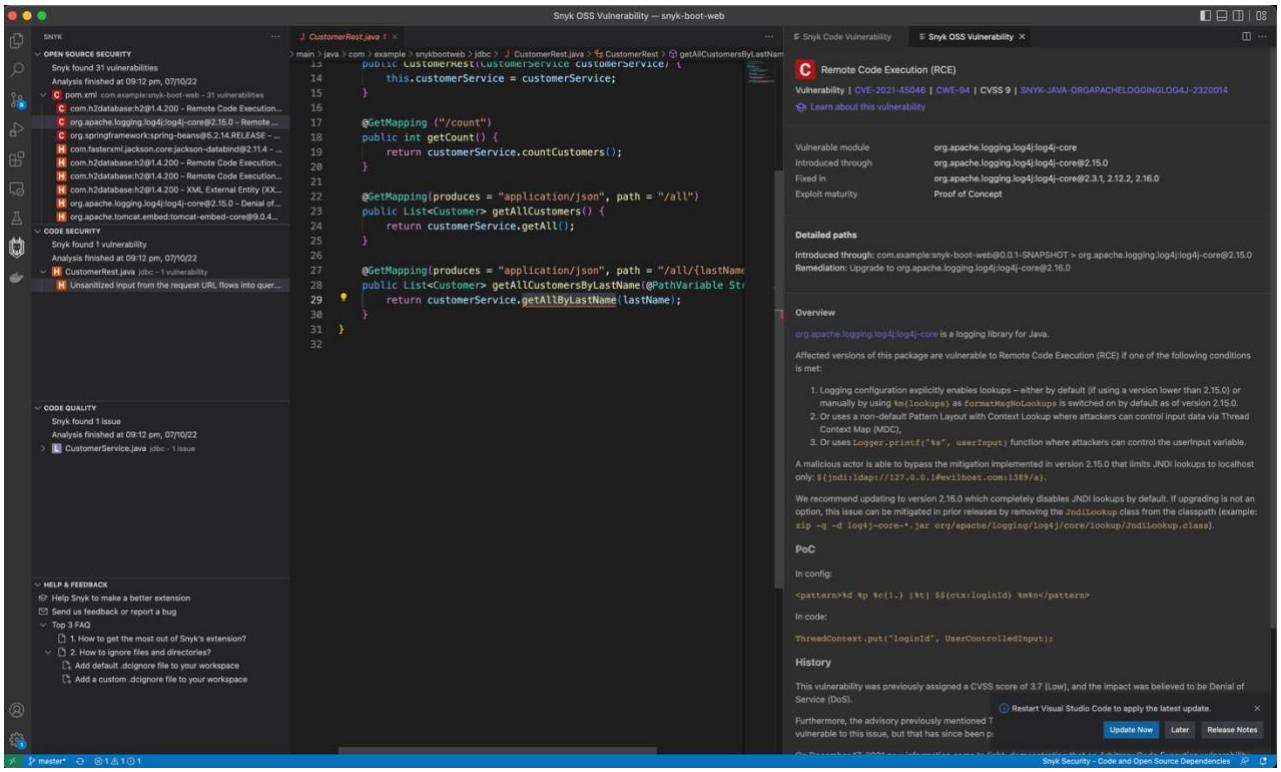
Task: Cài đặt Snyk CLI, sử dụng các công cụ của Snyk để scan và xuất report thành file HTML

b. Snyk IDE

Snyk IDE hỗ trợ nhiều IDE khác nhau như Eclipse, JetBrains, Visual Studio, Visual Studio Code

Task: Cài đặt Snyk plugin/extension vào IDE đang sử dụng và quan sát kết quả scan

- Sử dụng các hướng dẫn sau để cài đặt Snyk IDE: https://docs.snyk.io/integrate-with-snyk/use-snyk-in-your-ide



c. Pre-commit hook với Synk CLI

Git hook là một tính năng mạnh mẽ của Git cho phép tùy chỉnh và tự động hóa quy trình làm việc với repository của mình. Khi được kích hoạt bởi các sự kiện nhất định như commit, push, hoặc merg, Git hook sẽ chạy các script được xác định trước để thực hiện các hành động tùy chỉnh. Điều này cho phép thực hiện các công việc như kiểm tra mã nguồn, kiểm soát chất lượng mã nguồn, tự động hóa quy trình triển khai.

Git hook pre-commit là một trong những hook phổ biến nhất trong Git, được sử dụng để thực hiện các hành động trước khi một commit được tạo. Trong trường hợp này, chúng ta có thể kết hợp pre-commit hook và Snyk CLI để tự động kiểm tra mã nguồn của chúng ta trước khi commit để đảm bảo rằng không có lỗ hổng bảo mật nào trong mã nguồn của chúng ta.

Bonus: Tạo một pre-commit hook gọi Snyk CLI để scan repository

- Tại thư mục repository, truy cập thư mục hook

```
cd .git/hooks
```

- Tại đây có nhiều file hook sample để chúng ta tham khảo như pre-commit.sample, pre-push.sample
- Tạo một file **pre-commit.sh** và viết script gọi tới Snyk CLI để tiến hành scan repository trước khi commit.
- Commit và quan sát kết quả.

D. YÊU CẦU & ĐÁNH GIÁ

- Sinh viên tìm hiểu và thực hành theo hướng dẫn, thực hiện theo nhóm đã đăng ký.

- Nộp báo cáo kết quả gồm chi tiết những việc (Report) mà nhóm đã thực hiện, quan sát thấy và kèm ảnh chụp màn hình kết quả (nếu có); giải thích cho quan sát (nếu có).
- Báo cáo:
  - File .PDF. Tập trung vào nội dung, không mô tả lý thuyết.
  - Đặt tên theo định dạng: [Mã lớp]-LabX_MSSV1_MSSV2.
  - Ví dụ: [NT534.K11.ANTN.1]-Lab1_1852xxxx_1852yyyy.
  - Nếu báo cáo có nhiều file, nén tất cả file vào file .ZIP với cùng tên file báo cáo.
  - Nộp file báo cáo trên theo thời gian đã thống nhất tại courses.uit.edu.vn.

Bài sao chép, trễ, … sẽ được xử lý tùy mức độ vi phạm.

HẾT

Chúc các bạn hoàn thành tốt!