ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG

CHÂU KIM LỘC

TRẦN QUANG KHA

KHÓA LUẬN TỐT NGHIỆP

TÍCH HỢP BỘ LƯU TRỮ TỰ ĐỘNG CÁC LỖ HỔNG BẢO MẬT CÙNG VỚI CÁC BẢN SỬA LỖI VÀO QUI TRÌNH ĐÁNH GIÁ BẢO MẬT LIÊN TỤC CHO CÁC ỨNG DỤNG WEB

INTEGRATING AUTOMATED COLLECTION OF VULNERABILITIES AND THEIR FIXES WITH CONTINUOUS SECURITY ASSESSMENT FOR WEB APPLICATIONS

KỸ SƯ NGÀNH AN TOÀN THÔNG TIN

TP. HỒ CHÍ MINH, 2022

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG

CHÂU KIM LỘC – 18521003

TRẦN QUANG KHA – 18520875

KHÓA LUẬN TỐT NGHIỆP

TÍCH HỢP BỘ LƯU TRỮ TỰ ĐỘNG CÁC LỖ HỔNG BẢO MẬT CÙNG VỚI CÁC BẢN SỬA LỖI VÀO QUI TRÌNH ĐÁNH GIÁ BẢO MẬT LIÊN TỤC CHO CÁC ỨNG DỤNG WEB

INTEGRATING AUTOMATED COLLECTION OF VULNERABILITIES AND THEIR FIXES WITH CONTINUOUS SECURITY ASSESSMENT FOR WEB APPLICATIONS

KỸ SƯ NGÀNH AN TOÀN THÔNG TIN

GIẢNG VIÊN HƯỚNG DẪN

TS. NGUYỄN TẤN CẦM

THS. ĐỖ HOÀNG HIỂN

TP. HỒ CHÍ MINH, 2022

THÔNG TIN HỘI ĐỒNG CHẤM KHÓA LUẬN TỐT NGHIỆP

Hội đồng chấm khóa luận tốt nghiệp, thành lập theo Quyết định số …… ngày …….. của Hiệu trưởng Trường Đại học Công nghệ Thông tin.

LỜI CẢM ƠN

Chúng tôi xin gửi lời tri ân và biết ơn đến TS. Nguyễn Tấn Cầm, ThS. Đỗ Hoàng Hiển đã trực tiếp quan tâm, hướng dẫn tận tình trong suốt quá trình thực hiện đề tài. Xin đặc biệt gửi lời cảm ơn trân trọng nhất đến ThS. Phan Thế Duy đã định hướng, dẫn dắt và đồng hành cùng chúng tôi trong toàn bộ những thành tựu chúng tôi đã đạt được.

Bên cạnh đó, chúng tôi cũng xin cảm ơn các thầy cô, anh chị đang công tác tại Phòng thí nghiệm An toàn thông tin - InSecLab vì đã luôn tạo điều kiện về chuyên môn lẫn kinh nghiệm trong các hoạt động nghiên cứu và thực hiện khoá luận.

Cuối cùng, do kiến thức chuyên môn còn hạn chế nên khóa luận chắc chắn không tránh khỏi những thiếu sót. Rất mong nhận được nhận xét, ý kiến đóng góp, phê bình từ quý thầy cô trong hội đồng để khóa luận được hoàn thiện hơn.

Nhóm thực hiện.

Mục lục

Danh sách hình vẽ

Danh mục từ viết tắt

CI	**C**ontinuous **I**tegration
CD	**C**ontinuos **D**elivery
NPM	**N**ode **P**ackaged **M**anager
SDLC	**S**ystems **D**evelopment **L**ife **C**ycle
SAST	**S**tatic **A**pplication **S**ecurity **T**esting
DAST	**D**ynamic **A**pplication **S**ecurity **T**esting
SCA	**S**oftware **C**omposition **A**nalysis
CVE	**C**ommon **V**ulnerabilites and **E**xposures
CWE	**C**ommon **W**eaknesses **E**numeration
OSS	**O**pen **S**ource **S**oftware
SRS	**S**oftware requirements secification
API	**A**pplication **P**rogramming **I**nterface
DevOps	**D**evelopment **O**perations
DevSecOps	**D**evelopment **S**ecurity **O**perations

TÓM TẮT KHOÁ LUẬN

Với sự bùng nổ về nhu cầu người dùng cùng với sự thịnh hành của vi dịch vụ, ngày càng nhiều công ty cần phải cập nhật và phát hành phần mềm của mình liên tục để đảm bảo sản phẩm của mình luôn được mới nhất. Tuy nhiên việc phải phát triển liên tục như vậy dẫn đến những khó khăn như chưa có sự nhất quán và đồng bộ giữa bộ phận phát triển và bộ phận vận hành. Do đó, vai trò DevOps được giới thiệu để giải quyết vấn đề này. DevOps được định nghĩa là sự kết hợp của những kiến thức cùng với vận dụng những công cụ để hỗ trợ, đảm bảo tính nhất quán, đồng bộ giữa hai bộ phận phát triển và vận hành khi bàn giao sản phẩm. Tuy nhiên, tính bảo mật trong DevOps vẫn chưa được đặt nặng khi mục tiêu của DevOps là thu hẹp khoảng cách giữa hai bộ phận, tiết kiệm thời gian và tài nguyên. Do đó, chúng tôi muốn đề cạp đến định nghĩa mới là DevSecOps với mục tiêu tương tự DevOps nhưng luôn kèm tính bảo mật xuyên suốt quy trình phát triển và vận hành. Trong phạm vi khóa luận, chúng tôi thiết kế và giới thiệu mô hình hệ thống tích hợp bảo mật cho vi dịch vụ, được xây dựng dựa trên các bộ công cụ mã nguồn mở có thể kết hợp với nhau để tạo thành hệ thống hoàn chỉnh từ giai đoạn phát triển đến phát hành sản phẩm. Bên cạnh đó, chúng tôi còn đề xuất những ý tưởng thiết kế, tiêu chí đánh giá, những công cụ mã nguồn mở có thể áp dụng cho việc thiết kế và phát triển phần mềm.

Chương 1

TỔNG QUAN ĐỀ TÀI

1.1 Lý do chọn đề tài

Với sự thay đổi chóng mặt của ngành công nghệ kèm theo sự cạnh tranh của các công ty phát triển phần mềm cho khách hàng từ đó nhu cầu tiết kiệm thời gian, nhân lực và chi phí được đưa lên hàng đầu. Nhưng với những lập trình viên (Developer) không có kiến thức chuyên sâu về quản trị hệ thống và phát triển sản phẩm sẽ khiến cho sản phẩm trở nên kém chất lượng, không được kiểm định chặt chẽ khi triển khai ra thị trường và thời gian cập nhật sản phẩm mới sẽ không được ổn định. Vì vậy, để giải quyết cho nhu cầu này DevOps (Development và Operations) được ra đời với tiêu chí loại bỏ rào cản giữa đội ngũ phát triển phần mềm và đội ngũ vận hành phần mềm. DevOps là một người có kiến thức về lập trình, CI/CD, triển khai vận hành quản trị hệ thống máy chủ, hạ tầng mạng.
Nhưng với thời đại tội phạm số ngày càng phát triển thì DevOps vẫn đặt tiêu chí tiết kiệm thời gian, chi phí lên hàng đầu. Nên tính bảo mật và quyền riêng tư chưa được chú trọng. Bên cạnh đó việc tích hợp bảo mật có thể tốn rất nhiều thời gian gây ảnh hưởng, trì trệ trong việc phát hành sản phẩm. Do đó để khắc phục, thuật ngữ DevSecOps được ra đời nhằm bổ sung tính an toàn bảo mật của sản phẩm nhưng vẫn giữ được giá trị trước đó của DevOps.
Để góp phần củng cố ý tưởng này, bài nghiên cứu này sẽ thực hiện triển khai mô hình DevSecOps cho ứng dụng web với mục tiêu sử dụng các phần mềm mã nguồn mở vào quy trình Development, Security, Operation và tích hợp thành một quy trình tự động hóa có kèm khả năng phát hiện lỗ hổng và đưa ra đề xuất sửa lỗi của tác giả Guru Bhandari [3].

1.2 Mục tiêu nghiên cứu

Nghiên cứu, thiết kế, xây dựng hệ thống cho các ứng dụng web giúp cải thiện độ bảo mật, tính liên tục của hệ thống đồng thời tiết kiệm được thời gian và tài nguyên.
Gợi ý các công cụ mã nguồn mở có thể tích hợp sử dụng với nhau xây dựng thành hệ thống.

1.3 Phạm vi nghiên cứu

Các công cụ mã nguồn mở phát hiện các lỗ hổng bảo mật, xây dựng và quản trị hệ thống
Tập trung nghiên cứu cách xây dựng mô hình hệ thống hoàn chỉnh, đảm bảo về yếu tố bảo mật và an toàn thông tin.

1.4 Đối tượng nghiên cứu

Những mô hình, thiết kế có liên quan về đề tài.
Các phần mềm mã nguồn mở phục vụ cho mục đích Development, Security và Operation.
Những khó khăn khi áp dụng đánh giá bảo mật vào quy trình tự động hóa.

1.5 Phương pháp thực hiện

Tìm hiểu kiến trúc, nguyên tắc của DevSecOps và những khó khăn khi áp dụng.
Tìm hiểu về CVEfixes, mô hình, phần mềm, công cụ quét mã nguồn mở và cách sử dụng hỗ trợ xây dựng mô hình.
Tìm hiểu các công trình nghiên cứu đã được thực hiện trong và ngoài nước.
Đưa ra phương pháp xây dựng và triển khai hệ thống.

Thực nghiệm và đánh giá kết quả.

1.6 Ý nghĩa khoa học và thực tiễn của đề tài

1.6.1 Ý nghĩa khoa học

Đề xuất hệ thống DevSecOps hoàn chỉnh các yếu tố bảo mật trong vòng đời phát triển trong và ngoài mã nguồn. Với việc sử dụng các công cụ quét cùng với bộ lưu trữ bản vá để đảm bảo tính an toàn của mã nguồn, ngoài ra tưởng lửa cũng được áp dụng để tăng tính an toàn của hệ thống. Bên cạnh đó, nhằm đảm bảo tính hợp thời của CVEfixes, chúng tôi còn chỉnh sửa, thiết kế lại cơ sở dữ liệu CVEfixes của nhóm tác giả Guru Bhandari, Amara Naseer và Leon Moonen [4], do bộ lưu trữ này chỉ lưu những CVE từ năm 2017 trở về trước. Đồng thời gợi ý một hướng tiếp cận có thể tích hợp CVEfixes vào hệ thống áp dụng quy trình DevSecOps. Chúng tôi hi vọng đề tài nghiên cứu này có thể góp phần xây dựng tư duy thiết kế hệ thống, cách kết hợp bảo mật vào qui trình phát triển phần mềm trong thời đại công nghệ ngày càng phát triển liên tục.

1.6.2 Ý nghĩa thực tiễn

Gợi ý bộ công cụ mã nguồn mở có thể sử dụng trong quy trình phát triển phần mềm. Cập nhật mới cơ sở dữ liệu các lỗ hổng và bán vá của chúng, đề xuất hướng tiếp cận với cơ sở dữ liệu này. Giới thiệu tư duy thiết kế và bảo trì hệ thống. Đề xuất mô hình có thể làm khuôn mẫu và mở rộng quy mô mà không bị hạn chế.

1.7 Cấu trúc Khóa luận tốt nghiệp

Khóa luận được tổ chức trong 5 chương như sau:

- Chương 1: TỔNG QUAN ĐỀ TÀI.
  Trình bày khái quát định hướng nghiên cứu của khóa luận mà chúng tôi muốn hướng tới.

- Chương 2: TỔNG QUAN TÌNH HÌNH NGHIÊN CỨU.
  Sơ lược một số công trình liên quan có cùng hướng nghiên cứu mà đề tài có tham khảo. Giới thiệu các công trình khoa học liên quan chúng tôi đã công bố trong thời gian thực hiện khoá luận.
- Chương 3: CƠ SỞ LÝ THUYẾT.
  Trình bày các định nghĩa, khái niệm cũng như những kiến thức nền tảng để có thể thực hiện được nghiên cứu.
- Chương 4: PHƯƠNG PHÁP THỰC HIỆN.
  Là phần trọng tâm của khoá luận, trình bày những nội dung chính về phương pháp thực hiện và mô hình được sử dụng.
- Chương 5: THỰC NGHIỆM VÀ ĐÁNH GIÁ KẾT QUẢ.
  Đề cập đến quá trình thực nghiệm cùng với kết quả thu được.
- Chương 6: TÀI LIỆU THAM KHẢO.

Chương 2

CƠ SỞ LÝ THUYẾT

2.1 Chu kỳ phát triển phần mềm

Chu kỳ phát triển phần mềm là một quá trình cho một dự án phần mềm. Bao gồm các kể hoạch chi tiết, mô tả hướng đi từ giai đoạn phát triển đến duy trì, thay đổi và nâng cấp. Quy trình là một trong những yếu tố quan trọng quyết định đến chất lượng sản phẩm và sự thành công của nhà phát triển đó. Một chu trình gồm sáu giai đoạn:

2.1.1 Thu thập và phân tích các yêu cầu

Đội ngũ phát triển sẽ lập kế hoạch cho các yêu cầu và nhận ra các rủi ro liên quan. Giai đoạn này giúp đưa ra bức tranh rõ ràng hơn về phạm vi và các vấn đề, hỗ trợ đội ngũ có cái nhìn tổng quan hơn.

2.1.2 Nghiên cứu khả thi

Khi giai đoạn phân tích yêu cầu hoàn thành, giai đoạn này sẽ xác định và ghi lại nhu cầu phần mềm. Được hỗ trợ bởi tài liệu đặc tả hay còn được biết đến là SRS bao gồm mọi thứ cần được thiết kế và phát triển trong vòng đời dự án.

2.1.3 Thiết kế

Tài liệu thiết kế về hệ thống và phần mềm trong giai đoạn này sẽ được chuẩn bị dựa theo tài liệu đặc tả của giai đoạn trước đó.

Đóng vai trò là đầu vào cho giai đoạn tiếp theo của mô hình chu kì phát triển phần mềm.

2.1.4 Giai đoạn lập trình

Bắt đầu phát triển thực tế sản phẩm được xây dựng, hiện thực hóa các chức năng của phần mềm đã được thiết kế sẵn ở giai đoạn trước đó. Các chức năng sau khi được phát triển sẽ được chuyển sang giai đoạn tiếp theo để thực hiện các kịch bản kiểm thử để đánh giá về hiệu năng và chất lượng.

2.1.5 Kiểm thử

Các kiểm thử viên sẽ tiến hành kiểm tra các chức năng theo các kịch bản kiểm thử được xây dựng. Nếu có lỗi phát sinh kiểm thử viên sẽ thực hiện báo lỗi về cho lập trinh viên để tiến hành sửa lỗi.
Quá trình này sẽ thực hiện liên tục nhiều lần cho đến khi các chức năng thực hiện đúng mục đích theo tài liệu đặc tả.

2.1.6 Triển khai và bảo trì

Sau khi hoàn thiện các chức năng và thông qua các kịch bản kiểm thử trong môi trường thử nghiệm, sản phẩm sẽ được chuyển sang môi trường vận hành và được bảo trì liên tục.
Giai đoạn này sẽ xảy ra ba hoạt động chủ yếu:

- Sửa lỗi: Lỗi được báo cáo sau khi phát sinh và được sửa nhanh nhất có thể.
- Nâng cấp: Sản phẩm sẽ được nâng cấp theo từng phiên bản mà nhà phát triển đề ra .
- Cải tiến: Thêm một số chức năng mới vào sản phẩm hiện có.

Trọng tâm chính của giai đoạn này là đảm bảo nhu cầu được đáp ứng liên tục và hệ thống hoạt động đúng theo những tiêu chí được đề ra.

2.2 Tổng quan về DevOps/DevSecOps, CI/CD, CVE và CVEfixes, Thu thập dữ liệu, Kubernetes, tình hình nghiên cứu và các công trình liên quan

2.2.1 DevOps

DevOps là văn hóa làm việc dựa theo sự hợp tác, bổ sung, giúp hai giai đoạn phát triển (Development) và vận hành (Operations) diễn ra trơn tru hơn.

- Giai đoạn phát triển: bao gồm phần việc của thiết kế viên, lập trình viên và kiểm thử viên.
- Giai đoạn vận hành: gồm sự tham gia của kỹ sư hệ thống, quản trị viên hệ thống, vận hành viên và kỹ sư mạng.

Đối với những sản phẩm vừa và nhỏ lập trình viên sẽ là người kiêm vị trí vận hành viên vì là người hiểu rõ sản phẩm của mình và cách triển khai nó. Nhưng khi sản phẩm bắt đầu có quy mô lớn hơn và sự thịnh hành của vi dịch vụ dẫn đến quy mô hệ thống tăng lên đáng kể từ vài chục máy chủ tăng đến hàng trăm hàng nghìn.
Khi đó nhu cầu chuyên môn trở nên gắt gao, tách giai đoạn phát triển và vận hành thành hai giai đoạn riêng biệt. Từ đó dẫn đến bài toán lập trình viên không còn toàn quyền triển khai và đội ngũ vận hành không thể hiểu rõ cách sản phẩm vận hành như lập trình viên khiến cho việc đồng bộ trở nên khó khăn dẫn đến tốn kém tài nguyên và thời gian.
Vì vậy DevOps ra đời để giải quyết khó khăn và mang đến những lợi ích:

- Tăng cường sự cộng tác chặt chẽ giữa nhóm phát triển và nhóm vận hành.
- Nâng cao tần suất triển khai, rút ngắn thời gian phát triển/cải tiến sản phẩm.
- Tận dụng các công cụ tự động hóa, giúp hạn chế rủi ro, giảm thiểu tỉ lệ thất bại.
- Thời gian phục hồi sản phẩm nhanh hơn.

2.3 DevSecOps

Mặc dù DevOps đã cung cấp tất cả nhưng yêu cầu cần thiết cho các công ty và tổ chức. Tuy nhiên, với công nghệ ngày càng phát triển thì những lo ngại về bảo mật càng trở thành mối quan ngại chính của các tổ chức. Những mỗi quan tâm này có thể là nguyên nhân dẫn đến dừng hoặc trì hoãn việc triển khai sản phẩm. Để giải quyết vấn đề đó, các tổ chức xem việc thúc đẩy cải thiển DevOps và kết hợp bảo mật vào tự động hóa xuyên suốt vòng đời phát triển phần mềm là nhu cầu cấp thiết. Không chỉ là một bước riêng biệt mà là một phần của toàn bộ vòng đời ứng dụng. Do đó DevSecOps được ra đời để giải quyết khó khăn này.
DevSecOps viết tắt của Development (phát triển), Security (bảo mật) và Operations(triển khai), đại diện cho sự tiển hóa và đồng bộ các bộ phận lại với nhau.Trước đây, bảo mật chỉ được một nhóm bảo mật áp dụng riêng biệt vào giai đoạn cuối của quy trình phát triển, việc này chỉ hiệu quả khi sản phẩm được cập nhật một hoặc hai lần một năm. Với việc các nhà phát triển áp dụng phương pháp Agile và DevOps nhằm mục đích giảm chu kỳ phát triển xuống còn vài tuần hoặc vài ngày thì cách bảo mật truyền thống này tạo ra một khúc mắc khó giải quyết được. Vì vậy, DevSecOps là nhu cầu cấp thiết cần được tiếp cận.
Với việc tích hợp ứng dụng và bảo mật cơ sở hạ tầng một cách liền mạch vào các quy trình của DevOps. DevSecOps giai quyết các vấn đề bảo mật khi chúng xuất hiện, khiến cho việc sửa lỗi dễ dàng hơn, nhanh hơn và ít tốn kém hơn, giúp cho ứng dụng và bảo mật cơ sở hạ tặng trở thành trách nhiệm chung của mọi bộ phận thay vì chỉ mỗi nhóm bảo mật. Với phương châm "an toàn hơn, phần mềm tung ra sớm hơn" ("Safer, Software Sooner") DevSecOps luôn đảm bảo tính an toàn của phần mềm bằng cách tự động hóa mà không làm trì trệ chu kỳ phát triển phần mềm.
Tương tự như DevOps, DevSecOps cũng gồm các thành phần:

- CI/CD: phân phối nhanh chóng và an toàn các sản phẩm
- Cơ sở hạ tầng dưới dạng mã: tài nguyên máy tính đáp ứng và co giãn bất cứ khi nào cần thay đổi
- Giám sát: mọi khía cảnh an ninh đều được giám sát chặt chẽ
- Ghi chép: tất cả các sự kiện bảo mật diễn ra đều được hệ thống ghi lại tỉ mỉ

- Vi dịch vụ: chia các hệ thống khối thành các phần nhỏ hơn, dễ quản lý hơn
- Giao tiếp giữa các bộ phận: các nhóm có thể dễ dàng liên lạc với nhau để đảm bảo cần thận mỗi bước của quy trình được quản lý đầy đủ và không bị bỏ sót

Ngoài ra còn có các phương pháp nổi bật:

- Liệt kê các điểm yếu chung (CWE): cải thiện chất lượng code và tăng mực độ bảo mật trong CI/CD
- Mô hình hóa mối đe dọa: thực hiện kiểm tra bảo mật trong quá trình phát triển để tiết kiệm thời gian và chi phí
- Kiểm tra bảo mật tự động: kiểm tra lỗ hổng bảo mật thường xuyên
- Quản lý sự cố: tạo ra một khuôn khổ tiêu chuẩn để ứng phó với các sự cố bảo mật

Lợi ích khi ứng dụng DevSecOps:

- Triển khai phần mềm nhanh chóng, tiết kiệm chi phí
- Chủ động và cải thiện bảo mật
- Tăng tốc vá lỗ hổng bảo mật
- Tự động hóa tương thích với sự phát triển hiện đại
- Quy trình có thể lặp lại và thích ứng

2.3.1 Tích hợp liên tục (CI), chuyển giao liên tục (CD)

Tích hợp liên tục (CI)

Tích hợp liên tục (Continuous Integration hay CI) là một phương pháp trong phát triển phần mềm tích hợp công việc một cách thường xuyên. Mỗi khi có thay đổi được thực hiện thì quy trình tích hợp sẽ tự động diễn ra bao gồm cả kiểm thử nhằm phát hiện lỗi nhanh nhất có thể và cho phép phát triển và kết hợp các tính năng của sản phẩm lại nhanh hơn.
Quy trình của tích hợp liên tục:

- Các lập trình viên sẽ đăng code của mình lên kho chứa code
- Hệ thống tích hợp kiểm tra xem có sự thay đổi nào trên kho chứa hay không
- Khi có sự thay đổi diễn ra thì hệ thống sẽ tự bắt đầu quy trình tích hợp lấy code từ kho chứa về và bắt đầu xây dựng, và kiểm thử sản phẩm
- Sau khi kết thúc tích hợp hệ thống sẽ gửi phản hồi về các thành viên
- Hệ thống trở lại trạng thái chờ tín hiệu thay đổi trên kho chứa code

Chuyển giao liên tục (CD)

Chuyển giao liên tục (Continuous Delivery hay CD) là quy trình chuyển giao tất cả thay đổi về code đã được xây dựng và kiểm thử trước đó sang môi trường kiểm thử để các kiểm thử viên có thể tiến hành các kịch bản kiểm thử như là kiểm tra giao diện, kiểm tra API và kiểm tra tích hợp trước khi chuyển sang môi trường triển khai sản phẩm.

Lợi ích của CI/CD:

- Giảm thiểu rủi ro nhờ phát hiện lỗi từ sớm, tăng chất lượng phần mềm qua các lần kiểm thử tự động
- Giảm thiểu những quy trình thủ công lặp đi lặp lại
- Có thể tự động chạy bất kể thời gian và địa điểm.

2.4 CVE, CVEfixes và Thu thập dữ liệu

2.4.1 CVE, CVEfixes

Lỗi bảo mật máy tính công khai (Common Vulnerabilities Exposures hay CVE) là danh sách các lỗ hổng bảo mật trên máy tính được tiết lộ công khai. Với mới lỗi bảo mật sẽ có một mã định danh đi kèm để phân biệt. CVE được tiến hành vào năm 1999 với mục đích xác định và phân loại các lỗ hổng thường gặp. Trong đó, mỗi lỗ hổng sẽ có một bản mô tả chi tiết tương ứng, các chuyên gia an ninh mạng

sẽ sử dụng bản mô phỏng này để thảo luận và đưa ra hướng khắc phục lỗ hổng. Các yêu tố hình thành một CVE:

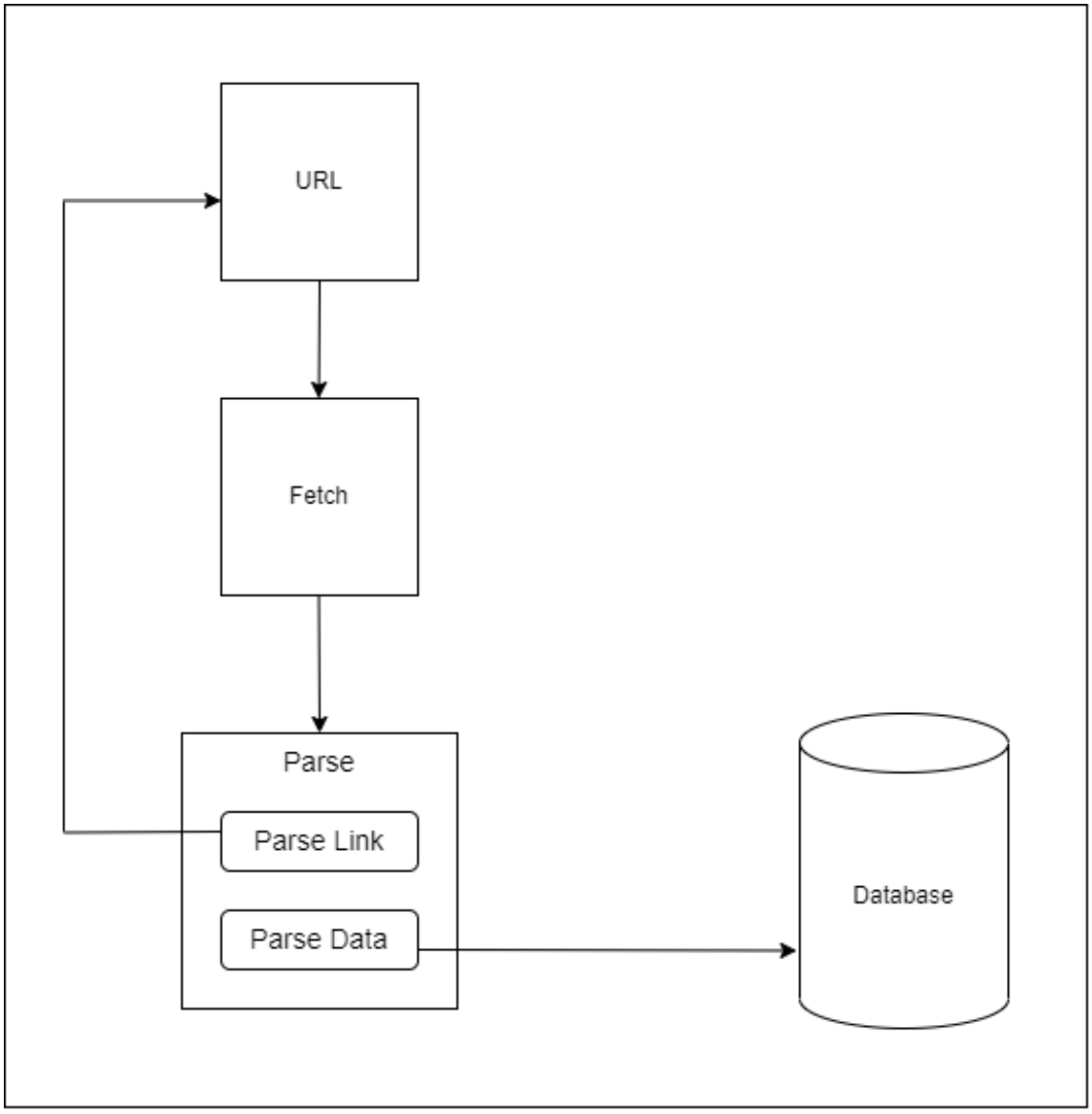
- Lỗ hổng tác động tiêu cực đến tình trạng an ninh: lỗ hổng gây tác động tiêu cực đến tình trạng bảo mật của đơn vị đang sử dụng sản phẩm có lỗ hổng.
- Lỗ hổng có thể khắc phục độc lập: quá trình khắc phục lỗ hổng có thể được thực hiện một cách độc lập mà không gay ảnh hưởng đến hệ thống mạng chung.
- Lỗ hổng chỉ ảnh hưởng đến sản phẩm sử dụng mã nguồn: nếu như lỗ hổng tác động đến những sản phẩm khác mã nguồn khác thì chúng phải được phân loại riêng rẽ

Hiện nay có rất nhiều công cụ để tìm ra những lỗ hỗng CVE nhưng vẫn chưa có những công cụ uy tín nào hỗ trợ việc tìm kiếm những bản vá (fixes) cho những lỗ hổng đó. Bài nghiên cứu của nhóm tác giả Guru Bhandari, Amara Naseer và Leon Moonen [4] giới thiệu về bộ lưu trữ những lỗ hổng cùng với các bản vá (CVEfixes). Đóng góp này của nhóm tác giả đã hỗ trợ mở ra rất nhiều cách áp dụng và cách sửa lỗi kịp thời khi có lỗ hổng xảy ra.

2.4.2 Thu thập dữ liệu (Data Crawling)

Thu thập dữ liệu là thu thập dữ liệu từ một trang bất kỳ, sau đó tiến hành phân tích mã nguồn HTML để đọc dữ liệu và lọc ra theo yêu cầu người dùng hoặc dữ liệu yêu cầu. Lợi ích của Thu thập dữ liệu:

- Giúp cho dữ liệu giàu nội dung hơn
- Cập nhật liên tục các dữ liệu mới
- Tiết kiệm nhân công và chi phí cho việc nhập liệu



HÌNH 2.1: Nguyên tắc hoạt động của Thu thập dữ liệu

Nguyên tắc hoạt động của Thu thập dữ liệu:

- Truy cập vào đường dẫn trang chứa dữ liệu cần cào
- Thu thập dữ liệu, nội dung trang
- Rút trích dữ liệu và nhập vào cơ sở dữ liệu, đồng thời rút trích ra đường dẫn kế tiếp và tiến hành thu thập

- Lặp lại cho đến khi hết đường dẫn có thể thu thập dữ liệu trong trang

2.5 Kubernetes

2.5.1 Tổng quan về Kubernetes

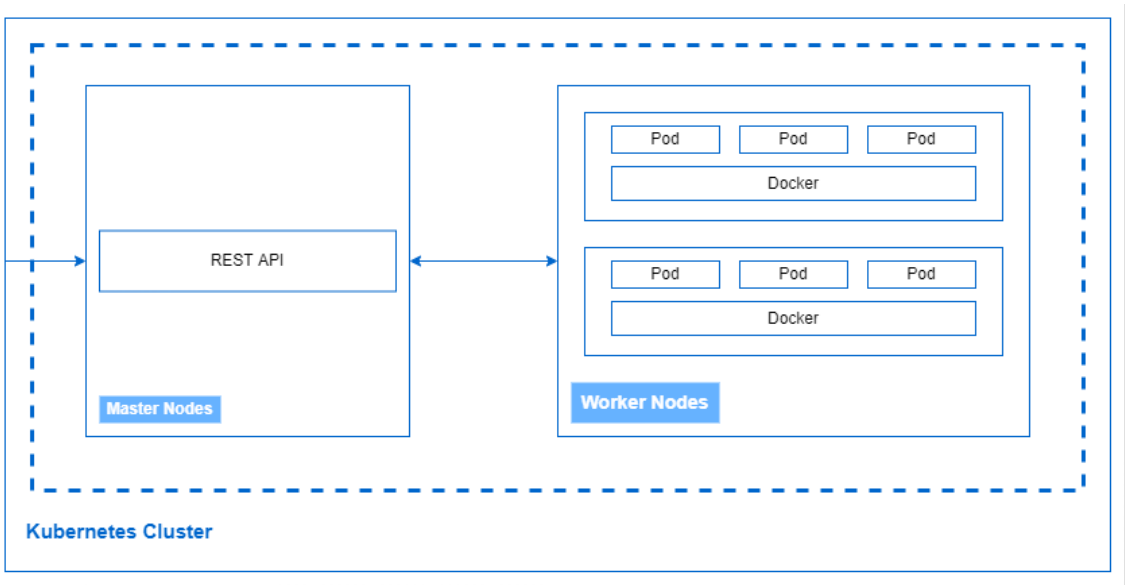
Kubernetes là công cụ mã nguồn mở được phát triển bởi đội ngũ kỹ sư Google, phục vụ cho việc kiểm soát và điều khiển các dịch vụ để phần mềm hoạt động một cách chặt chẽ. Do sự thay đổi chóng mặt về sự mô hình phát triển phần mềm, đi từ phát triển phần mềm theo hướng khối sang phát triển theo hướng vi dịch vụ, chính điều đó là tác nhân đẩy mạnh đến nhu cầu phát triển và hỗ trợ vi dịch vụ. Kubernetes tạo nền tảng vững chắc cho các doanh nghiệp có thể tự tin thiết lập, xây dựng và phát triển phần mềm doanh nghiệp của họ một cách dễ dàng và an toàn hơn.

2.5.2 Sự khác biệt giữa Docker và Kubernetes

Docker là ứng dụng mã nguồn mở cho phép giả lập hệ điều hành mà không cần thiết phải thay đổi hệ điều hành để cho ứng dụng có thể chạy được. Với Docker chúng ta có thể tạo ra nhiều dịch vụ mà không lo lắng về môi trường cài đặt, mạng và dung lượng cho từng ứng dụng, là một ứng dụng mang tính đột phá và đang là xu hướng để phát triển phần mềm ở thời điểm hiện tại. Tuy nhiên Docker lại mang nhiều điểm hạn chế so với Kubernetes:

- Chỉ có thể tạo ra nhiều dịch vụ trong cùng một máy duy nhất
- Có thể là nơi được sử dụng để phát triển và phát hành ứng dụng theo mô hình vi dịch vụ. Tuy nhiên, trên thực tế Docker chỉ nên là nơi phát triển ứng dụng trước khi ra sản phẩm hơn là phát hành sản phẩm.
- Việc sử dụng một máy duy nhất để phát hành mô hình vi dịch vụ sẽ phát sinh nhiều vấn đề. Ví dụ nếu máy đó bị sập thì toàn bộ hệ thống vi dịch vụ sẽ bị sập hoàn toàn.
- Không hỗ trợ nhiều về mặt kiểm soát như kiểm soát luồng, an ninh bảo mật, quản lý dung lượng, ... như Kubernetes.

Với Kubernetes các doanh nghiệp có thể dễ dàng mở rộng hoặc thu hẹp phạm vi của hệ thống vi dịch vụ, tự động kiểm tra tình trạng của từng dịch vụ để thuận tiện cho kiểm soát luồng, tự động nhân bản các dịch vụ đề phòng sự cố, quản lý cài đặt mạng, băng thông một cách dễ dàng hơn.



HÌNH 2.2: Kiến trúc của Kubernetes

Cụm máy tính Kubernetes (Kubernetes Cluster): bao gồm 1 loạt các cụm máy tính được cài đặt ứng dụng Docker và Kubernetes liên kết với nhau, mỗi cụm máy tính luôn luôn gồm một máy chủ (Master Node) để điều khiển và kiểm soát các đơn vị máy làm việc (Worker Nodes).
Đơn vị máy (Nodes) bao gồm 2 thành phần chính:

- Cụm máy chủ: đóng vai trò vừa là nơi để kiểm soát các đơn vị làm việc nhưng cũng đồng thời cung cấp một cổng REST API tạo cầu nối cho việc giao tiếp giữa người dùng và hệ thống bên trong.
- Cụm máy làm việc: là những máy được cài đặt Docker cho việc giả lập dịch vụ và Kubernetes cho việc liên kết thành các cụm máy tính với nhau.

Pods: Là những dịch vụ được giả lập và chạy trong một đơn vị máy.
Bảo mật và cấu hình (Secret and Configuration): cho phép doanh nghiệp có thể bảo vệ các thông tin bí mật như khóa, mật khẩu, ... giúp doanh nghiệp dễ dàng

tự động thiết lập và triển khai môi trường cho dịch vụ một cách nhanh chóng chỉ cần một tệp cấu hình.

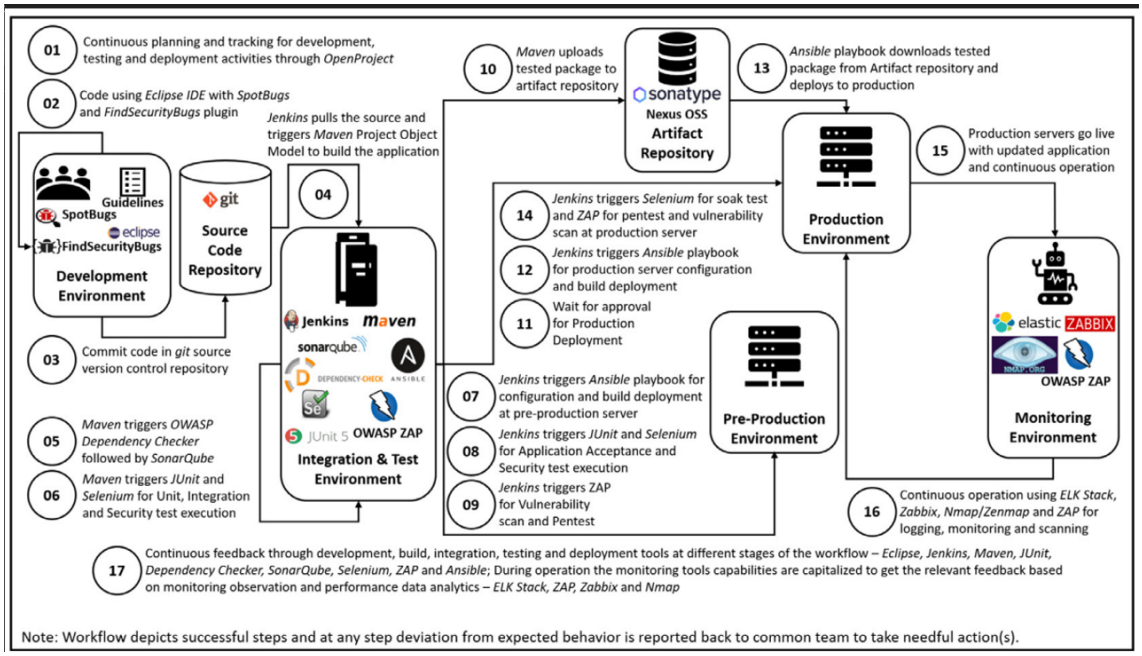
2.6 Tình hình nghiên cứu và các công trình liên quan

Một số nghiên cứu liên quan đến những khó khăn khi áp dụng DevSecOps và giải pháp xây dựng hệ thống cùng với các mô hình triển khai nghiên cứu có thể kể đến như nghiên cứu của nhóm tác giả Roshan N. Rajapakse [1], nghiên cứu của nhóm tác giả Kennedy A. Torkura [2] hay giải pháp mô hình ADOC [3] của nhóm tác giả Rakesh Kumar.

Cụ thể, trong nghiên cứu [1], nghiên cứu của nhóm tác giả Roshan N. Rajapakse phân tích cụ thể những khó khăn điển hình khi áp dụng DevSecOps thay cho DevOps thông thường. Cụ thể hơn, nhóm tác giả phân tích chuyên sâu từng giai đoạn trong Systems Development Life Cycle (SDLC) sẽ như thế nào nếu áp dụng DevSecOps cùng với đó là những quy trình CI/CD với những bất cập khi áp dụng Continuous Security vào quy trình, cũng như là những khó khăn về mặt nhân lực giữa các đội ngũ Development, Operation, và Security khi hợp tác với nhau.

Nghiên cứu [2] giới thiệu khái quát về khái niệm Continuous Security Assessment cho Microservices và Cloud Native Applications (CNA) cùng với đó là các ý tưởng về tiêu chí đánh giá bảo mật như là Security Gateway và Security Health EndPoints cho các ứng dụng để đảm bảo an toàn thông tin và quyền riêng tư.

Ở nghiên cứu [3] nhóm tác giả đề xuất sử dụng mô hình "A conceptual model for automated DevSecOps using open-source software over cloud" (ADOC) bằng việc sử dụng các phần mềm công cụ mã nguồn mở open-source software (OSS) hỗ trợ cho việc tích hợp Continuous Security Assessment trong quy trình CI/CD bằng những công cụ như Static Application Security Testing (SAST), Dynamic Application Security Testing (DAST), SonarQube. Cùng với đó là workflow đề xuất của tác giả qua từng giai đoạn từ development đến operations.



HÌNH 2.3: Mô hình để xuất ADOC của nhóm tác giả Rakesh Kumar[3]

Bài báo khoa học của nhóm tác giả Guru Bhandari, Amara Naseer và Leon Moonen [4] đề cập một bộ sưu tập các lỗ hổng cùng với các bản vá sửa lỗi đã góp phần hỗ trợ cho việc tự động hóa đánh giá bảo mật trở nên tin cậy hơn khi đề ra bộ sưu tập CVEfixes cùng với các ứng dụng nó có thể tích hợp.

2.7 Một số cải tiến so với khoá luận trước

Trong đề tài này, nhóm nghiên cứu dự kiến thiết kế mô hình DevSecOps cho một ứng dụng dựa theo các đề xuất về việc tích hợp đánh giá bảo mật liên tục, đồng thời đề xuất ra bộ công cụ mã nguồn mở có thể kết hợp với nhau tạo thành hệ thống hoàn chỉnh cho ứng dụng.

Đề tài tập trung vào xây dựng một hệ thống hoàn chỉnh áp dụng cho ứng dụng web từ giai đoạn phát triển đến giai đoạn triển khai bằng việc kiểm tra tính bảo mật cũng như khả năng xuất hiện lỗ hổng một cách liên tục thay vì chỉ kiểm tra vào giai đoạn cuối.

Chương 3

PHƯƠNG PHÁP THỰC HIỆN

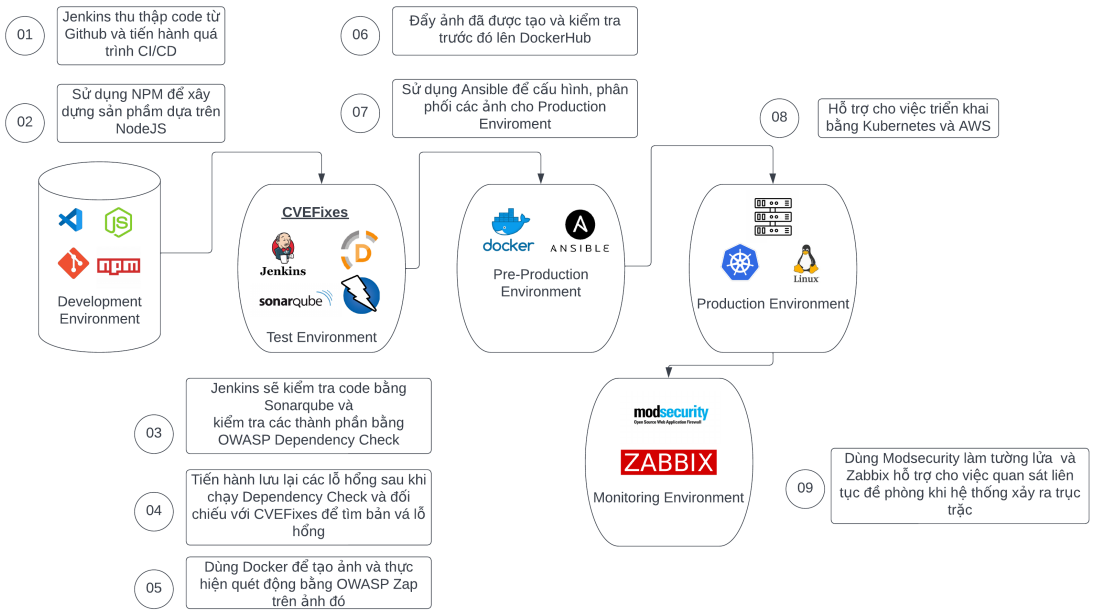
3.1 Giới thiệu những công cụ mã nguồn mở, công nghệ được sử dụng

Nhằm tối ưu hóa cho việc thiết kế hệ thống, sau đây là những công nghệ, công cụ mã nguồn mở được sử dụng:

- Visual Studio Code: trình biên tập lập trình code được phát triển bởi Microsoft
- Git: là một hệ thống quản lý phiên bản phân tán, giúp việc quản lý code và làm việc nhóm trở nên đơn giản hơn
- NodeJS: nền tảng giúp xây dưng các ứng dụng web cho ngôn ngữ JavaScript
- NPM: công cụ tạo và quản lý các thư viện JavaScript cho NodeJS.
- SonarQube: nền tảng mã nguồn mở giúp kiểm tra chất lượng code của dữ án
- OWASP Dependency Check: công cụ dùng để phân tích thành phần trong thư viện được sử dụng, kiểm tra xem có lỗ hổng bảo mật
- OWASP ZAP: công cụ quét động với chức năng tìm các lỗ hổng có thể khai thác của ứng dụng web khi triển khai
- Jenkins: nền tảng mã nguồn mở phục vụ cho việc tích hợp liên tục và triển khai liên tục CI/CD

- Docker: nền tảng hỗ trợ việc ảo hóa các ứng dụng
- Ansible: công cụ quản lý cấu hình, tạo điều kiện thuận lợi cho công việc cài đặt, quản lý và bảo trì từ xa
- Kubernetes: hệ thống phục vụ cho việc quản lý các ứng dụng, giúp thuận lợi trong việc cấu hình và tự động hóa việc tiển khai ứng dụng, hỗ trợ thay đổi kích thước hệ thống
- CVEfixes: bộ lưu trữ các bản vá các lỗ hổng bảo mật

3.2 Mô hình triển khai



HÌNH 3.1: Hình ảnh vòng đời phát triển phần mềm của DevSecOps

Với từng giai đoạn phát triển của chu kỳ phát triển phần mềm, chúng tôi sẽ sử dụng những phương pháp và công cụ tương ứng nhằm đảm bảo tính thực tế của mô hình đồng thời giới thiệu rõ mục tiêu và nhiệm vụ của từng giai đoạn. Ở đây chúng tôi sẽ dựa theo vòng đời phát triển phần mềm của DevSecOps làm minh họa cho hệ thống. Và Jenkins sẽ là công cụ tích hợp để chúng tôi thực quy trình CI/CD hỗ trợ cho việc thiết kế hệ thống.

3.3 Giai đoạn thiết kế

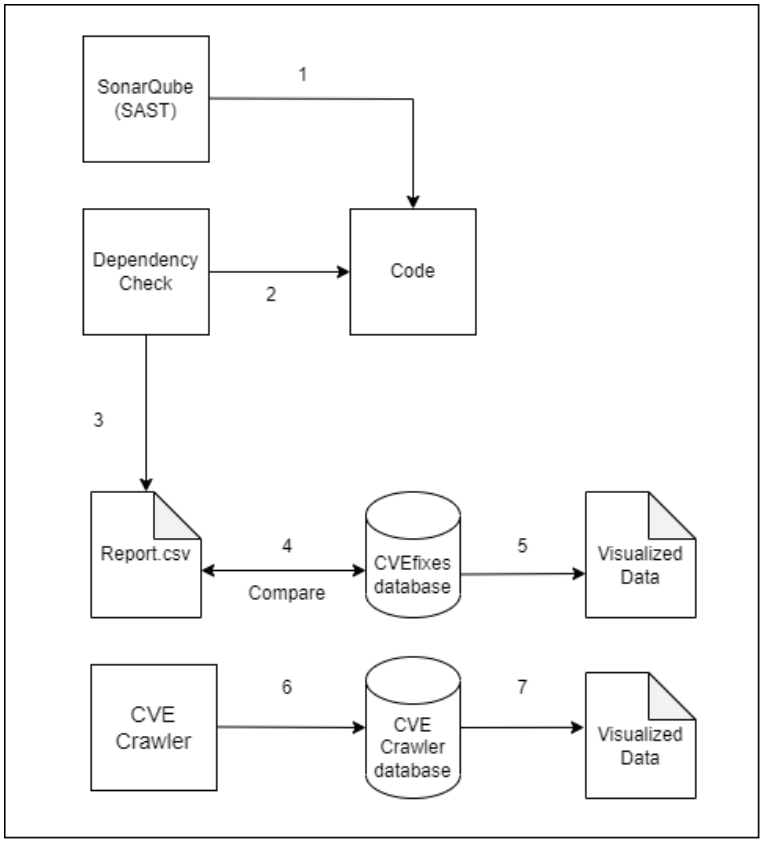
Như những phương pháp thiết kế khác, chúng tôi sẽ đảm nhận vai trò là những lập trình viên thiết kế ra sản phẩm thử nghiệm cho hệ thống DevSecOps là một ứng dụng web vi dịch vụ cho chức năng ghi chú lại những việc cần làm. Vi dịch vụ chúng tôi thiết kế sẽ gồm:

- Một front-end sử dụng thư viện JavaScript mã nguồn mở ReactJS thông qua nền tảng NodeJS
- Bốn back-end với các chức năng hỗ trợ việc lưu trữ, tương tác cho front-end

Ở giai đoạn này chúng tôi sẽ dùng Visual Studio Code để thiết kế mã nguồn, và dùng Git làm công cụ để đẩy lên kho lưu trữ GitHub.

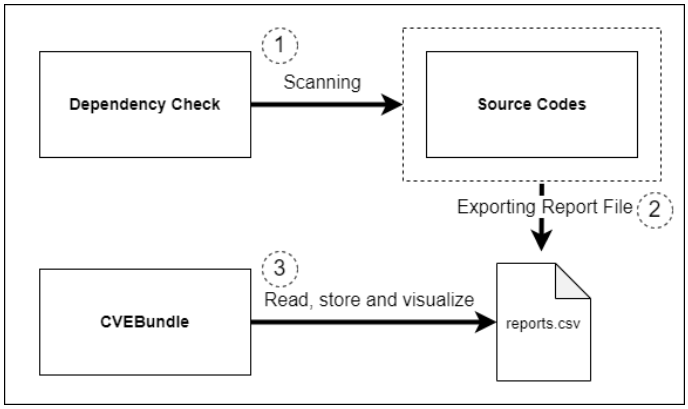
3.4 Giai đoạn xây dựng

Sau khi sản phẩm được thiết kế và đẩy lên kho chứa GitHub. Sản phẩm được kéo về và xây dựng thành thông qua công cụ NPM. Bên cạnh đó để tăng cường khả năng tìm kiếm lỗi và đảm bảo bảo mật. Chúng tôi sẽ áp dụng công cụ quét mã nguồn tĩnh (SAST) SonarScanner của SonarQube để kiểm tra chất lượng mã nguồn.



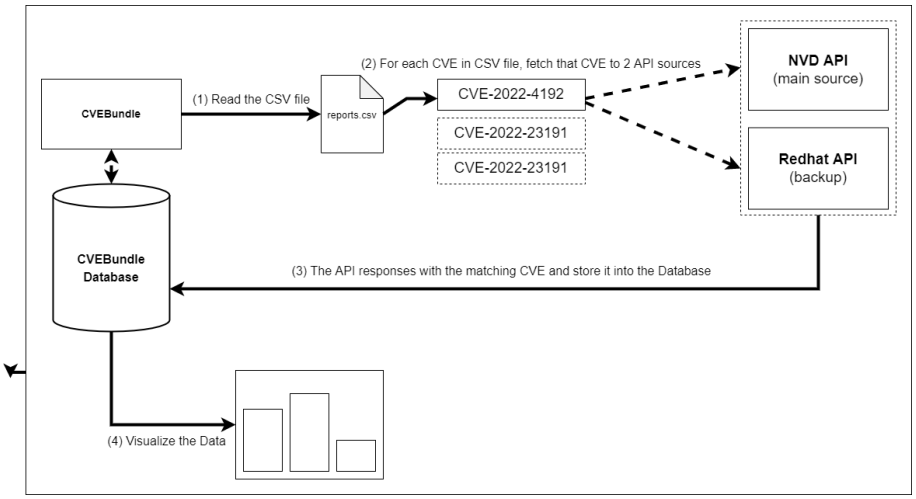
HÌNH 3.2: Luồng hoạt động giai đoạn xây dựng

Ngoài ra để dò quét các lỗ hổng bảo mật CVE, OWASP DependencyCheck (SCA) sẽ được dùng để quét toàn bộ các thư viện đi kèm. Các lỗ hổng bảo mật CVE sau khi được quét sẽ được xuất thành tập tin có định dạng là csv và được so sánh với cơ sở dữ liệu CVEfixes xuất ra thông tin hoàn chỉnh về các lỗ hổng.



HÌNH 3.3: Luồng hoạt động của DependencyCheck với công cụ CVEBundle

CVEBundle là công cụ do chúng tôi thiết kế thực hiện chức năng tương tác giữa tập tin báo cáo được xuất ra từ DependencyCheck với cơ sở dữ liệu CVEBundle, lấy dữ liệu từ NVD API và RedHat API (do cơ sở dữ liệu CVEfixes quá cũ) để thu thập được các dữ liệu mới hơn, sau đó để xuất ra các thông tin cần thiết như là mã định danh lỗ hổng, độ nghiêm trọng của lỗ hổng và sau đó phác thảo dưới dạng cột xác định lỗ hổng nghiêm trọng cần được vá.



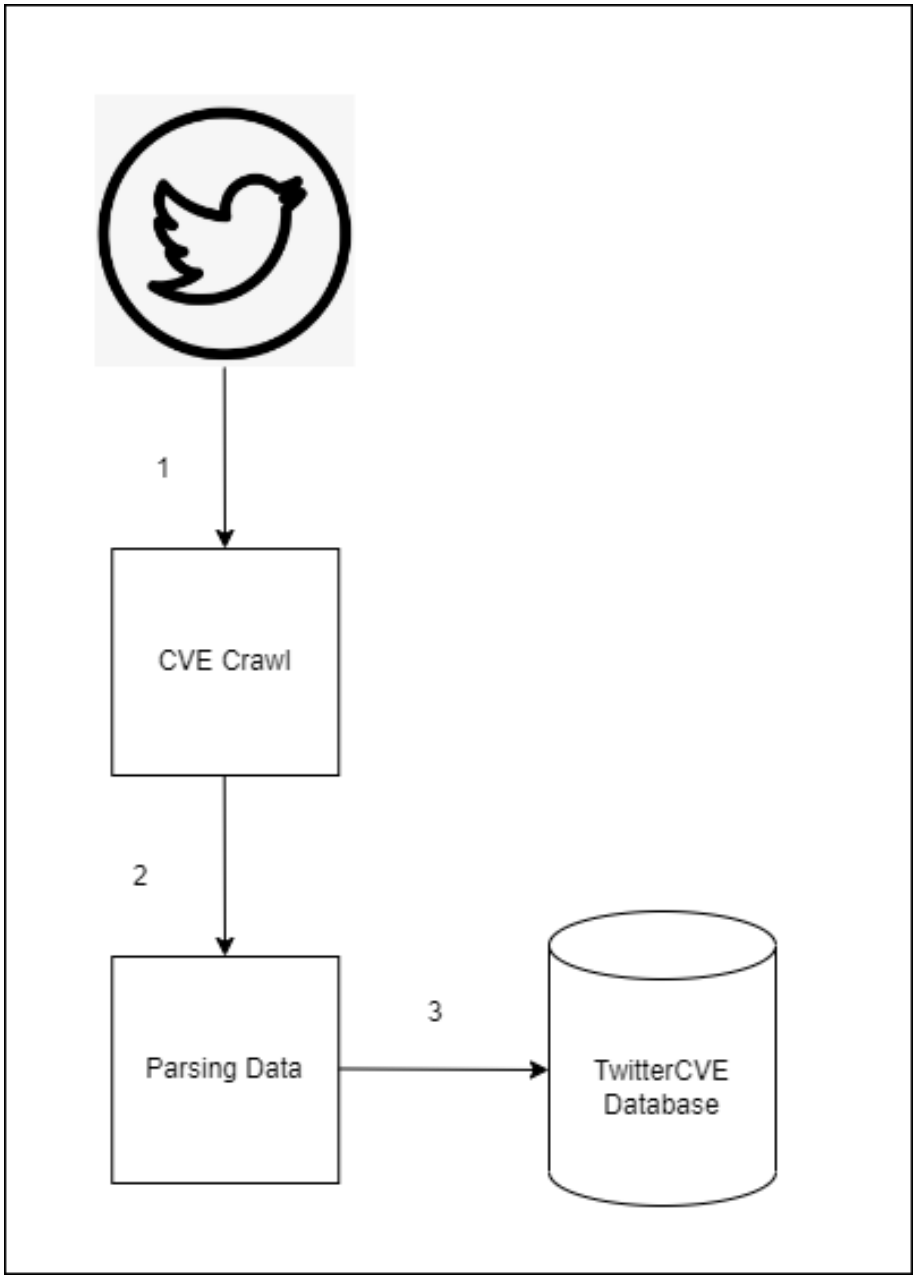
HÌNH 3.4: Cách thức lấy dữ liệu của công cụ CVEBundle

Các giai đoạn hoạt động của CVEBundle

- CVEBundle sẽ đọc dữ liệu của tập tin báo của DependencyCheck

- Với mỗi mã định danh CVE công cụ sẽ thu thập dữ liệu từ nguồn NVD API, nếu dữ liệu không có trên NVD API thì công cụ sẽ tự động thu thập dữ liệu từ RedHat API (do RedHat API chứa những lỗ hổng mới nhất ở thời điểm hiện tại, nhưng NVD API đầy đủ hơn về chi tiết dữ liệu)
- Sau khi thu thập dữ liệu, công cụ sẽ tự động lưu vào cơ sở dữ liệu
- Sau đó sẽ lấy dữ liệu thu thập được phác thảo thành mô hình

Cuối cùng để đánh giá mức độ của lỗ hổng, chúng tôi tự thiết kế công cụ cào dữ liệu trên Twitter để thu thập các lỗ hổng đang được bàn tán, thịnh hành trên mạng xã hội này sau đó so sánh với các lỗ hổng của vi dịch vụ đang phát triển. Nếu các lỗ hổng của vi dịch vụ nằm trong thịnh hành, được bàn tán. Lỗ hổng trong vi dịch vụ đó sẽ được đánh giá độ nguy hiểm là cao và cần được xử lí ngay lập tức.



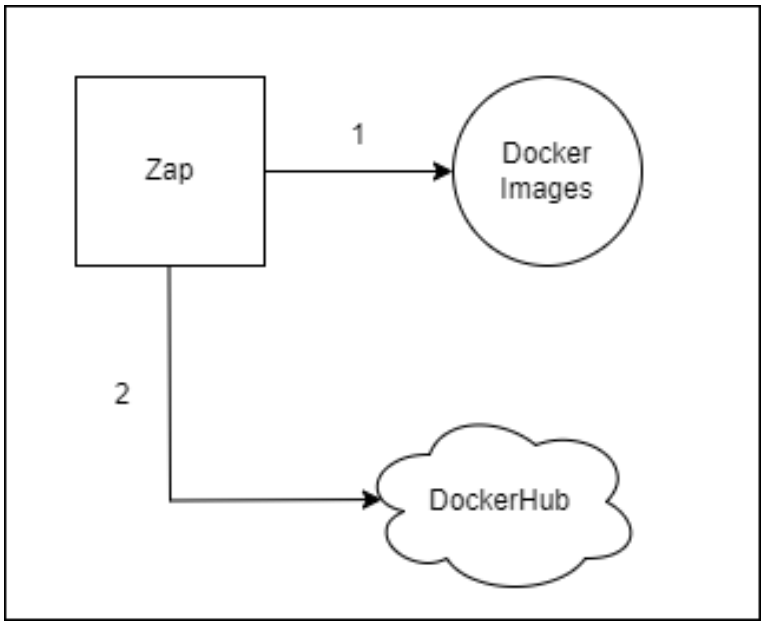
HÌNH 3.5: Cách thức cào dữ liệu từ Twitter

Các bước cào dữ liệu từ Twitter:

- Truy cập đường dẫn đến trang Twitter
- Tìm các bài tweet có từ khóa CVE hoặc các nhãn dán có từ khóa CVE (CVE)
- Thu thập các bài tweet về

- Rút trích các dữ liệu bao gồm mã định danh CVE, ngày bài tweet được đăng, đường dẫn bài tweet, tên người đăng...
- Thông tin về mã định danh CVE đó sẽ được lấy từ NVD API và lưu vào cơ sở dữ liệu
- Hệ thống sẽ tính toán số lần CVE được đề cập và phác thảo thành hình ảnh

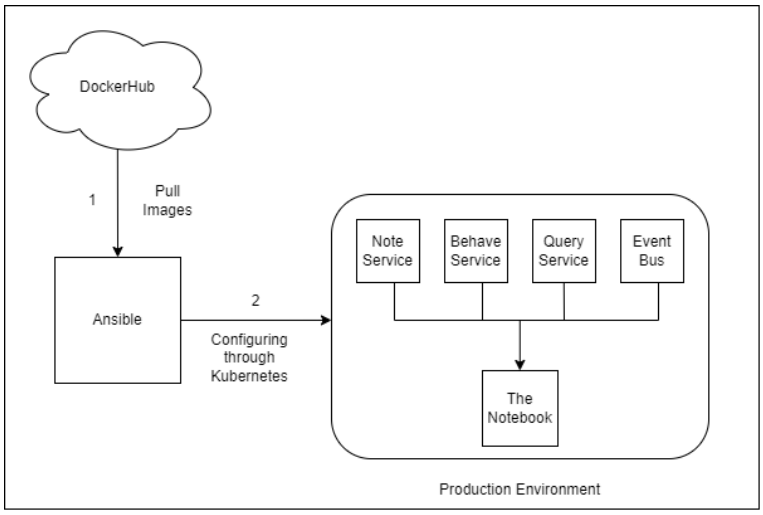
3.5 Giai đoạn kiểm thử



HÌNH 3.6: Luồng hoạt động giai đoạn kiểm thử

Trong giai đoạn xây dựng, ứng dụng web sẽ được ảo hóa thành một ảnh thông qua Docker. Ứng dụng web sau khi được ảo hóa sẽ được chuyển sang môi trường thử nghiệm và được tiến hành quét mã nguồn động (DAST) thông qua công cụ OWASP ZAP để kiểm thử có lỗ hổng có thể xảy ra khi ứng dụng được chạy. Sau khi quá trình quét hoàn tất, ảnh của ứng dụng được tạo trước đó sẽ được đẩy lên DockerHub để bắt đầu chuyển giao sang môi trường triển khai sản phẩm.

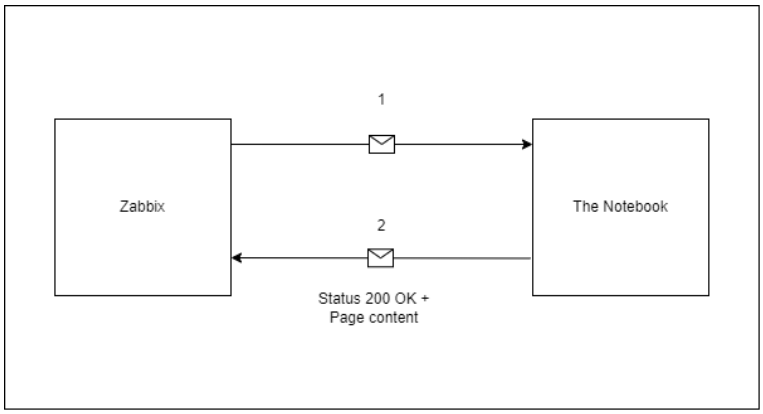
3.6 Giai đoạn phát hành và triển khai



HÌNH 3.7: Luồng hoạt động giai đoạn triển khai

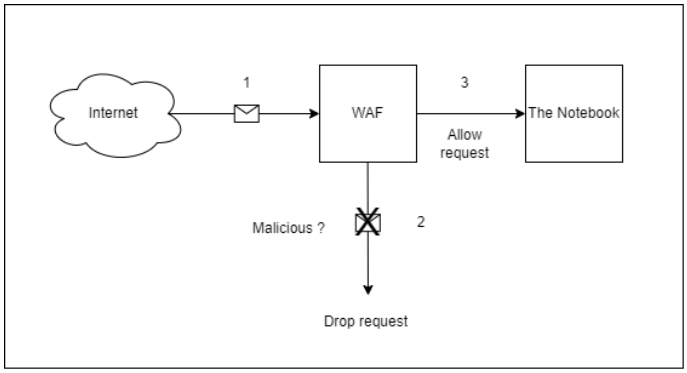
Ở môi trường triển khai sản phẩm, Ansible sẽ được dùng để cấu hình cho môi trường triển khai. Bao gồm việc kéo các ảnh từ DockerHub và vận hành với Kubernetes để đảm bảo tính đồng bộ và tính sẵn sàng khi có một chức năng của ứng dụng xảy ra sự cố.

3.7 Giai đoạn quan sát



HÌNH 3.8: Hình ảnh mô tả cách Zabbix tương tác đến ứng dụng để quan sát

Sau khi được triển khai ở môi trường vận hành, chúng tôi sẽ quan sát và ghi lại những báo cáo về ứng dụng mạng thông qua Zabbix thông qua phương thức gửi yêu cầu đến ứng dụng và chờ nội dung trang cùng với trạng thái trả về.



HÌNH 3.9: Hình ảnh mô tả cách thức tường lửa hoạt động

Đồng thời tường lửa dành cho ứng dụng web của Modsecurity kết hợp với bộ luật của OWASP cũng được chúng tôi sử dụng để tăng tính an toàn và chặn những cuộc tấn công mạng có thể xảy ra. Nếu phát hiện hành động bất thường tường lửa sẽ tự động ngăn chặn không cho người dùng tiếp tục sử dụng.

Chương 4

THỰC NGHIỆM VÀ ĐÁNH GIÁ KẾT QUẢ

4.1 Hiện thực

Thiết bị	Cấu hình	Công cụ chính	Địa chỉ IP
Test Environment	Ubuntu 20.04, 2 Processors 4GB RAM, 25GB Hard Disk	Jenkins,SonarQube, Zap, DependencyCheck, Ansible, Docker, CVEBundle, Crawler	192.168.213.134
Production Environment	Ubuntu 20.04, 2 Processors 4GB RAM, 25GB Hard Disk	Kubernetes, Nginx, Modsecurity, Zabbix	192.168.213.132

Để hiện thực mô hình, hai môi trường giả lập được tạo:

- Môi trường kiểm thử: phục vụ cho việc xây dựng và kiểm tra các lỗ hổng trước khi triển khai
- Môi trường triển khai: phát hành và quan sát để xử lí kịp thời các sự cố

Ở môi trường kiểm thử, Jenkins sẽ là công cụ chủ yếu cho việc tích hợp và triển khai liên tục, ngoài ra các công cụ như SonarQube, Zap, DependencyCheck sẽ được ảo hóa thông qua Docker để thực hiện chức năng quét lỗ hổng.
Ở môi trường triển khai, tường lửa và Zabbix được cài đặt để tiến hành quan sát liên tục, và tiến hành xử lí kịp thời khi có sự cố xảy ra.
Để thực hiện hóa qui trình tích hợp liên tục, một Jenkinsfile cũng được chúng tôi thiết kế để xây dựng hệ thống:

```
pipeline {
    agent any

    environment {
        registry = "mrpapdiamond/micro1"
        registryCredential = 'dockerhub'
        dockerImage = ''
    }

    stages {
        # Ly    d    liu    v    t   kho   cha
        stage('Clone'){
            steps{
                git branch: 'main', url:
                    'https://github.com/mrpapdiamond/notebook.git'
            }
        }
        #SAST Tool  kim  tra  cc   l    hng    bng    cch   qut   tnh
        stage('SonarTests') {
            steps{
                script{
                    docker.image('newtmitch/sonar-scanner').inside('-v
                        /var/run/docker.sock:/var/run/docker.sock
                        --entrypoint="" ') {
                        sh "/usr/local/bin/sonar-scanner
                            -Dsonar.projectKey=NodeTesting
                            -Dsonar.projectName=FirstMicroserviceProject
                            -Dsonar.projectBaseDir=/var/lib/jenkins/workspace/micropipel
                            -Dsonar.sources=/var/lib/jenkins/workspace/micropipeline
                            -Dsonar.host.url=http://192.168.213.134:9000/
                            -Dsonar.login=73f3ea5d58a441df30b520105d9954ad073920e5"
                    }
                }
            }
        }
```

```
# Kim tra cht lng cc th vin v xut file csv
stage('SCA'){
 steps{
    sh 'npm install'
    dependencyCheck additionalArguments: '-s . -o
        /var/lib/jenkins/report/notebook -f ALL --disableNodeAudit
        --disableYarnAudit --disableBundleAudit', odcInstallation:
        'SCA'
    }
}
# Xy dng nh
stage('Building Docker Image'){
    steps{
        script{
            dockerImage = docker.build registry +
                ":v1.$BUILD_NUMBER"
            dockerImageLatest = docker.build registry
        }
    }
}
# y nh ln DockerHub
stage('Deploying Docker Image to Dockerhub'){
    steps{
        script{
            docker.withRegistry('',registryCredential){
            dockerImage.push()
            dockerImageLatest.push()
            }
        }
    }
}
# Chy Docker
stage('Running Docker Image'){
    steps{
        script{
            dockerImage.run('-p 80:80 --name micro1')
```

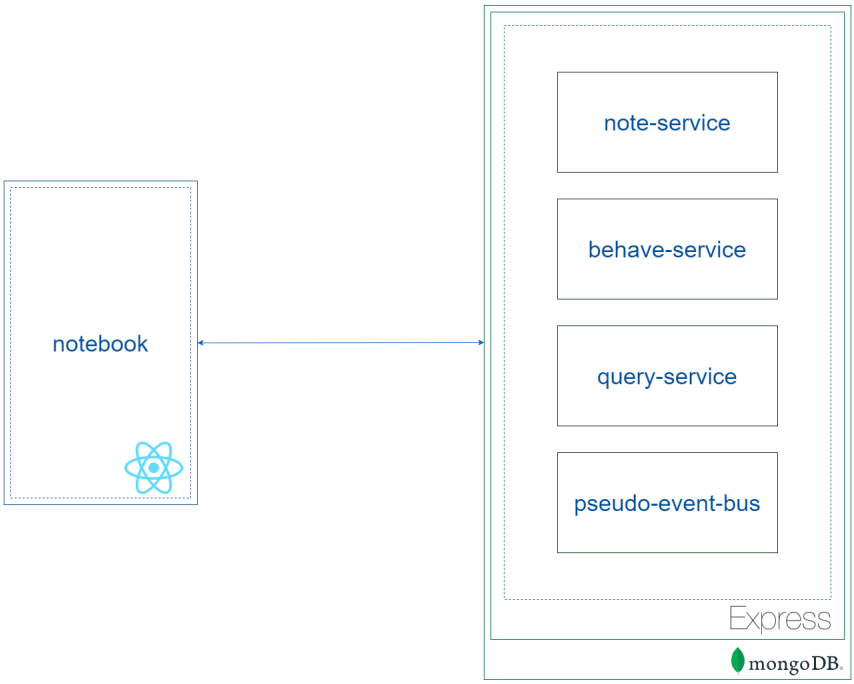
```
        }
    }
}
#DAST kim tra dch v
stage('DAST'){
    steps{
        sh 'docker run --user root -v
            /var/lib/jenkins/report:/zap/wrk/ -dt --name owasp
            owasp/zap2docker-stable /bin/bash'
        sh 'docker exec owasp zap-baseline.py -t
            http://192.168.213.134:80/ -r report.html -I'
        sh 'docker cp owasp:/zap/wrk/report.html
            /var/lib/jenkins/report/notebook/report.html'
        sh 'docker rm -f owasp'
    }
}
#Scan v xut file cvefixes
stage('CVEBundle'){
    steps{
        sh 'python3 /var/lib/jenkins/cvebundle/cvebundle.py --scan
            --path=/var/lib/jenkins/report/notebook/dependency-check-report.csv
            --name=notebook'
    }
}
stage('Cleaning up'){
    steps{
        sh "docker rm -f micro1"
        sh "docker rmi -f $registry:v1.$BUILD_NUMBER"
        sh "docker rmi -f $registry:latest"
    }
}
stage('Ansible'){
    steps{
        //ansiblePlaybook credentialsId: 'Ansible',
            disableHostKeyChecking: true, installation: 'Ansible',
            inventory: 'dev.inv', playbook: 'playbook.yaml'
```

```
                sh 'ansible-playbook -i dev.inv playbook.yaml'
            }
        }
    }
}
```

Trong Jenkinsfile, mỗi giai đoạn sẽ đảm nhiệm vai trò:

- Lấy dữ liệu về từ kho chứa
- Công cụ SAST kiểm tra các lỗ hổng bằng cách quét tĩnh
- Kiểm tra chất lượng các thư viện và xuất file cs
- Xây dựng ảnh và đẩy lên DockerHub
- Chạy ảnh và kiểm tra dịch vụ bằng công cụ quét động DAST
- Chạy công cụ CVEBundle và xuất các lỗ hổng cùng các bản vá
- Dọn dẹp ở bên môi trường kiểm thử và dùng Ansible để cấu hình các dịch vụ cho môi trường chuyển khai

4.2 Thực nghiệm qui trình CI/CD cho vi dịch vụ



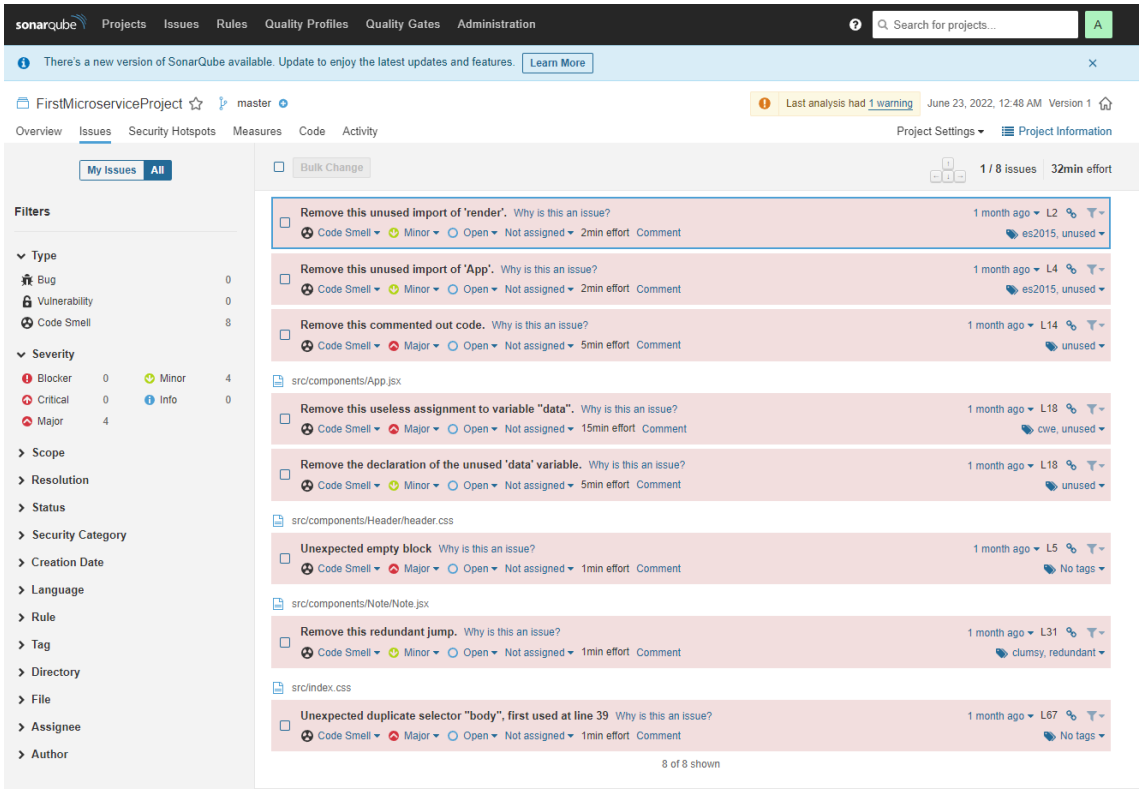
HÌNH 4.1: Cấu trúc mô hình vi dịch vụ

Vi dịch vụ được thiết kế thành năm phần với:

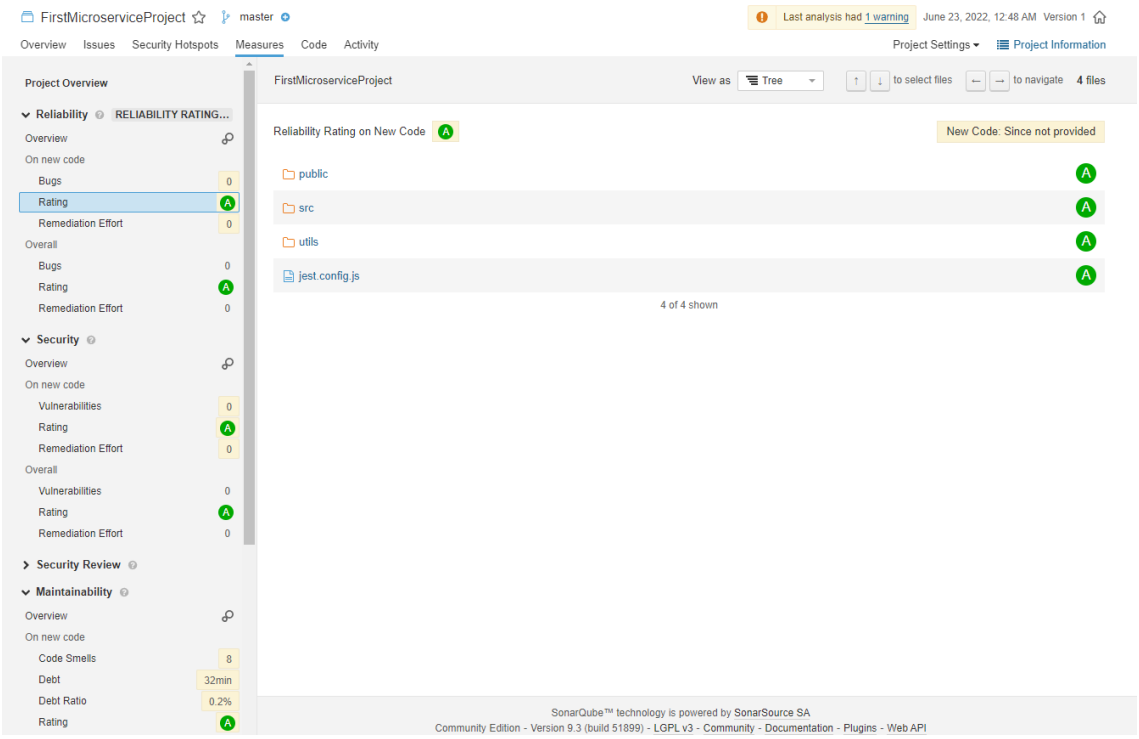
- Front-end Notebook đảm nhận vai trò giao diện người dùng
- Note-service đảm nhận vai trò ghi chú
- Behave-service đảm nhận vai trò kiểm tra trạng thái ghi chú (Lên kế hoạch, Đang thực hiện, Hoàn thành)
- Query-service đảm nhận vai trò thu thập toàn bộ dữ liệu ghi chú và trạng thái
- Event-Bus đảm nhận vai trò gửi và nhận sự kiện đến các dịch vụ

4.2.1 Giai đoạn xây dựng

Đầu tiên Jenkins sẽ lấy mã nguồn từ Github và bắt đầu qui trình CI/CD. Mã nguồn sẽ được kiểm tra bởi công cụ quét tĩnh SonarQube để kiểm tra chất lượng mã nguồn và các lỗ hổng có thể xuất hiện.

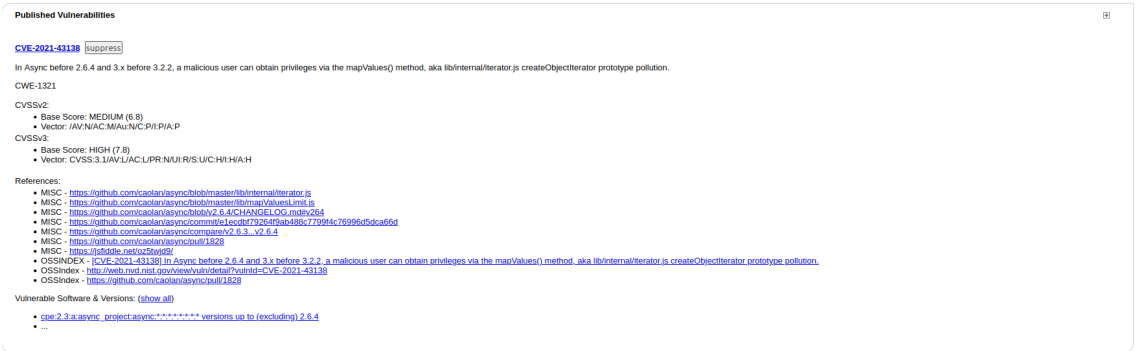


HÌNH 4.2: Mã nguồn được quét bởi SonarQube



HÌNH 4.3: SonarQube đánh giá tính bảo mật, tin cậy, và khả năng duy trì

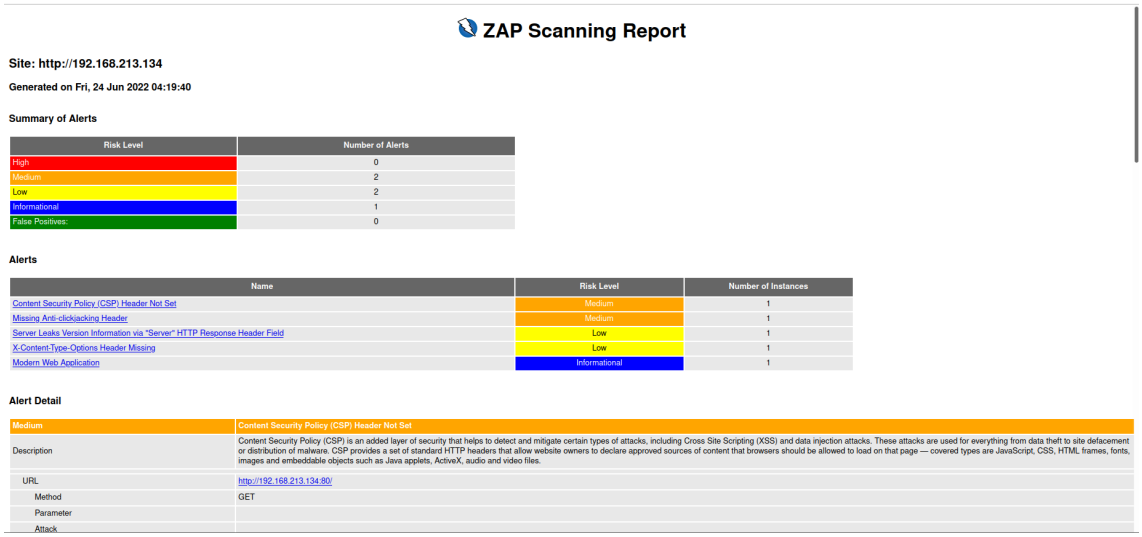
Sau khi quét, SonarQube thông báo các lỗi sai trùng lặp, các thư viện không được dùng đồng thời gợi ý sửa các lỗi này để giúp hệ thống an toàn hơn. Ngoài ra SonarQube còn đánh giá về độ bảo mật, độ tin cậy, và khả năng duy trì của mã nguồn.



HÌNH 4.4: DependencyCheck báo cáo kiểm tra thư viện và thành phần đi kèm

Sau đó, OWASP Dependency Check được kích hoạt để kiểm tra các thư viện và thành phần đi kèm để đánh giá chất lượng và xuất file csv các lỗ hổng CVE xuất hiện trong mã nguồn sau đó tích hợp vào CVEfixes trong giai đoạn sau. Bản báo cáo bao gồm các thông tin: mã định danh CVE, độ nghiêm trọng dựa theo hai phiên bản CVSSv2 và CVSSv3. Cuối cùng bản báo cáo này sẽ được công cụ CVEBundle đọc và xuất thành dữ liệu hoàn chỉnh ở giai đoạn sau.

4.2.2 Giai đoạn kiểm thử



ZAP Scanning Report

Site: http://192.168.213.134

Generated on Fri, 24 Jun 2022 04:19:40

Summary of Alerts

Risk Level	Number of Alerts
High	0
Medium	2
Low	2
Informational	1
False Positives:	0

Alerts

Name	Risk Level	Number of Instances
Content Security Policy (CSP) Header Not Set	Medium	1
Missing Anti-clickjacking Header	Medium	1
Server Leaks Version Information via "Server" HTTP Response Header Field	Low	1
X-Content-Type-Options Header Missing	Low	1
Modern Web Application	Informational	1

Alert Detail

Medium	Content Security Policy (CSP) Header Not Set
Description	Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting (XSS) and data injection attacks. These attacks are used for everything from data theft to site defacement or distribution of malware. CSP provides a set of standard HTTP headers that allow website owners to declare approved sources of content that browsers should be allowed to load on that page — covered types are JavaScript, CSS, HTML frames, fonts, images and embeddable objects such as Java applets, ActiveX, audio and video files.
URL	http://192.168.213.134:80/
Method	GET
Parameter	
Attack	

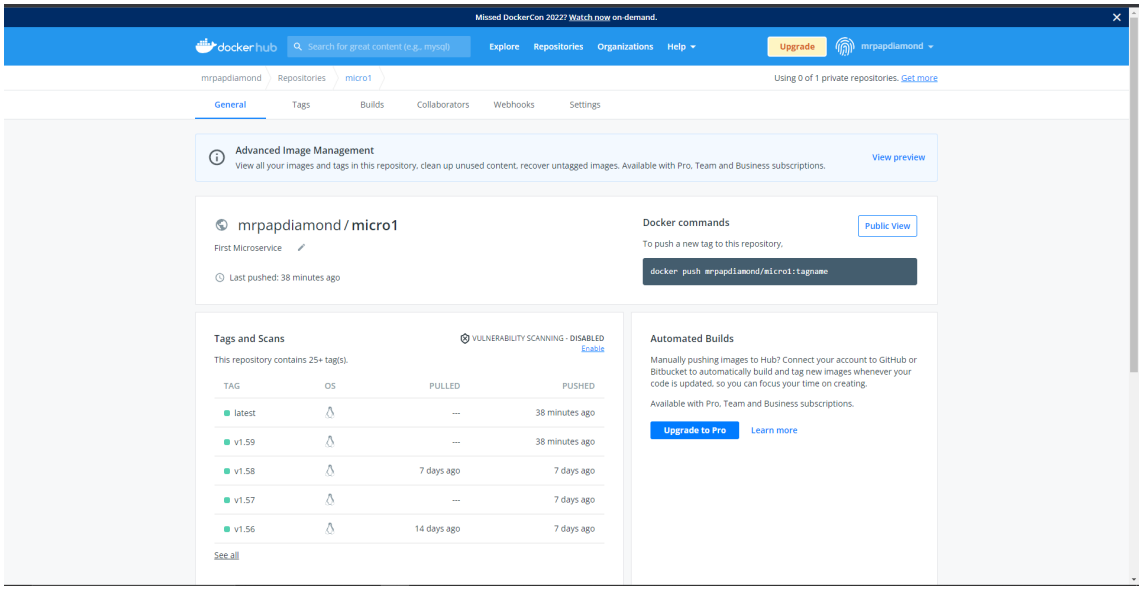
HÌNH 4.5: Báo cáo sau khi quét xong của Zap

Sau khi hoàn thành xây dựng, một ảnh ảo sẽ được tạo và chạy trên môi trường kiểm thử để kiểm tra thông qua công cụ quét động OWASP ZAP. Sau khi quét Zap sẽ cảnh báo các lỗ hổng có thể được khai thác và độ nghiêm trọng của lỗ hổng đó.

Medium	Content Security Policy (CSP) Header Not Set
Description	Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting (XSS) and data injection attacks. These attacks are used for everything from data theft to site defacement or distribution of malware. CSP provides a set of standard HTTP headers that allow website owners to declare approved sources of content that browsers should be allowed to load on that page — covered types are JavaScript, CSS, HTML frames, fonts, images and embeddable objects such as Java applets, ActiveX, audio and video files.
URL	http://192.168.213.134:80/
Method	GET
Parameter	
Attack	
Evidence	
Instances	1
Solution	Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header, to achieve optimal browser support: "Content-Security-Policy" for Chrome 25+, Firefox 23+ and Safari 7+, "X-Content-Security-Policy" for Firefox 4.0+ and Internet Explorer 10+, and "X-WebKit-CSP" for Chrome 14+ and Safari 6+.
Reference	https://developer.mozilla.org/en-US/docs/Web/Security/CSP/Introducing_Content_Security_Policy https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html http://www.w3.org/TR/CSP/ http://w3c.github.io/webappsec/specs/content-security-policy/csp-specification.dev.html http://www.html5rocks.com/en/tutorials/security/content-security-policy/ http://caniuse.com/#feat=contentsecuritypolicy http://content-security-policy.com/
CWE Id	693
WASC Id	15
Plugin Id	10038

HÌNH 4.6: Thông tin về lỗ hổng sau khi được quét

Với mỗi lỗ hổng, sẽ có các thông tin được đi kèm gồm: chú thích về lỗ hổng, giải pháp xử lí, mã định danh CWE và các bài viết liên quan đến lỗ hổng này,...



HÌNH 4.7: Ảnh được đưa lên DockerHub với hai phiên bản

Sau khi kiểm tra hoàn tất ảnh sẽ được đẩy lên DockerHub bao gồm hai phiên bản latest và phiên bản được đánh số thứ tự để phục vụ cho việc kiểm soát phiên bản, chuẩn bị chuyển giao qua môi trường phát hành.

4.2.3 Giai đoạn phát hành và triển khai

```
---
- hosts: all
```

```
#become: True
tasks:
- name: Deleting previous k8s state
  shell: kubectl delete -f /home/ubuntu/Desktop/k8s-bundle-prod.yaml
- name: Removing previous container
  shell: docker rm -f notebook
- name: Removing previous image
  shell: docker rmi -f mrpapdiamond/micro1:latest
- name: Initiating K8s
  shell: minikube start --ports=192.168.213.132:30000:30000
      --ports=192.168.213.132:30010:30010
      --ports=192.168.213.132:30020:30020
      --ports=192.168.213.132:30030:30030
- name: Apply K8s YAML file
  shell: kubectl apply -f /home/ubuntu/Desktop/k8s-bundle-prod.yaml
- name: Running docker container
  shell: docker run -p 3000:80 -d --name notebook mrpapdiamond/micro1
```

Với nội dung của playbook, Ansible sẽ thực hiện:

- Xóa các cấu hình trước đó của Kubernetes ở môi trường triển khai
- Xóa front-end đang chạy trước đó
- Tiến hành khởi động lại Kubernetes và mở các cổng liên quan đến dịch vụ
- Cấu hình các dịch vụ back-end được chạy dựa theo những thay đổi trước đó
- Khởi động lại front-end sau khi được thay đổi

Sau khi hoàn thành giai đoạn kiểm thử, ảnh trên DockerHub sẽ được Ansible kéo sang môi trường phát hành và tiến hành cấu hình dựa trên playbook. Sau đó sẽ được đồng bộ hóa với các dịch vụ trước đó thông qua Kubernetes.
Nội dung cấu hình dịch vụ của Kubernetes:

```
apiVersion: v1
kind: ConfigMap
```

```
metadata:
  name: k8s-config
data:
  mongo-url: mongo-service
  note-url: note-service
  behave-url: behave-service
  query-url: query-service
  event-bus-url: event-bus-service

---

apiVersion: v1
kind: Secret
metadata:
  name: k8s-secret
type: Opaque
data:
  DB_USERNAME: YWRtaW4=
  DB_PASSWORD: WVdSdGFXNHRjR0Z6YzNkdmNtUUs=
  DB_PATH_NOTE:
      bW9uZ29kYitzcnY6Ly9zaGVsbGRvZzpDb21wdXRlclNjaWVuY2UwQG5vdGVib29rLmd1ZXcyLm1v
  bmdvZGIubmV0L25vdGUtbW9uZ28/cmV0cnlXcml0ZXM9dHJ1ZSZ3PW1ham9yaXR5
  DB_PATH_BEHAVE:
      bW9uZ29kYitzcnY6Ly9zaGVsbGRvZzpDb21wdXRlclNjaWVuY2UwQG5vdGVib29rLmd1ZXcyLm1vbmd
  vZGIubmV0L2JlaGF2ZS1tb25nbz9yZXRyeVdyaXRlcz10cnVlJnc9bWFqb3JpdHk=
  DB_PATH_QUERY:
      bW9uZ29kYitzcnY6Ly9zaGVsbGRvZzpDb21wdXRlclNjaWVuY2UwQG5vdGVib29rLmd1ZXcyLm1v
  bmdvZGIubmV0L3F1ZXJ5LW1vbmdvP3JldHJ5V3JpdGVzPXRydWUmdz1tYWpvcml0eQ==
  DB_PATH_EVENT:
      bW9uZ29kYitzcnY6Ly9zaGVsbGRvZzpDb21wdXRlclNjaWVuY2UwQG5vdGVib29rLmd1ZXcyLm1v
  bmdvZGIubmV0L2V2ZW50LW1vbmdvP3JldHJ5V3JpdGVzPXRydWUmdz1tYWpvcml0eQ==

---

apiVersion: apps/v1
kind: Deployment
```

```
metadata:
  name: note-deployment
  labels:
    app: note
spec:
  replicas: 1
  selector:
    matchLabels:
      app: note
  template:
    metadata:
      labels:
        app: note
    spec:
      containers:
        - name: note-service
          image: mrpapdiamond/micro3:latest
          imagePullPolicy: Always
          ports:
            - containerPort: 30010
          env:
            - name: ENV
              value: "production"
            - name: ENTRY_POINT
              value: "k8s"
            - name: PORT
              value: "30010"
            - name: DB_PATH
              valueFrom:
                secretKeyRef:
                  name: k8s-secret
                  key: DB_PATH_NOTE
            - name: DB_USERNAME
              valueFrom:
                secretKeyRef:
                  name: k8s-secret
```

```
                key: DB_USERNAME
            - name: DB_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: k8s-secret
                  key: DB_PASSWORD
            - name: EVENT_BUS_ROUTE
              valueFrom:
                configMapKeyRef:
                  name: k8s-config
                  key: event-bus-url
            - name: EVENT_BUS_PORT
              value: "30000"
---
apiVersion: v1
kind: Service
metadata:
  name: note-service
spec:
  type: NodePort
  selector:
    app: note
  ports:
    - protocol: TCP
      port: 30010
      targetPort: 30010
      nodePort: 30010

---

apiVersion: apps/v1
kind: Deployment
metadata:
  name: event-bus-deployment
  labels:
    app: event-bus
```

```
spec:
  replicas: 1
  selector:
    matchLabels:
      app: event-bus
  template:
    metadata:
      labels:
        app: event-bus
    spec:
      containers:
        - name: event-bus-service
          image: mrpapdiamond/micro2:latest
          imagePullPolicy: Always
          ports:
            - containerPort: 30000
          env:
            - name: ENV
              value: "production"
            - name: ENTRY_POINT
              value: "k8s"
            - name: PORT
              value: "30000"
            - name: DB_PATH
              valueFrom:
                secretKeyRef:
                  name: k8s-secret
                  key: DB_PATH_EVENT
            - name: DB_USERNAME
              valueFrom:
                secretKeyRef:
                  name: k8s-secret
                  key: DB_USERNAME
            - name: DB_PASSWORD
              valueFrom:
                secretKeyRef:
```

```
              name: k8s-secret
              key: DB_PASSWORD
          - name: NOTE_EVENT_ROUTE
            valueFrom:
              configMapKeyRef:
                name: k8s-config
                key: note-url
          - name: NOTE_EVENT_PORT
            value: "30010"
          - name: BEHAVE_EVENT_ROUTE
            valueFrom:
              configMapKeyRef:
                name: k8s-config
                key: behave-url
          - name: BEHAVE_EVENT_PORT
            value: "30020"
          - name: QUERY_EVENT_ROUTE
            valueFrom:
              configMapKeyRef:
                name: k8s-config
                key: query-url
          - name: QUERY_EVENT_PORT
            value: "30030"
---
apiVersion: v1
kind: Service
metadata:
  name: event-bus-service
spec:
  type: NodePort
  selector:
    app: event-bus
  ports:
    - protocol: TCP
      port: 30000
      targetPort: 30000
```

```
    nodePort: 30000

---

apiVersion: apps/v1
kind: Deployment
metadata:
  name: query-deployment
  labels:
    app: query
spec:
  replicas: 1
  selector:
    matchLabels:
      app: query
  template:
    metadata:
      labels:
        app: query
    spec:
      containers:
        - name: query-service
          image: mrpapdiamond/micro5:latest
          imagePullPolicy: Always
          ports:
            - containerPort: 30030
          env:
            - name: ENV
              value: "production"
            - name: ENTRY_POINT
              value: "k8s"
            - name: PORT
              value: "30030"
            - name: DB_PATH
              valueFrom:
                secretKeyRef:
```

```
                name: k8s-secret
                key: DB_PATH_QUERY
            - name: DB_USERNAME
              valueFrom:
                secretKeyRef:
                  name: k8s-secret
                  key: DB_USERNAME
            - name: DB_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: k8s-secret
                  key: DB_PASSWORD
            - name: EVENT_BUS_ROUTE
              valueFrom:
                configMapKeyRef:
                  name: k8s-config
                  key: event-bus-url
            - name: EVENT_BUS_PORT
              value: "30000"
---
apiVersion: v1
kind: Service
metadata:
  name: query-service
spec:
  type: NodePort
  selector:
    app: query
  ports:
    - protocol: TCP
      port: 30030
      targetPort: 30030
      nodePort: 30030

---
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: behave-deployment
  labels:
    app: behave
spec:
  replicas: 1
  selector:
    matchLabels:
      app: behave
  template:
    metadata:
      labels:
        app: behave
    spec:
      containers:
        - name: behave-service
          image: mrpapdiamond/micro4:latest
          imagePullPolicy: Always
          ports:
            - containerPort: 30020
          env:
            - name: ENV
              value: "production"
            - name: ENTRY_POINT
              value: "k8s"
            - name: PORT
              value: "30020"
            - name: DB_PATH
              valueFrom:
                secretKeyRef:
                  name: k8s-secret
                  key: DB_PATH_BEHAVE
            - name: DB_USERNAME
              valueFrom:
```

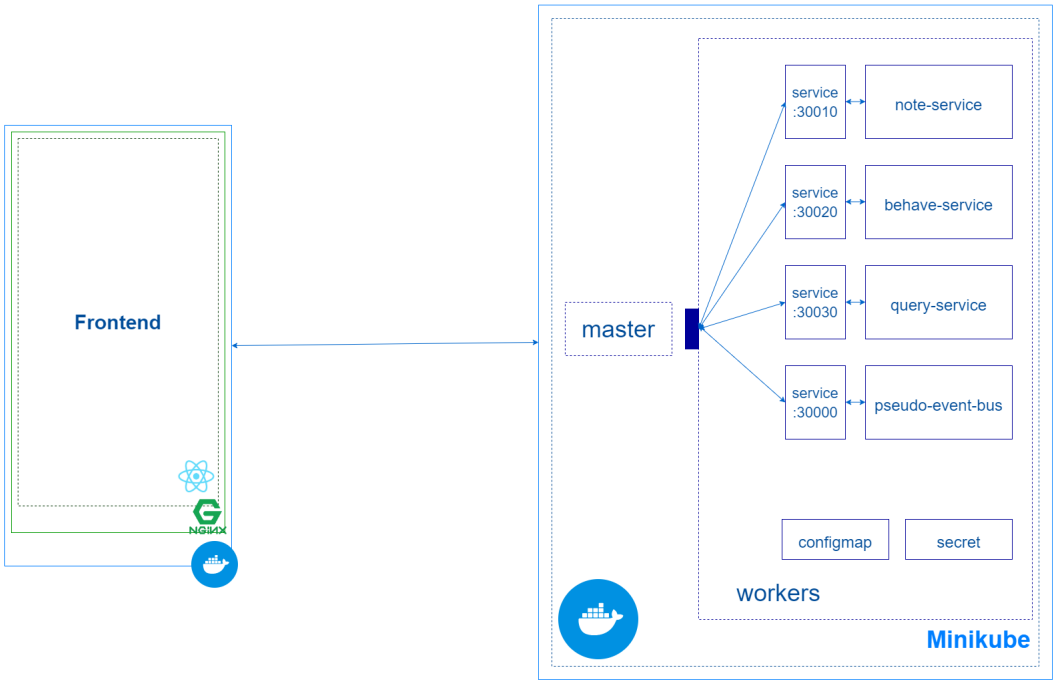
```
              secretKeyRef:
                name: k8s-secret
                key: DB_USERNAME
            - name: DB_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: k8s-secret
                  key: DB_PASSWORD
            - name: EVENT_BUS_ROUTE
              valueFrom:
                configMapKeyRef:
                  name: k8s-config
                  key: event-bus-url
            - name: EVENT_BUS_PORT
              value: "30000"
---
apiVersion: v1
kind: Service
metadata:
  name: behave-service
spec:
  type: NodePort
  selector:
    app: behave
  ports:
    - protocol: TCP
      port: 30020
      targetPort: 30020
      nodePort: 30020
```

Một dịch vụ muốn được tạo thông qua Kubernetes cần phải cấu hình qua Deployment (phụ trách thiết lập môi trường và số cổng tương ứng để có thể chạy được) và Service (phụ trách làm cầu nối kết nối giữa người dùng và dịch vụ đó):

- Deployment sẽ lấy ảnh từ DockerHub tương ứng với dịch vụ, mở cổng từ 30000 đến 30020 để chạy dịch vụ, cấu hình các biến môi trường cần thiết để

chạy dịch vụ

- Service sẽ kết nối các dịch vụ với nhau và tạo kết nối giữa người dùng đến các dịch vụ mong muốn.



HÌNH 4.8: Cấu trúc tổng thể về các dịch vụ chạy trong Kubernetes

Sau khi được cấu hình hoàn chỉnh vi dịch vụ sẽ bao gồm:

- Một front-end chạy bằng Docker với máy chủ web là Nginx
- Bốn back-end được chạy phía sau bao gồm: pseudo-event-bus (cổng 30000), note-service (cổng 30010), behave-service (cổng 30020), query-service (cổng 30020)
- configmap chứa các thông tin cấu hình các dịch vụ và secrect chứa các biến môi trường đã được mã hóa

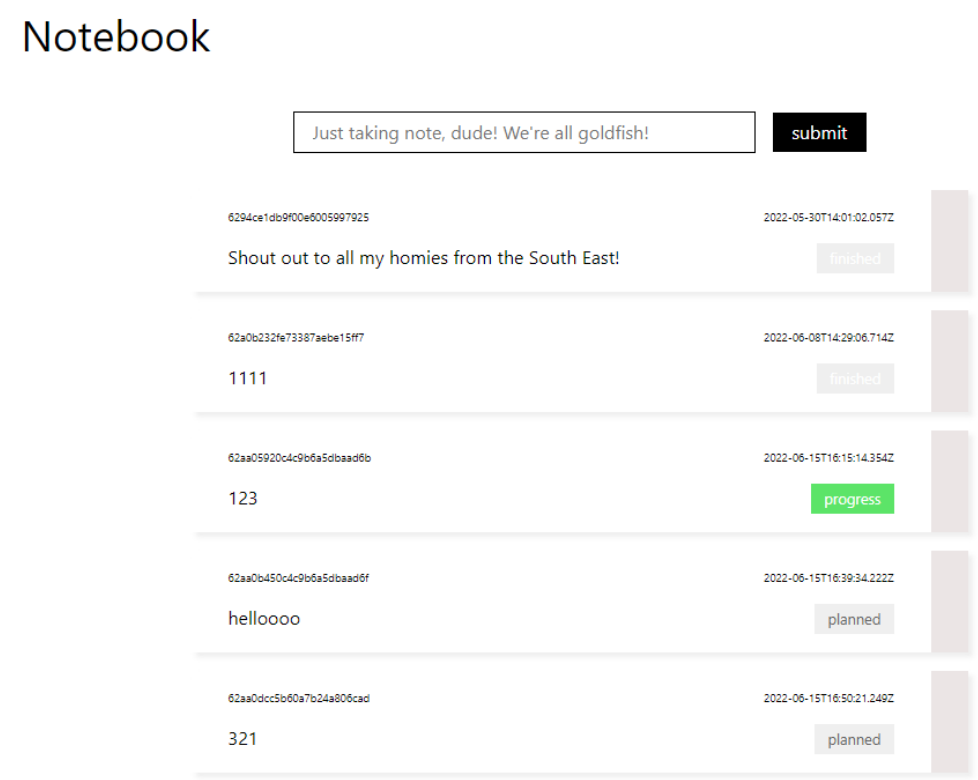
```
ubuntu@ubuntu:~/Desktop$ kubectl get all
NAME                                        READY   STATUS    RESTARTS      AGE
pod/behave-deployment-5d88bdfdcb-z29jq      1/1     Running   1 (11m ago)   47h
pod/event-bus-deployment-fc47889dc-pmvrb    1/1     Running   1 (11m ago)   47h
pod/note-deployment-66c8fcdd9d-c5252        1/1     Running   1 (11m ago)   47h
pod/query-deployment-6d5ccbdb4d-6kbjd       1/1     Running   1 (11m ago)   47h

NAME                          TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)           AGE
service/behave-service        NodePort    10.101.60.18     <none>        30020:30020/TCP   47h
service/event-bus-service     NodePort    10.103.181.235   <none>        30000:30000/TCP   47h
service/kubernetes            ClusterIP   10.96.0.1        <none>        443/TCP           17d
service/note-service          NodePort    10.108.126.17    <none>        30010:30010/TCP   47h
service/query-service         NodePort    10.101.153.254   <none>        30030:30030/TCP   47h

NAME                                    READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/behave-deployment       1/1     1            1           47h
deployment.apps/event-bus-deployment    1/1     1            1           47h
deployment.apps/note-deployment         1/1     1            1           47h
deployment.apps/query-deployment        1/1     1            1           47h

NAME                                              DESIRED   CURRENT   READY   AGE
replicaset.apps/behave-deployment-5d88bdfdcb      1         1         1       47h
replicaset.apps/event-bus-deployment-fc47889dc    1         1         1       47h
replicaset.apps/note-deployment-66c8fcdd9d        1         1         1       47h
replicaset.apps/query-deployment-6d5ccbdb4d       1         1         1       47h
ubuntu@ubuntu:~/Desktop$
```

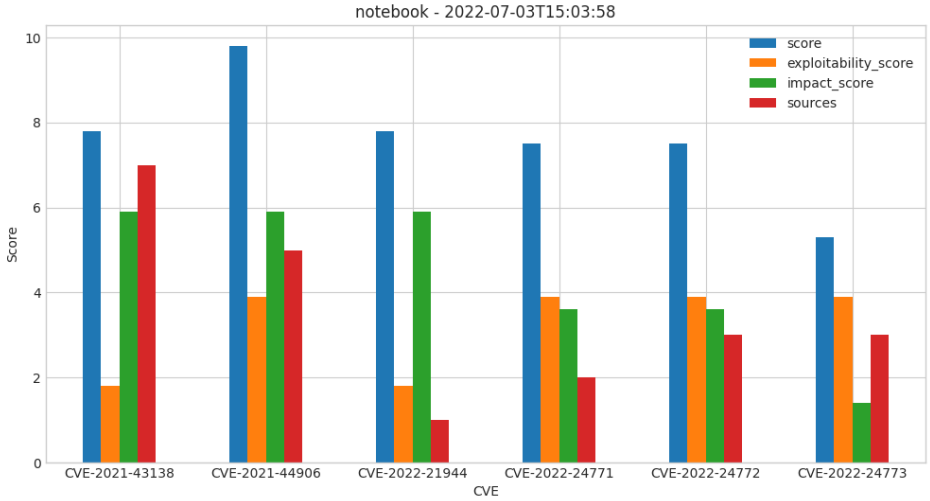
HÌNH 4.9: Các dịch vụ được chạy đồng bộ thành công



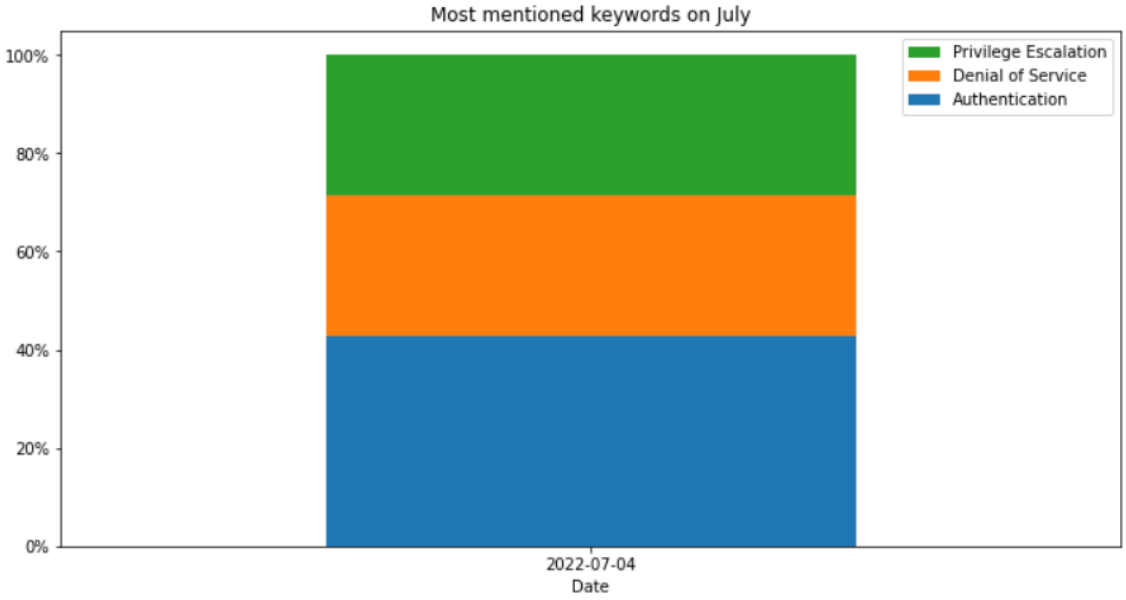
HÌNH 4.10: Kiểm tra dịch vụ chạy trên trình duyệt

Dịch vụ được chạy trên trình duyệt bao gồm:

- Các ghi chú do người dùng thực hiện được đảm nhận bởi Note-service
- Nút thanh trạng thái ghi chú được đảm nhận bởi Behave-service
- Nút bấm xóa các ghi chú được đảm nhận bởi Query-service
- Event-bus đảm nhận vai trò gửi và nhận sự kiện đến các dịch vụ khác



HÌNH 4.11: Phác thảo lỗ hổng của vi dịch vụ và độ nghiêm trọng



HÌNH 4.12: Các lỗ hổng CVE được chuyển sang keyword tương ứng

Giai đoạn này CVEBundle cũng được thực hiện để so sánh thông tin lỗ hổng của dịch vụ với cơ sở dữ liệu CVE đã được thu thập các dữ liệu về lỗ hổng. Dữ liệu sau đó được phác thảo để quan sát độ nghiêm trọng và tiến hành vá lỗi sau đó.

```
ubuntu@ubuntu:/data/jenkins/cvebundle$ sqlite3 Data/Project.db
SQLite version 3.31.1 2020-01-27 19:55:54
Enter ".help" for usage hints.
sqlite> SELECT cve_id, reference FROM project_collections;
CVE-2021-43138|https://github.com/caolan/async/blob/master/lib/mapValuesLimit.js
CVE-2022-21944|https://bugzilla.suse.com/show_bug.cgi?id=1194470
CVE-2021-44906|https://github.com/Marynk/JavaScript-vulnerability-detection/blob/main/minimist%20PoC.zip
CVE-2022-24771|https://github.com/digitalbazaar/forge/security/advisories/GHSA-cfm4-qjh2-4765
CVE-2022-24772|https://github.com/digitalbazaar/forge/commit/3f0b49a0573ef1bb7af7f5673c0cfebf00424df1
CVE-2022-24773|https://github.com/digitalbazaar/forge/commit/3f0b49a0573ef1bb7af7f5673c0cfebf00424df1
sqlite>
```

HÌNH 4.13: Cơ sở dữ liệu gợi ý các đường dẫn hướng dẫn vá lỗi

Sau khi xuất dữ liệu, các bản đề xuất vá lỗi sẽ được lưu vào cơ sở dữ liệu để hỗ trợ quá trình vá.
Bản gợi ý vá cho lỗ hổng CVE-2021-43138 được người dùng jimmywarting đề xuất trên GitHub

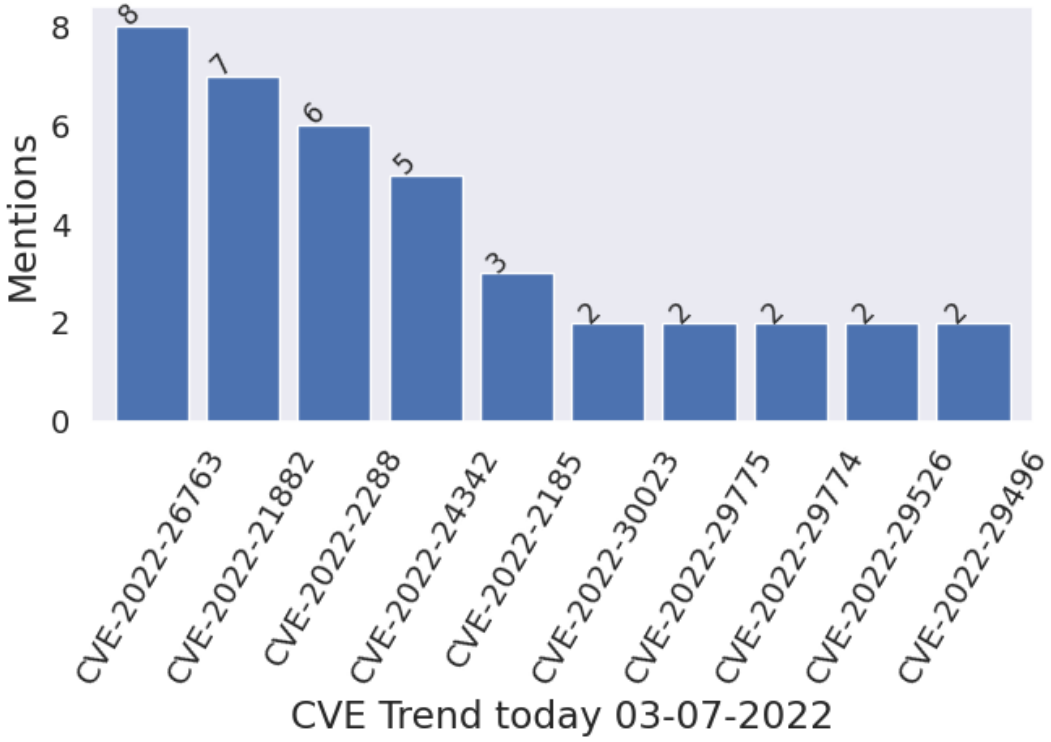
```
import eachOfLimit from './internal/eachOfLimit.js'
import awaitify from './internal/awaitify.js'
import once from './internal/once.js'
import wrapAsync from './internal/wrapAsync.js'

/**
 * The same as [`mapValues`]{@link module:Collections.mapValues} but runs
    a maximum of `limit` async operations at a
 * time.
 *
 * @name mapValuesLimit
 * @static
 * @memberOf module:Collections
 * @method
 * @see [async.mapValues]{@link module:Collections.mapValues}
 * @category Collection
 * @param {Object} obj - A collection to iterate over.
 * @param {number} limit - The maximum number of async operations at a
    time.
 * @param {AsyncFunction} iteratee - A function to apply to each value
    and key
 * in `coll`.
 * The iteratee should complete with the transformed value as its result.
```

```
 * Invoked with (value, key, callback).
 * @param {Function} [callback] - A callback which is called when all
    `iteratee`
 * functions have finished, or an error occurs. `result` is a new object
    consisting
 * of each key from `obj`, with each transformed value on the right-hand
    side.
 * Invoked with (err, result).
 * @returns {Promise} a promise, if no callback is passed
 */
function mapValuesLimit(obj, limit, iteratee, callback) {
    callback = once(callback);
    var newObj = {};
    var _iteratee = wrapAsync(iteratee)
    return eachOfLimit(limit)(obj, (val, key, next) => {
        _iteratee(val, key, (err, result) => {
            if (err) return next(err);
            newObj[key] = result;
            next(err);
        });
    }, err => callback(err, newObj));
}

export default awaitify(mapValuesLimit, 4)
```

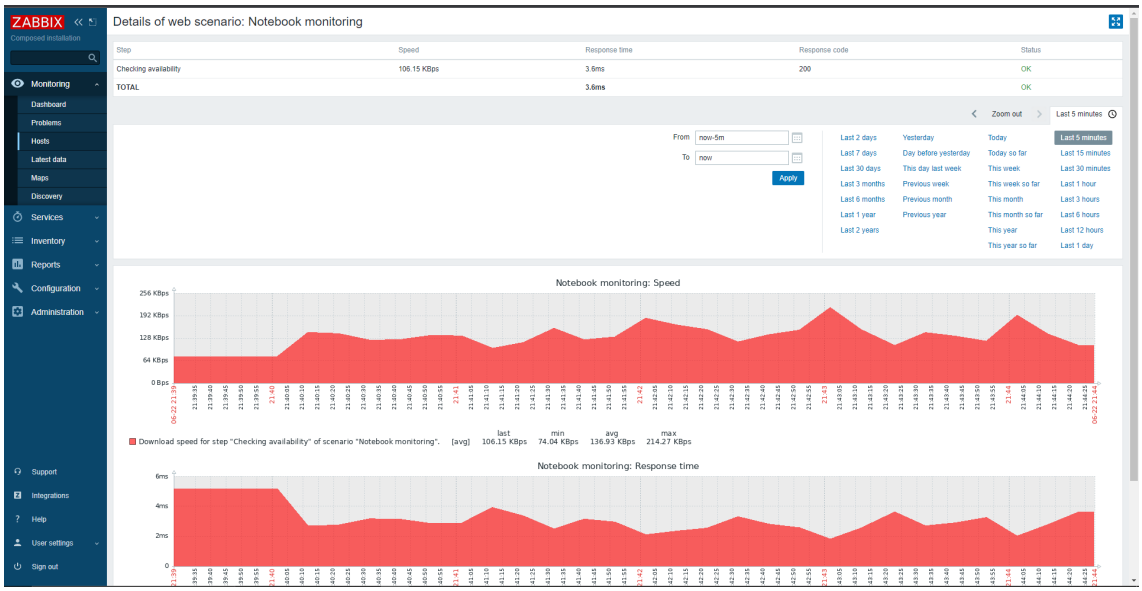
Cuối cùng, công cụ TwitterCrawler sẽ được thực hiện để thu thập các dữ liệu lỗ hổng CVE đang được bàn tán trên Twitter, thu thập đầy đủ thông tin về lỗ hổng đó trên NVD API và lưu vào cơ sở dữ liệu. Sau đó thống kê và xuất ra bản báo cáo hình ảnh.



HÌNH 4.14: Thống kê các lỗ hổng thịnh hành, đang được bàn tán trên Twitter

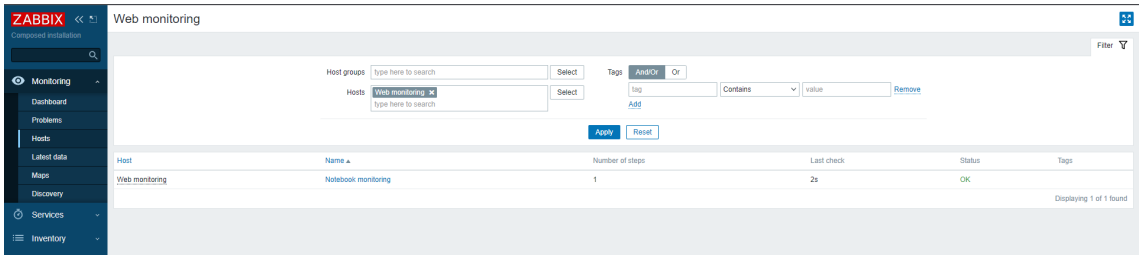
Nếu các lỗ hổng xuất hiện trên vi dịch vụ thuộc vào nhóm thịnh hành, những lỗ hổng này sẽ được liệt kê vào nhóm ưu tiên cần phải xử lí trước.

4.2.4 Giai đoạn quan sát

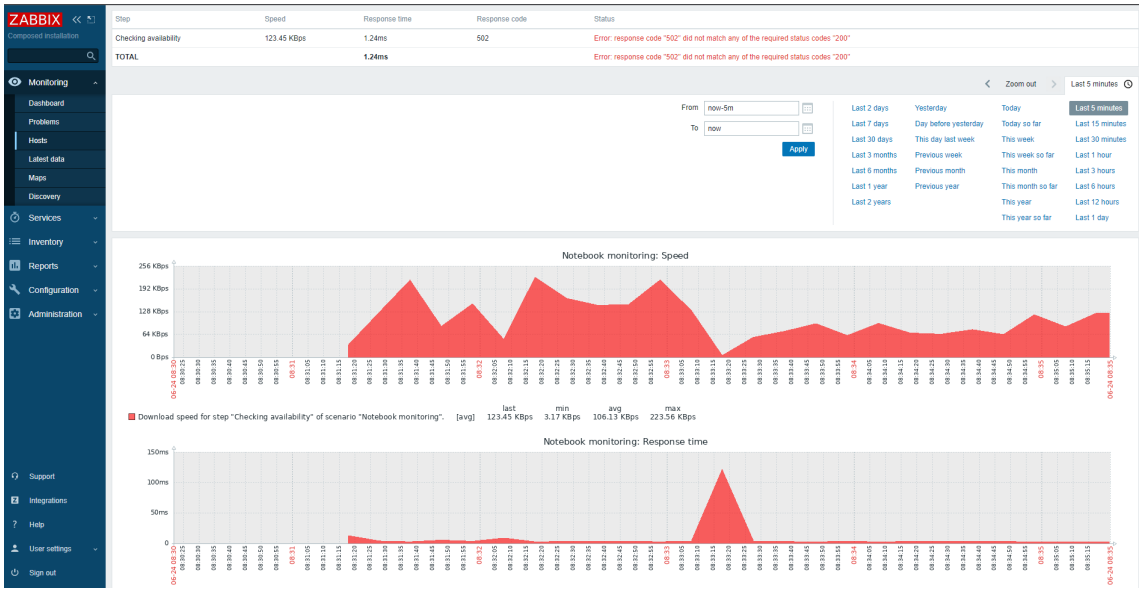


HÌNH 4.15: Quan sát ứng dụng web thông qua Zabbix, dịch vụ trả về trạng thái 200

Nhằm tăng đảm bảo tính an toàn và tối ưu tốc độ xử lí kịp thời. Ở giai đoạn quan sát Zabbix được sử dụng để đảm bảo ứng dụng web sẽ luôn được hoạt động dựa vào cơ chế gửi yêu cầu đến dịch vụ và xem phản hồi, trạng thái trả về của dịch vụ đó. Trong trường hợp xảy ra sự cố, các phiên bản ảnh cũ sẽ được sử dụng làm phiên bản dự phòng để xử lí vấn đề. Ngoài ra, tưởng lửa dành cho ứng dụng web cũng được triển khai để giảm thiểu các đợt tấn công.



HÌNH 4.16: Trạng thái trả về được Zabbix kiểm tra



HÌNH 4.17: Trường hợp dịch vụ xảy ra sự cố trả về trạng thái 502

Trong quá trình Zabbix quan sát nếu xảy ra sự cố, hệ thống sẽ được sửa chữa bằng cách tạm thời quay trở về phiên bản trước đó trong khi đội ngũ tiến hành xử lí.

HÌNH 4.18: Thử nghiệm kịch bản tấn công vào tường lửa

4.3 Đánh giá kết quả

4.3.1 Đánh giá mô hình triển khai

Kết quả được đánh giá sau quá trình được thực nghiệm trên máy ảo với cấu hình:

- Hệ điều hành: Ubuntu 20.04
- RAM: 4GB

- Ổ cứng: 25GB
- Nhân xử lí: 2 nhân

Đánh giá thời gian thực hiện có thể có sự chênh lệch giữa các đường truyền mạng và cấu hình máy tính khác nhau.
Hệ thống được thiết kế về cơ bản đã đáp ứng được các nhu cầu thiết yếu của DevSecOps như:

- Tiết kiệm thời gian và tài nguyên nhất có thể với thời gian trung bình từ 150-300 giây
- Đảm bảo tính liên tục và khả năng xử lí kịp thời khi xảy ra sự cố
- Tự động hóa toàn bộ quá trình từ thiết kế đến triển khai
- Hạn chế các lỗ hổng có thể xảy ra, dò quét và tìm ra hướng giải quyết ngay sau khi phát sinh lỗ hổng

Hạn chế của mô hình:

- Chưa thể tự động vá các lỗ hổng khi được quét
- Do hạn chế về kinh nghiệm nên các bộ luật về bảo mật chưa được áp dụng toàn vẹn
- Mô hình mới chỉ được thực nghiệm dựa vào vi dịch vụ do chúng tôi tự phát triển

Dịch vụ/Số lần thực nghiệm	1	2	3	4	5	6	7	8
Notebook (Front-end)	288.4	205.3	232.5	153.3	210.6	250.1	220.6	260.3
Behave-Service (Back-end)	163.7	111.9	171.5	177.6	143.5	163.6	176.5	111.5
Note-Service (Back-end)	129.5	113.6	165.5	164.6	168.7	92.8	105.5	120.6
Query-Service (Back-end)	181.5	164.5	163.6	132.8	150.6	160.8	160.3	151.9
Event Bus (Back-end)	201.6	158.6	187.5	168.4	181.3	196.4	178.6	165.3

BẢNG 4.1: Bảng số liệu thời gian chạy trung bình tính bằng giây

4.3.2 Đánh giá CVEfixes và Data Crawling

Bộ lưu trữ giúp cho việc theo dõi thông tin, vá các lỗ hổng trở nên nhanh và tốn ít thời gian hơn, hỗ trợ trong việc tăng tính bảo mật cho hệ thống.
Việc thu thập dữ liệu trên Twitter giúp cho hệ thống cập nhật liên tục các lỗ hổng đang thịnh hành, được bàn tán theo thời gian thực, khiến cho việc kiểm soát các lỗ hổng trở nên dễ dàng hơn. Hỗ trợ cho việc xử lí sự cố, phân chia sự ưu tiên dựa vào độ thịnh hành của lỗ hổng.

Chương 5

KẾT LUẬN

5.1 Kết quả

Do công nghệ ngày càng phát triển, các công ty và doanh nghiệp chú trọng đẩy mạnh khía cạnh bảo mật nhằm bảo đảm thông tin người dùng và dữ liệu công ty. Vì vậy, áp dụng yếu tố bảo mật vào hệ thống là điều kiện tiên quyết.
Xuyên suốt bài nghiên cứu, chúng tôi đã gợi ý mô hình DevSecOps như là giải pháp cho vấn đề trên, đồng thời gợi ý các công cụ mã nguồn mở có thể sử dụng. Đặc biệt là cách tích hợp các bản báo cáo các lỗ hổng bảo mật với cơ sở dữ liệu các lỗ hổng và bản vá của chúng để tối ưu thời gian xử lí các lỗ hổng này.
Hệ thống được xây dựng có thể tự động hóa qua từng giai đoạn, yếu tố bảo mật và chất lượng mã nguồn luôn được đảm bảo xuyên suốt. Ngoài ra chúng tôi còn gợi ý bộ công cụ mã nguồn mở có thể tích hợp với nhau phục vụ cho mục đích thiết kế hệ thống, đảm bảo tính liên tục và tiết kiệm thời gian. Bên cạnh đó, việc áp dụng kiểm soát phiên bản còn giúp cho hệ thống có thể quay lại trạng thái trước đó trong trường hợp phiên bản mới nhất xuất hiện lỗi.
Kết quả cho thấy việc áp dụng các kỹ thuật, nguyên tắc và các công cụ có thể phức tạp, nhưng khi nắm rõ cơ chế và kết hợp lại với nhau thì quá trình tự động hóa luôn đảm bảo cho hệ thống được hiệu quả, tiết kiệm và hạn chế các lỗi khi làm thủ công.
Các lỗ hổng CVE sau khi được quét sẽ được so sánh với cơ sở dữ liệu lưu trữ các thông tin và bản vá, sau đó sẽ được trích xuất và hỗ trợ lập trình viên trong việc vá lỗi và xử lí các lỗ hổng.

5.2 Hướng phát triển

Do hạn chế thời gian và trình độ môn, chúng tôi chỉ hoàn thành tích hợp việc tìm kiếm lỗ hổng và bản vá ở mức độ thiết kế ra công cụ. Chúng tôi dự định sẽ tiếp tục phát triển công cụ tích hợp này thành một ứng dụng web mà người dùng có thể tương tác dễ dàng qua các giao diện thay vì phải gõ những lệnh ở máy tính. Bên cạnh đó, tính năng cào dữ liệu Twitter để tìm các từ khóa CVE thông dụng để so sánh với lỗ hổng CVE của dịch vụ đang triển khai cũng đang được chúng tôi phát triển.

Chương 6

TÀI LIỆU THAM KHẢO

[1] Roshan N. Rajapakse, Mansooreh Zahedi, M. Ali Babar and Haifeng Shen, “Challenges and solutions when adopting DevSecOps: A systematic review”, in Information and Software Technology, 2022
[2] Kennedy A. Torkura, Muhammad I.H. Sukmana and Christoph Meinel, “Integrating Continuous Security Assessments in Microservices and Cloud Native Applications”, 2017
[3] Rakesh Kumar, Rinkaj Goyal, “Modeling continuous security: A conceptual model for automated DevSecOps using open-source software over cloud (ADOC)”, in Computer and Security, 2020
[4] Guru Bhandari, Amara Naseer and Leon Moonen, “CVEfixes: automated collection of vulnerabilities and their fixes from open-source software”, 2021
[5] Akond Ashfaque Ur Rahman, Laurie Williams, "Security practices in DevOps", 2016
[6] Rajavi Desai and T N Nisha, "Best Practices for Ensuring Security in DevOps: A Case Study Approach", 2021
[7] "DevOps là gì? Cần học gì để trở thành DevOps", https://topdev.vn/blog/devops-la-gi/
[8] "DevOps là gì? DevOps thành công nhất định phải sở hữu 6 kỹ năng và tố chất này", https://itviec.com/blog/devops-la-gi/
[9] "NATIONAL VULNERABILITY DATABASE", https://nvd.nist.gov/developers/vulnerabilities
[10] "Red Hat", https://access.redhat.com/documentation/
[11] "What is CI/CD?", https://www.redhat.com/en/topics/devops/what-is-ci-cd

[12] "DevSecOps là gì và khác thế nào với DevOps", https://itguru.vn/blog/devsecops-la-gi-va-khac-the-nao-voi-devops/
[13] "CI, CD và ... DevOps ???", https://viblo.asia/p/ci-cd-va-devops-07LKXYXDZV4
[14] "DevSecOps tương lai của an ninh bảo mật phần mềm", http://antoanthongtin.vn/giai-phap-khac/devsecops-tuong-lai-cua-an-ninh-bao-mat-phan-mem-107884
[15] "Kubernetes", https://kubernetes.io/
[16] "Docker", https://www.docker.com/
[17] Peter Putz, "Kubernetes vs Docker: What's the difference?", https://www.dynatrace.com vs-docker/