HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

Nguyễn Huy Tâm

ĐỀ CƯƠNG
ĐỀ ÁN TỐT NGHIỆP THẠC SĨ KỸ THUẬT

(Theo định hướng ứng dụng)

HÀ NỘI - 2024

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

Nguyễn Huy Tâm

Nghiên cứu giải pháp đảm bảo an toàn cho hệ thống Private Docker Registry của tổ chức

CHUYÊN NGÀNH : HỆ THỐNG THÔNG TIN

MÃ SỐ 8.48.01.04 (Hệ thống thông tin)

ĐỀ CƯƠNG ĐỀ ÁN TỐT NGHIỆP THẠC SĨ KỸ THUẬT

(Theo định hướng ứng dụng)

NGƯỜI HƯỚNG DẪN KHOA HỌC

TS. ĐINH TRƯỜNG DUY

HÀ NỘI - 2024

LỜI CAM ĐOAN

Tôi cam đoan rằng công trình nghiên cứu này là thành quả của công sức cá nhân của tôi và không sao chép từ bất kỳ nguồn nào khác. Tất cả thông tin được trình bày trong đề án này đều là sản phẩm của công việc cá nhân hoặc được tổng hợp từ nhiều nguồn tài liệu khác nhau.. Các dữ liệu và kết quả được trình bày trong đề án đều là trung thực và chưa từng được công bố trong bất kỳ công trình nghiên cứu nào khác.

Tác giả đề án

Nguyễn Huy Tâm

LỜI CẢM ƠN

Lời đầu tiên, em xin gửi lời cảm ơn chân thành và sâu sắc tới thầy TS. Đinh Trường Duy, người đã trực tiếp hướng dẫn tận tình, chu đáo, chia sẻ những ý kiến và kinh nghiệm quý báu trong suốt quá trình em thực hiện đề tài thực tập.

Sau đó, em xin gửi lời cảm ơn các thầy, cô trong Học Viện nói chung và Khoa Đào tạo Sau đại học nói riêng đã luôn nhiệt huyết, tận tình trong từng bài giảng và tạo điều kiện thuận lợi nhất cho em trong thời gian học tập và nghiên cứu tại trường Học Viện Công Nghệ Bưu Chính Viễn Thông.

Bên cạnh đó, em xin gửi lời cảm ơn tới các bạn đồng nghiệp và bạn bè đã luôn bên cạnh, hỗ trợ và động viên em trong suốt quá trình thực hiện đề tài. Những ý kiến đóng góp, chia sẻ và sự giúp đỡ của các bạn đã giúp em hoàn thiện hơn những công việc của mình.

Em cũng xin cảm ơn các nhân viên trong Học viện, những người đã làm việc âm thầm và cống hiến để tạo ra một môi trường học tập và nghiên cứu thuận lợi. Sự nhiệt tình và chu đáo của các bạn đã góp phần không nhỏ vào thành công của em.

Em hy vọng rằng những kiến thức và kinh nghiệm thu được trong thời gian thực tập này sẽ là nền tảng vững chắc cho em trong những bước đi tiếp theo của sự nghiệp.

Em xin chân thành cảm ơn!

Hà Nội, tháng 05 năm 2025

Học viên thực hiện

Nguyễn Huy Tâm

MỤC LỤC

BẢNG VIẾT TẮT VÀ THUẬT NGỮ

TỪ VIẾT TẮT VÀ THUẬT NGỮ	TIẾNG ANH	TIẾNG VIỆT
Image	Image	Hình ảnh
CI/CD	Continuous Integration and Continuous Delivery	Tích hợp liên tục/triển khai liên tục
Private	Private	Riêng tư
Public	Public	Công khai
Container	Container	Thùng chứa
Developers	Developer	Nhà phát triển
Sysadmin	Sysadmin	Quản trị hệ thống
VM	Virtual Machine	Virtual Machine: máy ảo
RBAC	Role-Based Access Control	Thiết lập kiểm soát quyền truy cập
SSL	Secure Sockets Layer	Lớp ổ cắm an toàn
LDAP	Lightweight Directory Access Protocol	Giao thức truy cập thư mục nhẹ
OAuth	Open Authorization	Mở quyền hạn
CA	Certificate Authority	Cơ quan cấp chứng chỉ
Repository	Repository	Kho Lưu trữ
HTTP	HyperText Transfer Protocol	Giao Thức Truyền Tải Siêu Văn Bản.

HTTPS	HyperText Transfer Protocol Secure	Giao Thức Truyền Tải Siêu Văn Bản Bảo Mật.
Registry	Registry	Nơi lưu trữ các container image
SSL/TLS	Secure Sockets Layer/ Transport Layer Security	Lớp Kết Nối An Toàn /Bảo mật lớp vận chuyển
SIEM	Security Information and Event Management	Quản lý sự kiện và thông tin bảo mật
CVE	Common Vulnerabilities and Exposures	Các lỗ hổng và sự phơi bày phổ biến
UI	User Interface	Giao diện người dùng
Push	Push	Đẩy
Pull	Pull	Kéo

DANH SÁCH HÌNH VẼ

DANH SÁCH BẢNG

MỞ ĐẦU

- Trong bối cảnh công nghệ phát triển nhanh chóng, việc quản lý và bảo mật các ứng dụng container trở nên ngày càng thiết yếu do container được sử dụng rộng rãi để triển khai, chạy và quản lý ứng dụng một cách hiệu quả, linh hoạt và tiết kiệm tài nguyên . Hiện nay, nhiều tổ chức và doanh nghiệp vẫn đang phụ thuộc vào các Public Registry như Docker Hub, điều này tiềm ẩn nhiều rủi ro về bảo mật và quyền riêng tư. Hơn nữa, việc thiếu kiểm soát đối với quyền truy cập và dữ liệu nhạy cảm có thể dẫn đến những vấn đề nghiêm trọng như rò rỉ thông tin, truy cập trái phép hoặc lạm dụng dữ liệu. Chính vì vậy, đề tài " Nghiên cứu giải pháp đảm bảo an toàn cho hệ thống Private Docker Registry của tổ chức" được lựa chọn nhằm cung cấp một giải pháp an toàn, hiệu quả và tùy biến cao cho việc quản lý các image Docker.

- Đề tài không chỉ mang ý nghĩa khoa học mà còn có giá trị ứng dụng thực tiễn cao, góp phần đáp ứng nhu cầu cấp thiết của các tổ chức trong việc bảo vệ dữ liệu và nâng cao hiệu suất làm việc. Tính khả thi của đề tài được thể hiện qua việc tích hợp và áp dụng các công nghệ mã nguồn mở hiện đại như Harbor, Trivy, Wazuh… để xây dựng một hệ thống Private Registry. Hệ thống này giúp tạo ra một môi trường làm việc an toàn, ổn định và hiệu quả cho các tổ chức và doanh nghiệp.

- Việc sử dụng container hóa trong phát triển phần mềm đã trở thành xu hướng toàn cầu với Docker là nền tảng phổ biến nhất nhờ khả năng đóng gói ứng dụng và các thành phần phụ thuộc vào các container nhẹ, dễ triển khai. Các Docker image (bao gồm mã nguồn, thư viện, cấu hình…) là các bản sao lưu trữ của ứng dụng cùng toàn bộ môi trường đóng vai trò trung tâm trong quá trình này. Để hỗ trợ cho quy trình này, việc lưu trữ và quản lý Docker image thông qua các Registry là rất quan trọng. Tuy nhiên, việc sử dụng các Public Registry gặp phải nhiều vấn đề về bảo mật và quản lý. Hiện nay các tổ chức hoặc doanh nghiệp đang sử dụng Public Registry để push và pull các image về chính vì vậy rủi ro về lộ thông tin và bảo mật đang là vấn đề hàng đầu.

- Những vấn đề còn tồn tại các vấn đề:

Bảo mật: Mặc dù Private Registry giúp tăng cường bảo mật, nhưng vẫn tồn tại các rủi ro như lỗ hổng trong phần mềm, tấn công vào cơ sở dữ liệu và mất mật khẩu.

Quản lý: Việc quản lý một lượng lớn image Docker có thể trở nên phức tạp, đặc biệt là khi không có một quy trình quản lý rõ ràng.

Tích hợp: Việc tích hợp Private Docker Registry với các công cụ và quy trình hiện có của tổ chức có thể gặp phải một số khó khăn.

- Đề án này sẽ tập chung vào việc triển khai Private Docker Registry sử dụng Harbor một công cụ mã nguồn mở cho các tổ chức hoặc doanh nghiệp yêu cầu bảo mật cao với dữ liệu nhạy cảm hoặc tài sản trí tuệ của tổ chức. Phát triển quy trình quản lý quyền truy cập cho nhân viên và quản lý, đánh giá hiệu quả của hệ thống nhằm cải tiến và tối ưu hóa hệ thống trong tương lai. Mục tiêu của đề án là xây dựng một Private Docker Registry sử dụng Harbor một công cụ mã nguồn mở và triển khai các giải pháp đảm bảo an toàn cho hệ thống, nhằm tạo ra một môi trường bảo mật cho việc quản lý image Docker phục vụ cho môi trường tổ chức và doanh nghiệp.

- Cụ thể, các kết quả cần đạt được bao gồm: Một hệ thống Private Registry hoạt động ổn định, bảo mật và hiệu quả, giảm thiểu rủi ro về bảo mật khi sử dụng image Docker, quản lý quyền truy cập, đánh giá hiệu quả hệ thống. Nghiên cứu sẽ được thực hiện tại các tổ chức, tập trung vào các tổ chức và doanh nghiệp yêu cầu tính bảo mật về thông tin.

-Phạm vi nghiên cứu gồm không gian là hệ thống nội bộ của tổ chức hoặc doanh nghiệp và thời gian là tập trung nghiên cứu và xây dựng hệ thống trong thời gian làm đồ án.

Để thực hiện đề tài "Nghiên cứu giải pháp đảm bảo an toàn cho hệ thống Private Docker Registry của tổ chức", đề án sẽ áp dụng một số phương pháp nghiên cứu cụ thể nhằm đảm bảo tính hiệu quả và khả thi của hệ thống. Các phương pháp này bao gồm:

- Khảo sát tài liệu: Đề án sẽ thực hiện nghiên cứu tài liệu liên quan đến các giải pháp Registry hiện có, bao gồm cả những ưu điểm và nhược điểm của

chúng. Tiếp đó là nghiên cứu các tài liệu về các biện bảo mật sau đó cấu hình các phương pháp bảo mật cho hệ thống. Cuối cùng là các tài liệu về công cụ giám sát để giám sát và nâng cao tính bảo mật nhằm xử lý các trường hợp.

- Thí nghiệm triển khai: Đề án sẽ thực hiện các thí nghiệm để triển khai hệ thống Private Docker Registry sử dụng Harbor. Các bước này sẽ bao gồm cài đặt, cấu hình và tối ưu hóa hệ thống nhằm đảm bảo tính hiệu quả và bảo mật. Ngoài ra, áp dụng các giải pháp bảo mật để quản lý quyền truy cập, mã hóa truyền dữ liệu, và tích hợp công cụ giám sát để phát hiện và xử lý sự cố kịp thời, đảm bảo an toàn và hiệu suất hệ thống.
- Công cụ và thiết bị: Để hỗ trợ cho quá trình nghiên cứu, các công cụ và thiết bị cần thiết sẽ bao gồm máy chủđể triển khai Registry, máy chủ để triển khai công cụ giám sát, máy tính cho việc phát triển và thử nghiệm

Nội dung của đề án bao gồm các phần sau:

Chương I: Tổng quan về Docker và Private Docker Registry

Trình bày tổng quan về các khái niệm liên quan cơ bản liên quan tới Docker, Container và Private Docker Registry, Docker Registry và lợi ích và vai trò của chúng trong thực tế

Chương II: Xây dựng hệ thống Private Docker Registry

Những yêu cầu phần cứng, phần mềm, khả năng bảo mật. Giới thiệu về kiến trúc của hệ thống các công cụ và các cách tổ chức và lưu trữ của các image. Đánh giá các ưu nhược điểm của các công cụ triển khai để lựa chọn

Chương III: Triển khai giải pháp đảm bảo an toàn cho hệ thống Private Docker Registry khi sử dụng Harbor

Các bước triển khai Private Docker Registry, các cấu hình, các cách kiểm tra hệ thống. Sử dụng các công cụ để phát hiện ra lỗ hổng trong image, các phương pháp giúp nâng cao bảo mật hệ thống như xác thực và mã hoá giao tiếp, và các công cụ theo dõi và giám sát hệ thống.

Chương IV: Đánh giá hiệu suất và khả năng mở rộng của Private Docker Registry khi sử dụng Harbor

Đánh giá hiệu suất của hệ thống thông qua thời gian push và pull của image trong các trường hợp khác nhau từ đó đưa ra giải pháp mở rộng để tăng tính sẵn sàng của hệ thống

CHƯƠNG I. TỔNG QUAN VỀ DOCKER VÀ PRIVATE DOCKER REGISTRY

1.1 Giới thiệu về Docker

1.1.1 Định nghĩa Docker và vai trò của Container trong phát triển phần mềm

Docker là một nền tảng mở để phát triển và chạy các ứng dụng. Docker cho phép tách biệt các ứng dụng của mình khỏi cơ sở hạ tầng để có thể phân phối phần mềm nhanh chóng. Với Docker, nó có thể quản lý cơ sở hạ tầng của mình theo cùng cách quản lý các ứng dụng. Bằng cách tận dụng các phương pháp của Docker để thử nghiệm và triển khai mã, nó có thể giảm đáng kể độ trễ giữa việc viết mã và chạy mã trong sản xuất [24]. Một số khái niệm phổ biến trong Docker:

Docker Client: Là công cụ cho phép người dùng tương tác với Docker thông qua các lệnh. Docker Client sử dụng API để gửi yêu cầu đến Docker Daemon.

Docker Daemon: Là server Docker, xử lý các yêu cầu từ API. Nó chịu trách nhiệm quản lý images, containers, networks và volumes.

Docker Volumes: Là phương pháp tối ưu để lưu trữ dữ liệu bền vững, hỗ trợ việc tạo và sử dụng ứng dụng một cách liên tục.

Docker Compose: Là công cụ giúp chạy ứng dụng với nhiều container một cách đơn giản. Thông qua file docker-compose.yml, người dùng có thể cấu hình và tái sử dụng các lệnh dễ dàng.

Docker image: là một tập hợp các tệp, thư viện, cấu hình, mã nguồn được đóng gói sẵn, dùng để tạo ra container.

Container: là phiên bản chạy được của một image. Đây là một môi trường cách ly, độc lập, chứa ứng dụng và các phụ thuộc, cho phép ứng dụng hoạt động mà không bị ảnh hưởng bởi hệ thống chủ. Container sử dụng image làm nền tảng trong quá trình hoạt động.

Vai trò của Container trong phát triển phần mềm:

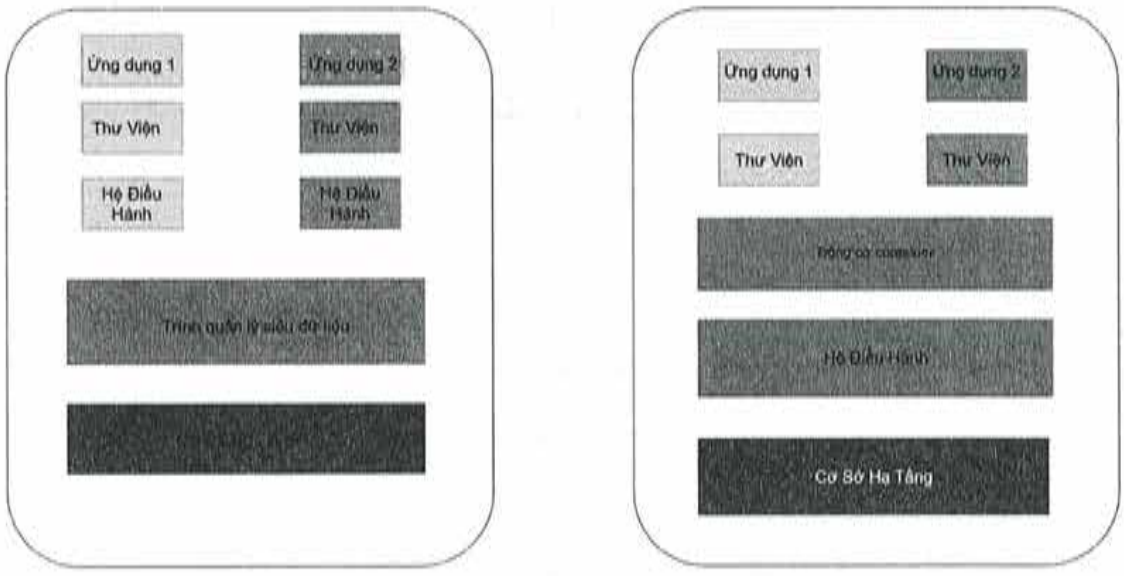
- Tính nhất quán: Containers đảm bảo ứng dụng hoạt động như nhau trên mọi môi trường, từ giai đoạn phát triển đến sản xuất, giúp loại bỏ các lỗi do sự khác biệt về môi trường.

- Tăng tốc phát triển: Containers cho phép các nhà phát triển nhanh chóng xây dựng, kiểm thử và triển khai các ứng dụng mà không cần lo lắng về việc thiết lập môi trường phức tạp.

- Tối ưu hóa tài nguyên: Các container chia sẻ tài nguyên hệ thống một cách hiệu quả, giúp giảm thiểu chi phí và tận dụng tối đa tài nguyên có sẵn.

1.1.2 So sánh Docker container và Virtual Machine

Cả Virtual Machine (máy ảo) và Container (thùng chứa) đều là những công nghệ ảo hóa giúp tối ưu hóa việc sử dụng tài nguyên phần cứng và phần mềm. Chúng thường được sử dụng trong quá trình triển khai ứng dụng, đặc biệt là ở giai đoạn chuẩn bị môi trường chạy mã nguồn cho người dùng cuối trong vòng đời phát triển phần mềm. Tuy nhiên, giữa VM và container Docker vẫn tồn tại một số điểm khác biệt quan trọng. Việc hiểu rõ những điểm khác biệt này sẽ giúp đề án đưa ra giải pháp phù hợp.



Hình 1.1 Kiến trúc của máy ảo (VM) và Container

Máy ảo (VM) là các hệ thống máy tính ảo được tạo ra bởi phần mềm ảo hóa, chạy tách biệt trên cùng một phần cứng vật lý hoặc một cụm máy chủ. Mỗi máy ảo hoạt động như một hệ thống độc lập, có đầy đủ các thành phần phần cứng ảo riêng như CPU, bộ nhớ, lưu trữ và card mạng [27].

Trong khi đó, container không ảo hóa toàn bộ máy tính mà chỉ ảo hóa ở cấp độ hệ điều hành. Các container chạy trực tiếp trên máy chủ vật lý, chia sẻ chung nhân (kernel) của hệ điều hành, và thường dùng chung các tệp nhị phân hoặc thư viện. Nhờ đó, container sử dụng tài nguyên nhẹ hơn và hiệu quả hơn so với máy ảo.

Bảng 1.1 Bảng so sánh giữa Máy ảo và container

Tính năng	Máy ảo	Container
Kích cỡ	Nặng	Nhẹ
Hiệu suất	Hiệu suất hạn chế	Hiệu suất gốc
Ảo hoá	Mỗi VM chạy trên hệ điều hành riêng	Tất cả container đều chia sẻ hệ điều hành với máy chủ
Áo hoá	Ảo hoá ở cấp độ phần cứng	Ảo hoá hệ điều hành
Thời gian	Thời gian khởi động lâu	Thời gian khởi động nhanh
Bảo mật	Tương đối an toàn	Ít an toàn hơn

1.1.3 Lợi ích của Docker trong việc triển khai ứng dụng

Việc sử dụng Docker mang lại nhiều lợi ích, giúp quy trình phát triển và triển khai ứng dụng hiệu quả, linh hoạt. Docker cho phép khởi động và dừng container chỉ trong vài giây, giúp tiết kiệm thời gian và tăng tốc thử nghiệm, triển khai. Người dùng có thể dễ dàng chạy container trên mọi hệ thống, từ máy cá nhân đến máy chủ đám mây, đảm bảo môi trường làm việc đồng nhất dù ở bất kỳ nền tảng nào.

Docker cũng đơn giản hóa việc thiết lập môi trường làm việc, chỉ cần cấu hình một lần và tái sử dụng cho các dự án hoặc thành viên mới. Điều này giảm thiểu sai sót, rút ngắn thời gian làm quen và tối ưu quy trình làm việc. Ngoài ra, Docker đảm bảo cách ly các container, giúp giữ môi trường sạch sẽ, bảo mật, và tránh ảnh hưởng lẫn nhau khi dừng hoặc xóa container.

Đảm bảo rằng sự cố trong một container không ảnh hưởng đến các container khác hoặc hệ thống máy chủ. Docker cung cấp các tính năng bảo mật như quản lý quyền truy cập, ký số image, quét lỗ hổng bảo mật.

1.1.4 Các rủi ro và hậu quả tiềm ẩn của việc triển khai Docker không an toàn

Việc triển khai Docker mà không chú trọng đầy đủ đến bảo mật có thể dẫn đến nhiều rủi ro và hệ quả nghiêm trọng. Một trong những rủi ro chính là việc sử dụng các image Docker dễ bị tấn công. Những image lỗi thời hoặc không đáng tin cậy có thể chứa các lỗ hổng đã biết, tạo cơ hội cho kẻ tấn công xâm nhập vào hệ thống. Do đó, việc quét thường xuyên các image container để phát hiện lỗ hổng và cập nhật chúng là rất cần thiết.

Một rủi ro nghiêm trọng khác là thoát khỏi container, nơi kẻ tấn công có thể chiếm quyền kiểm soát của container sau đó rời khỏi môi trường container và truy cập vào hệ thống máy chủ của container đó hoặc các container khác trên cùng một máy chủ để thực hiện các hoạt động trái phép. Điều này có thể cho phép kẻ tấn công giành quyền kiểm soát hệ thống máy chủ hoặc truy cập vào dữ liệu nhạy cảm trong các container khác.

Cấu hình mạng không chính xác cũng có thể tạo ra các lỗ hổng bảo mật. Những cấu hình này có thể dẫn đến việc các dịch vụ bị lộ ra ngoài ý muốn, tạo cơ hội cho kẻ tấn công di chuyển ngang giữa các container. Việc vô tình hoặc cố ý làm lộ thông tin nhạy cảm và các biến môi trường cũng có thể bị kẻ tấn công khai thác để truy cập trái phép vào các hệ thống hoặc dữ liệu quan trọng.

Tóm lại, việc triển khai Docker mà không có các biện pháp bảo mật đầy đủ có thể dẫn đến một loạt các rủi ro như việc khai thác các lỗ hổng các image đã biết, không phân đúng quyền cho người dùng đến các cuộc tấn công nghe lén trung gian khi hệ thống chưa được mã hoá giao tiếp khiến lộ thông tin cá nhân của người dùng, gây ra những hậu quả nghiêm trọng cho hoạt động kinh doanh và uy tín của tổ chức.

1.1.5 Một số cách tấn công thường gặp trên hệ thống Docker

- Tấn công Man-in-the-Middle (MITM):

+ *Cơ Chế Tấn Công*: Xảy ra khi kẻ tấn công bí mật chèn vào giữa quá trình giao tiếp của máy khách và Docker registry, cho phép chúng chặn, nghe lén và thậm chí sửa đổi dữ liệu đang được truyền đi. Kỹ thuật này tạo điều kiện cho kẻ tấn công can thiệp vào quá trình phân phối image, có khả năng dẫn đến việc triển khai các ứng dụng chứa mã độc hoặc đánh cắp thông tin nhạy cảm. Việc thiếu các kết nối an toàn và cơ chế xác thực mạnh mẽ sẽ biến các registry không được bảo vệ thành mục tiêu dễ dàng cho các cuộc tấn công MITM [28].

+ *Kỹ thuật*: Chặn lưu lượng truy cập: Kẻ tấn công có thể sử dụng các công cụ như Wireshark hoặc tcpdump để thu thập các gói mạng được truyền giữa máy khách và registry. Nếu kết nối không được mã hóa bằng HTTPS, nội dung của các gói này, bao gồm cả thông tin nhạy cảm, có thể được xem ở dạng văn bản thuần túy.

- Lộ thông tin nhạy cảm trong Docker Image:

+ *Các Nguồn Rò Rỉ Phổ Biến*: Lộ thông tin nhạy cảm trong Docker images là một vấn đề nghiêm trọng với nhiều nguồn rò rỉ phổ biến. Một trong những nguyên nhân chính là việc các nhà phát triển vô tình nhúng thông tin nhạy cảm như mật khẩu, khóa API và khóa SSH trực tiếp vào mã ứng dụng hoặc tệp cấu hình bên trong image, thường do sự bất cẩn hoặc thiếu nhận thức về các rủi ro bảo mật. Thêm vào đó, thông tin nhạy cảm cũng có thể bị rò rỉ qua các biến môi trường được định nghĩa trong Dockerfile hoặc trong quá trình

chạy container. Mặc dù biến môi trường có thể hữu ích cho việc cấu hình, việc lưu trữ bí mật ở đây có thể khiến chúng dễ bị truy cập bởi các quy trình không được ủy quyền. Hơn nữa, ngay cả khi các bí mật được xóa trong một layer sau của image, chúng vẫn có thể tồn tại trong các layer trước đó đã được lưu trong bộ nhớ cache do kiến trúc layer của Docker. Ngoài ra, các tệp (.env), Dockerfile và tệp thông tin đăng nhập ứng dụng cũng là những nguồn phổ biến gây rò rỉ thông tin nhạy cảm trong các registry bị lộ. Một nghiên cứu đã chỉ ra rằng một tỷ lệ đáng kể các image trong Docker Hub chứa các bí mật như khóa riêng và khóa API [5].

+ *Kỹ thuật tấn công*: trong Docker việc phân tích layer image, trong đó kẻ tấn công có thể pull docker image và kiểm tra từng layer để tìm kiếm các bí mật bị lộ. Các công cụ như dive [6] cho phép kiểm tra chi tiết nội dung của từng layer, giúp xác định các tệp hoặc cấu hình chứa thông tin nhạy cảm. Ngay cả khi một tệp chứa bí mật đã bị xóa ở layer cuối cùng, nội dung của nó vẫn có thể tồn tại ở một layer trung gian. Bên cạnh đó, kẻ tấn công có quyền truy cập vào một container đang chạy có thể kiểm tra các biến môi trường để tìm kiếm các bí mật lưu trữ ở đó. Việc này có thể được thực hiện bằng các lệnh như *docker inspect* hoặc bằng cách truy cập vào shell của container [7].

1.2 Tổng quan về Private Docker Registry

1.2.1 Khái niệm Private Docker Registry

Private registries trong Docker là một giải pháp mạnh mẽ giúp đơn giản hóa quá trình lưu trữ, quản lý và chia sẻ các Docker images trong một môi trường bảo mật. Bằng cách sử dụng private registries, các nhà phát triển có thể đảm bảo rằng ứng dụng của họ sẽ hoạt động nhất quán và an toàn trên nhiều môi trường khác nhau. Việc thiết lập và quản lý private registries khá đơn giản và có thể thực hiện thông qua một vài bước cơ bản. Bằng cách tận dụng các tính năng của private registries, các nhà phát triển và doanh nghiệp có thể nâng cao hiệu quả và chất lượng của quy trình phát triển phần mềm.

1.2.2 Vai trò của Private Docker Registry trong quản lý và phân phối Container Image

Private Docker Registry đóng vai trò thiết yếu trong việc quản lý và phân phối container images, đặc biệt trong môi trường doanh nghiệp hoặc khi cần bảo mật thông tin nhạy cảm. Một trong những lợi ích lớn nhất của việc sử dụng Private Docker Registry là khả năng bảo mật. Private Registry cho phép doanh nghiệp kiểm soát quyền truy cập vào các images của mình, đảm bảo rằng chỉ những người dùng hoặc nhóm được ủy quyền mới có thể thực hiện các thao tác tải lên hoặc tải xuống [30]. Điều này không chỉ bảo vệ mã nguồn và dữ liệu nhạy cảm mà còn giúp ngăn chặn các mối đe dọa từ bên ngoài. Hơn nữa, nhiều private registries hỗ trợ mã hóa, giúp bảo vệ dữ liệu trong quá trình truyền tải và lưu trữ, từ đó tăng cường tính bảo mật cho các images.

Private Docker Registry cũng giúp tăng hiệu suất trong quy trình phát triển. Khi sử dụng một registry nội bộ, việc tải lên và tải xuống images thường nhanh hơn so với việc truy cập các public registry, nhờ vào việc lưu trữ gần gũi hơn với môi trường phát triển. Điều này không chỉ giúp giảm độ trễ mà còn cải thiện tốc độ làm việc của các nhà phát triển, đặc biệt trong những môi trường có băng thông hạn chế hoặc không ổn định [32].

1.2.3 So sánh giữa Public Docker Registry và Private Docker Registry

Public Registries: Các public registries như Docker Hub cho phép bất kỳ ai cũng có thể truy cập và sử dụng các Docker images được lưu trữ ở đó. Điều này rất hữu ích cho việc chia sẻ các images với cộng đồng, nhưng không phù hợp cho các ứng dụng chứa mã nguồn hoặc dữ liệu nhạy cảm.

Private Registries: Các private registries cho phép lưu trữ các Docker images trong một môi trường riêng tư, chỉ có những người được ủy quyền mới có thể truy cập. Điều này rất quan trọng cho các doanh nghiệp và các dự án yêu cầu bảo mật cao.

Bảng 1.2 Bảng so sánh giữa private registry và public registry

Tiêu chí	Private Docker Registry	Docker Private Registry
Quyền truy cập	Kiểm soát truy cập	Mọi người đều có thể truy cập
Bảo mật	Bảo mật cao với mã hoá và kiểm soát truy cập	An toàn hạn chế, có lỗ hổng bảo mật
Tốc độ tải lên/xuống	Tốc độ nhanh hơn	Tốc độ chậm, phụ thuộc vào tình trạng mạng
Chi phí	Yêu cầu về đầu tư cơ sở hạ tầng và bảo trì	Thường miễn phí hoặc chi phí thấp
Lưu trữ	Lưu trữ tập trung, dễ quản lý và dọn dẹp	Lưu trữ phân tán, có thể khó quản lý
Tích hợp	Dễ dàng tích hợp vào quy trình CI/CD	Có hỗ trợ nhưng không linh hoạt bằng

1.2.4 Ứng dụng thực tế của Private Docker Registry trong môi trường doanh nghiệp

Private Docker Registry đóng vai trò quan trọng trong môi trường doanh nghiệp, mang lại nhiều ứng dụng thực tế giúp tối ưu hóa quy trình phát triển, bảo mật và quản lý images. Đối với nhiều doanh nghiệp, phần mềm độc quyền, các ứng dụng tùy chỉnh và công cụ nội bộ của họ đại diện cho một nguồn lợi thế cạnh tranh đáng kể và thể hiện tài sản trí tuệ có giá trị. Việc lưu trữ các tài sản quan trọng này trên các public registry vốn dĩ mang theo rủi ro tiếp xúc, truy cập trái phép hoặc thậm chí đánh cắp tài sản trí tuệ, điều này có thể làm suy yếu nghiêm trọng vị thế thị trường của công ty. Một Private Docker Registry trực tiếp giải quyết vấn đề này bằng cách tạo ra một môi trường an toàn, cô lập và được kiểm soát chặt chẽ [30].

Private Docker Registry là một công cụ thiết yếu giúp doanh nghiệp tăng cường quản lý container images. Một trong những lợi ích quan trọng nhất là các doanh

nghiệp thường phải đối mặt với các yêu cầu quy định nghiêm ngặt đòi hỏi kiểm soát và khả năng truy xuất nguồn gốc hoàn toàn đối với các tạo phẩm phần mềm của họ. Một private registry đảm bảo tuân thủ các chính sách quản trị dữ liệu và yêu cầu kiểm toán khác nhau bằng cách cung cấp một môi trường được kiểm soát để lưu trữ và phân phối image [31].

1.2.5 Một số ví dụ tấn công phổ biến liên quan đến Docker container

Các nhóm hacker như TeamTNT, WatchDog, Kinsing và Rocke đã trở thành mối đe dọa nghiêm trọng đối với môi trường Docker bằng cách khai thác các cấu hình API không an toàn. Chúng tận dụng các Docker API được mở công khai hoặc không được bảo mật đúng cách để triển khai các container độc hại. Những container này thường được sử dụng để thực hiện các cuộc tấn công từ chối dịch vụ phân tán (DDoS) hoặc khai thác tài nguyên hệ thống như CPU và GPU để đào tiền điện tử trái phép [2].

TeamTNT, một trong những nhóm nổi bật, không chỉ triển khai mã độc để đào tiền điện tử mà còn tìm cách đánh cắp thông tin đăng nhập AWS từ các container bị xâm nhập, mở rộng khả năng tấn công trên hạ tầng đám mây. Kinsing lại tập trung khai thác các lỗ hổng phần mềm và mở rộng quyền truy cập để chiếm quyền kiểm soát toàn bộ máy chủ. Rocke và WatchDog cũng sử dụng các kỹ thuật tương tự, nhưng với trọng tâm tối ưu hóa việc khai thác tài nguyên hệ thống cho hoạt động đào tiền mã hóa. Các cuộc tấn công này không chỉ tiêu tốn tài nguyên, gây thiệt hại tài chính mà còn làm tăng nguy cơ rò rỉ dữ liệu nhạy cảm, ảnh hưởng nghiêm trọng đến tính bảo mật và ổn định của hệ thống bị tấn công.

Các sự cố bảo mật thực tế đã cho thấy những hậu quả nghiêm trọng của việc bảo mật Docker yếu kém. Vụ xâm nhập cụm Kubernetes của Tesla vào năm 2018 là một ví dụ điển hình, khi kẻ tấn công đã giành được quyền truy cập thông qua một console không được bảo vệ và khai thác cụm này để đào tiền điện tử. Vụ rò rỉ thông tin đăng nhập Docker Hub vào năm 2019 cũng cho thấy những rủi ro liên quan đến việc sử dụng các repository của bên thứ ba mà không có các biện pháp bảo mật đầy đủ, khi thông tin nhạy cảm của người dùng và access token đã bị lộ [3].

Các nghiên cứu đã cho thấy rằng một tỷ lệ đáng kể các Docker Images chính thức trên Docker Hub tồn tại các lỗ hổng bảo mật nghiêm trọng. Trung bình, mỗi image chứa nhiều lỗ hổng, chủ yếu bắt nguồn từ việc sử dụng các gói phần mềm đã lỗi thời hoặc không được cập nhật [4].

1.3 Kết luận chương I

Docker và công nghệ container đã trở thành nền tảng quan trọng trong phát triển, triển khai và vận hành ứng dụng hiện đại. Việc sử dụng Docker mang lại nhiều lợi ích vượt trội như đảm bảo tính nhất quán môi trường, tăng tốc quy trình phát triển, tối ưu hóa tài nguyên, dễ dàng mở rộng và tích hợp với các hệ thống tự động hóa CI/CD. So với máy ảo truyền thống, Docker container có kích thước nhẹ hơn, khởi động nhanh hơn và hiệu suất gần sát với hệ điều hành gốc, nhờ vào việc chia sẻ nhân hệ điều hành và tài nguyên một cách hiệu quả.

Private Docker Registry đóng vai trò then chốt trong việc quản lý, phân phối container image một cách an toàn, kiểm soát quyền truy cập và tăng hiệu suất truyền tải. Dù vậy, nếu triển khai Docker không an toàn, hệ thống có thể đối mặt với nhiều rủi ro như lộ thông tin nhạy cảm, tấn công từ chối dịch vụ, khai thác lỗ hổng trong image hoặc API quản trị không được bảo vệ. Thực tế đã ghi nhận nhiều vụ tấn công thành công vào hệ sinh thái Docker do bảo mật yếu kém, gây thiệt hại nghiêm trọng về tài nguyên, dữ liệu và uy tín doanh nghiệp.

Tóm lại, Docker và Private Docker Registry đem lại nhiều lợi ích thiết thực cho phát triển phần mềm hiện đại nhưng đòi hỏi phải được triển khai với các biện pháp bảo mật nghiêm ngặt. Việc nhận diện, đánh giá và chủ động phòng ngừa các rủi ro bảo mật là yếu tố then chốt giúp doanh nghiệp tận dụng tối đa tính năng của Docker, đồng thời đảm bảo an toàn cho hệ thống và dữ liệu trong môi trường sản xuất thực tế.

CHƯƠNG II. XÂY DỰNG HỆ THỐNG PRIVATE DOCKER REGISTRY

2.1 Các công cụ triển khai Private Docker Registry

2.1.1 Docker Registry

Docker Registry là hệ thống lưu trữ và phân phối cho các image Docker. Cùng một image có thể có nhiều phiên bản khác nhau. Nó tương tự như một kho lưu trữ, nơi nó có thể lưu trữ, quản lý và chia sẻ các image Docker của mình, cả public và private [26]. Khi triển khai Docker Registry sẽ tạo ra một kho chứa image riêng, cho phép lưu trữ và phân phối các image Docker mà không cần phụ thuộc vào các dịch vụ công cộng. Điều này không chỉ đảm bảo an toàn cho tài sản mã nguồn mà còn giúp tối ưu hóa quy trình phát triển phần mềm.

Với Docker Registry, người dùng có thể thiết lập một môi trường kiểm soát hoàn toàn, cho phép chỉ những người dùng được ủy quyền mới có quyền truy cập và quản lý các image. Việc cài đặt và cấu hình Docker Registry cũng rất đơn giản, chỉ cần vài lệnh Docker là đã có thể khởi động một Registry hoạt động.

2.1.2 Harbor

Harbor là một giải pháp lưu trữ image Docker mạnh mẽ và toàn diện, được thiết kế để cung cấp một Private Registry với nhiều tính năng bảo mật và quản lý tiên tiến. Khi triển khai Harbor, tổ chức có thể dễ dàng tạo và quản lý một Private Docker Registry, giúp bảo vệ tài sản mã nguồn và dữ liệu nhạy cảm khỏi các mối đe dọa bên ngoài. Harbor không chỉ giúp tổ chức duy trì an toàn cho các image Docker mà còn tối ưu hóa quy trình phát triển phần mềm [15].

Harbor còn sở hữu tính năng quét lỗ hổng bảo mật tích hợp, giúp phát hiện và thông báo về các lỗ hổng trong các image Docker ngay khi chúng được tải lên. Tính năng này cực kỳ quan trọng trong việc đảm bảo rằng các ứng dụng được triển khai luôn đáp ứng các tiêu chuẩn bảo mật cao nhất.

Ngoài ra, Harbor hỗ trợ nhiều loại lưu trữ backend, cho phép tùy chọn giữa lưu trữ cục bộ hoặc đám mây, từ đó giúp dễ dàng mở rộng khi nhu cầu tăng lên.

Giao diện người dùng thân thiện của Harbor giúp quản lý và theo dõi các image Docker trở nên dễ dàng hơn, đồng thời cung cấp các báo cáo và phân tích chi tiết về việc sử dụng image.

2.1.3 Đánh giá ưu nhược điểm và lựa chọn công cụ triển khai

Việc lựa chọn giữa Harbor và Docker Registry cho Private Registry phụ thuộc vào nhu cầu. Dưới đây là bảng so sánh giữa hai công cụ.

Bảng 2.1 Bảng so sánh các tính năng của Harbor và Docker Registry

Tính năng	Harbor	Docker Registry
Giao diện	Thân thiện, trực quan, dễ sử dụng và quản lý.	Cơ bản, đơn giản, chủ yếu thao tác qua dòng lệnh (CLI).
Quản lý người dùng	Hỗ trợ quản lý người dùng, phân quyền truy cập chi tiết (RBAC)	Khả năng quản lý người dùng và phân quyền hạn chế, cần cấu hình thêm.
Quản lý dự án	Cho phép tạo các dự án để nhóm các repository lại với nhau, quản lý dễ dàng hơn.	Không có khái niệm dự án, quản lý repository riêng lẻ.
Quét lỗ hổng bảo mật	Tích hợp sẵn công cụ Trivy để quét lỗ hổng bảo mật trong image.	Cần tích hợp thêm các công cụ
Mở rộng	Dễ mở rộng, thích hợp cho doanh nghiệp và tổ chức lớn	Hạn chế, phù hợp cho nhu cầu nhỏ
Replication (Nhân bản)	Hỗ trợ replication giữa nhiều Harbor	Không có sẵn
Tài liệu & cộng đồng	Được CNCF bảo trợ, cộng đồng lớn, tài liệu đầy đủ	Rộng rãi, do Docker phát triển

Yêu cầu phần cứng	Yêu cầu cấu hình phần cứng cao hơn (CPU, RAM, Disk).	Yêu cầu cấu hình phần cứng thấp hơn.
Chi phí	Miễn phí (mã nguồn mở).	Miễn phí (mã nguồn mở).

Docker Registry: Phù hợp cho các dự án nhỏ, môi trường phát triển hoặc thử nghiệm với yêu cầu đơn giản, triển khai nhanh và tiết kiệm tài nguyên.

Harbor: Lựa chọn tối ưu cho doanh nghiệp cần hệ thống Private Registry với bảo mật cao, quản lý phân quyền chi tiết, quét lỗ hổng, và khả năng mở rộng cho môi trường sản xuất.

Với những ưu điểm vượt trội và khả năng đáp ứng tốt nhu cầu của tổ chức, Harbor là lựa chọn lý tưởng cho việc xây dựng Private Registry.

2.2 Yêu cầu hệ thống

2.2.1 Yêu cầu về phần cứng và phần mềm

Harbor là một giải pháp quản lý container images với nhiều tính năng, bao gồm quản lý quyền truy cập, xác thực người dùng và quét bảo mật. Những tác vụ này yêu cầu CPU và RAM đủ mạnh để xử lý mà không gây ra độ trễ.

Dung lượng lưu trữ đảm bảo đủ không gian cho ít nhất một số lượng lớn images và dữ liệu liên quan. Việc sử dụng SSD giúp tăng tốc độ truy cập và ghi dữ liệu, cải thiện hiệu suất tổng thể của registry. Thông tin phần cứng cần thiết.

Bảng 2.2 Thông tin phần cứng cần thiết

Tài Nguyên	Tối thiểu	Khuyến khích
CPU	2 CPU	4CPU
Mem	4GB	8GB
Disk	40GB	160GB

Harbor được phát triển chủ yếu cho môi trường Linux, nơi các công cụ và thư viện hỗ trợ container hoạt động tối ưu. Hệ điều hành này cũng mang lại tính bảo mật và hiệu suất cao hơn. Nó hoạt động như một registry cho Docker images, vì

vậy việc cài đặt Docker là bắt buộc để chạy và quản lý các container của Harbor. Hơn nữa, Docker Compose giúp đơn giản hóa quá trình triển khai và quản lý các dịch vụ liên quan đến Harbor, cho phép dễ dàng cấu hình và khởi động tất cả các thành phần cần thiết với một lệnh duy nhất.

Bảng 2.3 Phần mềm yêu cầu của hệ thống

Phần mềm	Phiên bản
Docker Engine	20.10.10-ce+ hoặc cao hơn
Docker Compose	v1.18.0+ hoặc docker compose v2 (docker-compose-plugin)
OpenSSL	Mới nhất

2.2.2 Dung lượng lưu trữ và khả năng mở rộng

Khi xây dựng hệ thống Private Docker Registry sử dụng Harbor, việc xác định dung lượng lưu trữ và khả năng mở rộng là yếu tố quan trọng để đảm bảo hệ thống hoạt động hiệu quả và đáp ứng nhu cầu trong tương lai. Dung lượng lưu trữ cần được tính toán dựa trên số lượng image, kích thước trung bình của mỗi image, và tốc độ tăng trưởng dự kiến.

Harbor hỗ trợ khả năng mở rộng linh hoạt thông qua việc tích hợp với các hệ thống lưu trữ phân tán như Amazon S3, Google Cloud Storage, hoặc các giải pháp lưu trữ đám mây khác. Điều này cho phép hệ thống có thể mở rộng dung lượng một cách dễ dàng mà không cần thay đổi cấu trúc hiện có. Ngoài ra, Harbor cũng hỗ trợ bản sao, giúp phân tải và tăng tính sẵn sàng cho hệ thống.

2.2.3 Các yêu cầu bảo mật cơ bản cho kho lưu trữ

Cấu hình bảo mật cho kho lưu trữ sử dụng RBAC

RBAC (Role-Based Access Control) là một cơ chế quản lý quyền truy cập dựa trên vai trò, được sử dụng trong Harbor để kiểm soát quyền truy cập của người dùng đối với các dự án và tài nguyên bên trong kho lưu trữ. Trong Harbor có thể gán các

vai trò khác nhau cho người dùng hoặc nhóm người dùng trong từng dự án, giúp quản lý và bảo mật hiệu quả hơn.

Bảng 2.3 Bảng so sánh giữa một số quyền của của người dùng dựa vào vai trò. [10]

Hành động	Project Admin	Maintainer	Developer	Limited Guest	Guest
Pull image	có	có	có	có	Có
Push image	có	có	có	Không	Không
Sửa cấu hình dự án	Không	có	có	có	Có
Xem cấu hình dự án	có	có	có	có	Có
Thêm Scanners	có	Không	Không	Không	Không
Scan image	có	có	Không	Không	Không

Cấu hình cài đặt trivy để quét các image trong kho lưu trữ

Trivy là một công cụ mã nguồn mở dùng để quét lỗ hổng bảo mật và các cấu hình sai trong các container images, mã nguồn [25]. Để kích hoạt nó phải kích hoạt nó trong tệp harbor.yml như sau:

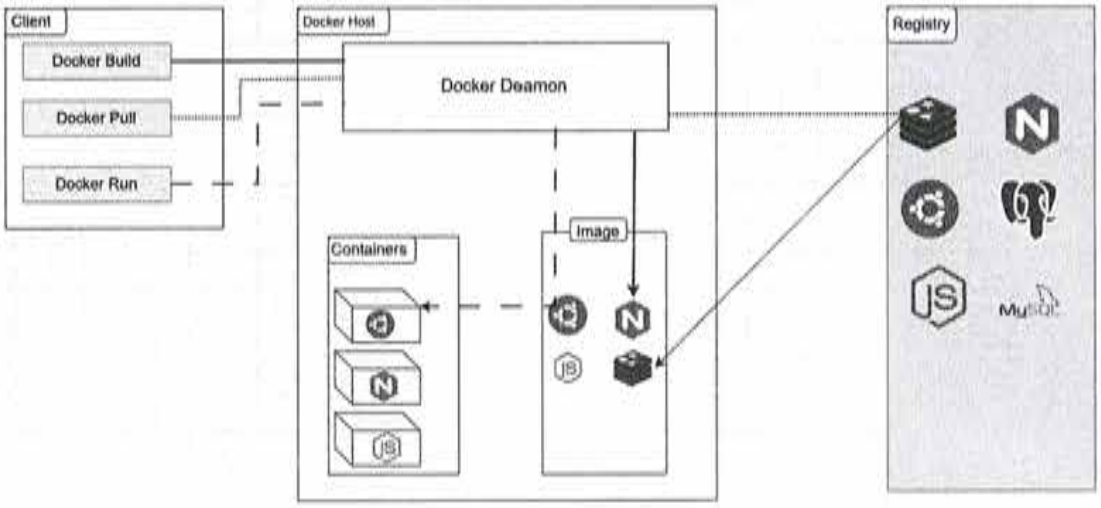
```
scanner:
  trivy:
    enabled: true
    ignoreUnfixed: false
    skipUpdate: false
    insecure: false
```

Sau đó khởi động lại Harbor lúc này khi muốn quét một image cụ thể ta chỉ cần thao tác trên hệ thống.

2.3 Thiết kế kiến trúc hệ thống

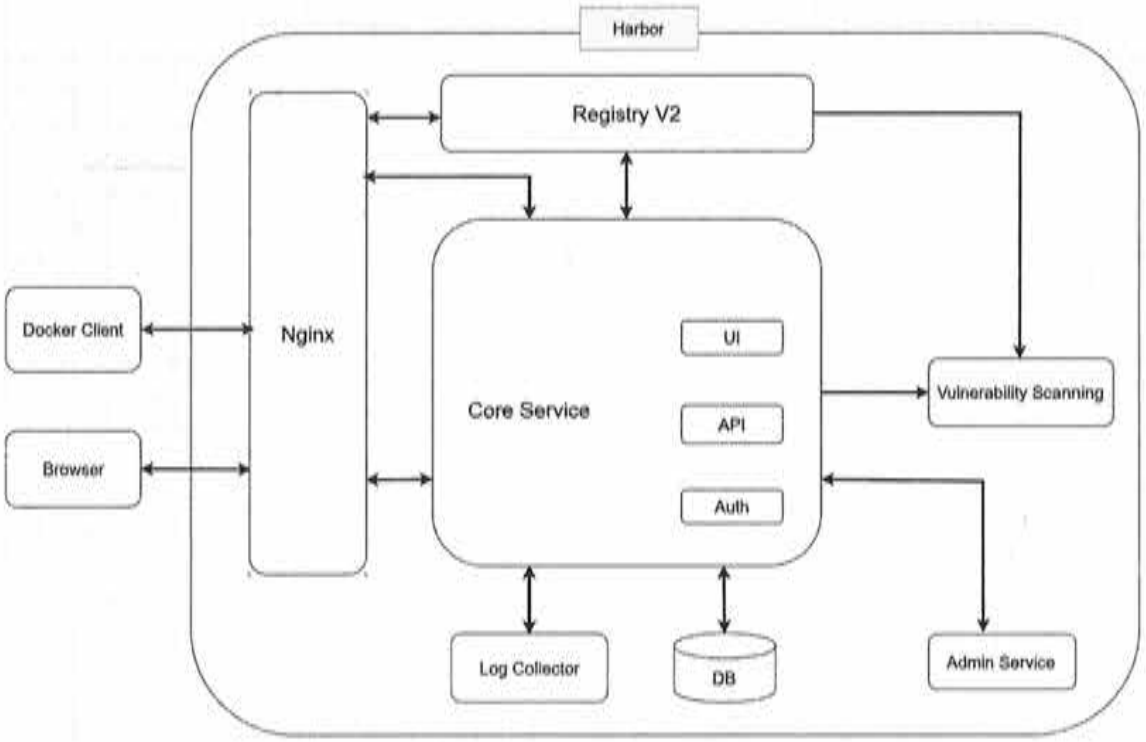
2.3.1 Mô hình kiến trúc Private Docker Registry

Trước khi tìm hiểu về kiến trúc của Private Docker Registry đề án sẽ tìm hiểu sơ qua về kiến trúc của Docker để có cái nhìn tổng quan.



Hình 2.1 Kiến trúc của Docker

Người dùng Docker giao tiếp với daemon Docker, thực hiện nhiệm vụ là xây dựng, chạy và phân phối các container Docker. Người dùng Docker và daemon có thể chạy trên cùng một hệ thống hoặc có thể kết nối máy khách Docker với daemon Docker từ xa. Máy khách Docker và daemon giao tiếp bằng cách sử dụng REST API, giao diện mạng [24] (hình 2.1).



Hình 2.2 Sơ đồ thiết kiến trúc của registry khi sử dụng Harbor.

Mỗi thành phần của Harbor được xây dựng như một container Docker và có thể triển khai bằng Docker Compose. Harbor cần các container sau: Nginx đóng vai trò proxy ngược, phân phối lưu lượng truy cập từ một giao diện người dùng và các lệnh push/pull Docker đến các dịch vụ backend. Registry v2, kho lưu trữ image chính thức của Docker, chịu trách nhiệm lưu trữ và xử lý các tệp image, yêu cầu mã thông báo xác thực cho mỗi lệnh push/pull để kiểm soát quyền truy cập. Cơ sở dữ liệu (PostgreSQL) lưu trữ thông tin người dùng, quyền, nhật ký kiểm tra và dữ liệu nhóm image. Dịch vụ cốt lõi cung cấp giao diện để quản lý image và quyền. Trình thu thập nhật ký tổng hợp nhật ký từ các thành phần qua trình điều khiển nhật ký Docker để phân tích. Luồng dữ liệu bắt đầu từ Nginx chuyển hướng yêu cầu đến UI hoặc Registry, với UI tương tác với cơ sở dữ liệu trong khi dịch vụ cốt lõi điều phối cấu hình hệ thống.

2.3.2 Cách tổ chức và lưu trữ Container image

Khi triển khai Harbor, việc tổ chức và lưu trữ container images là một yếu tố quan trọng để đảm bảo hiệu suất, độ tin cậy và khả năng mở rộng. Harbor hỗ trợ nhiều loại backend lưu trữ khác nhau, bao gồm local disk, NFS, và cloud storage như Amazon S3, Google Cloud Storage, hoặc Azure Blob Storage.

Local Disk

Thư mục mặc định trên ổ đĩa máy chủ là **/data/registry** trên máy chủ triển khai và có thể thay đổi đường dẫn này

Cấu trúc thư mục:

```
/data/registry
├── docker
│   └── registry
│       └── v2
│           ├── blobs
│           └── repositories
└── chart_storage
```

Ưu điểm: đơn giản và có hiệu suất cao do truy cập nhanh do lưu trữ trực tiếp trên ổ đĩa máy chủ

Nhược điểm: Khó mở rộng khi lưu trữ tăng

NFS (Network File System)

Cách tổ chức:

Mount NFS: Mount một thư mục NFS từ máy chủ khác vào thư mục lưu trữ của Harbor (ví dụ: /data/registry).

Cấu trúc thư mục: Tương tự như local disk, nhưng dữ liệu được lưu trữ trên NFS server.

Ưu điểm:

Dễ mở rộng: Có thể tăng dung lượng lưu trữ bằng cách mở rộng NFS server.

Nhược điểm:

Hiệu suất phụ thuộc vào mạng: Nếu mạng chậm, hiệu suất đọc/ghi sẽ bị ảnh hưởng.

Cần quản lý NFS server: Đòi hỏi kiến thức về quản trị hệ thống.

Cloud Storage (Amazon S3)

Cách tổ chức:

Bucket: Harbor lưu trữ images trong các bucket của dịch vụ cloud storage.

Cấu trúc lưu trữ:

Docker images được lưu trữ dưới dạng các object trong bucket.

Metadata và thông tin repository được lưu trữ trong các thư mục con.

Ưu điểm:

Khả năng mở rộng vô hạn: Cloud storage có thể mở rộng dung lượng một cách linh hoạt.

Tính sẵn sàng cao: Dữ liệu được sao lưu tự động và có tính chịu lỗi cao.

Tích hợp dễ dàng: Harbor hỗ trợ sẵn các dịch vụ cloud storage phổ biến.

Nhược điểm:

Chi phí: Sử dụng cloud storage có thể tốn kém, đặc biệt khi lưu trữ lượng lớn dữ liệu.

Phụ thuộc vào mạng: Hiệu suất phụ thuộc vào kết nối mạng đến cloud provider.

2.4 Triển khai Private Docker Registry sử dụng Harbor

2.4.1 Yêu cầu và chuẩn bị về môi trường

Để triển khai hệ thống, việc chuẩn bị môi trường là một bước quan trọng nhằm đảm bảo hệ thống hoạt động hiệu quả và an toàn. Điều này bao gồm việc xác định các yêu cầu phần cứng và phần mềm cần thiết, cũng như cấu hình mạng phù hợp.

Các yêu cầu về môi trường cũng chính là những yêu cầu về phần cứng và phần mềm được nhắc đến ở bảng 2.2 và 2.3 ở trên.

2.4.2 Các bước cài đặt Harbor

Thiết lập kho lưu trữ APT của Docker.

sudo apt-get update

sudo apt-get install ca-certificates curl

sudo install -m 0755 -d /etc/apt/keyrings

sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc

sudo chmod a+r /etc/apt/keyrings/docker.asc

*echo *

"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]

*https://download.docker.com/linux/ubuntu *

*$(. /etc/os-release && echo "${UBUNTU CODENAME: -$VERSION_CODENAME}") stable" | *

sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update

Cài đặt các gói Docker.

sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

Kiểm tra version Docker

sudo docker version

Client: Docker Engine - Community

Version: 28.0.1

Tải xuống gói cài đặt và cấu hình Harbor

Có thể tải xuống trình cài đặt Harbor từ trang phát hành chính thức. Tải xuống trình cài đặt trực tuyến hoặc trình cài đặt ngoại tuyến tuỳ lựa chọn của người sử dụng [13].

Hình 2.3 Hình ảnh các gói cài đặt của Harbor

Giải nén file

tar -zvxf harbor-online-installer-v2.12.2.tgz

Thay đổi tệp cấu hình của Harbor. Tạm thời sẽ chỉ để HTTP và thay đổi hostname thành IP public của EC2.

hostname:18.140.220.206

http:

port

port: 80

cài đặt Harbor bằng cách chạy ./install.sh

./install.sh

Sau khi các thành phần khởi chạy vào trang web kiểm tra với đường dẫn: http://<ip>. Sau đó tiến hành đăng nhập vào trang Web.

Hình 2.4 Giao diện trang web hệ thống sau khi cài đặt thành công

2.5 Cách cấu hình cơ bản

2.5.1 Lưu trữ image

Tất cả các dự án mà người dùng tạo đều được lưu trữ trên thư mục mặc định của Harbor server là /data/registry/docker/registry/v2/repositories.

/data/registry/docker/registry/v2/repositories# ls

library private public

Như có thể thấy câu lệnh ở trên có 3 dự án là library, private, public. Cũng có thể thay đổi đường dẫn lưu trữ image mong muốn trong file docker-compose.yml

```
services:
 registry:
  volumes:
   - /data/registry: /storage: z
```

2.5.2 Tích hợp với người dùng Docker

Đầu tiên sửa cấu hình tệp daemon trên đường dẫn /etc/docker/daemon.json vì do Docker mặc định sử dụng https kể từ Docker 1.3.2 và Harbor sử dụng http thay vì https theo mặc định. Chính vì lý do này nếu muốn đăng nhập vào hệ thống mà sử dụng http thì cần chỉnh sửa file config này.

```
root@ip-10-0-5-156:/home/ubuntu/harbor# nano /etc/docker/daemon.json
{
"insecure-registries": ["18.140.220.206"]
}
```

Sau khi chỉnh sửa thì tiến hành restart lại Docker và đăng nhập vào hệ thống. Thông tin tài khoản đã đăng nhập được lưu trong tệp /root/.docker/config.json. Miễn là thông tin trong /root/.docker/config.json không bị xóa, không cần phải nhập tên người dùng và mật khẩu khi đăng nhập lại.

```
docker login 18.140.220.206
Login Succeeded
```

2.6 Tích hợp và kiểm tra hệ thống

2.6.1 Các loại dự án

Có hai mức độ truy cập được lần lượt là:

Public: Tất cả người dùng đều có quyền đọc đối với các dự án công khai. Phương pháp này rất tiện lợi khi muốn chia sẻ một số kho lưu trữ với người khác.

Private: Chỉ những người có quyền người dùng cụ thể mới có thể truy cập vào các dự án riêng tư. Phương pháp này cũng thuận tiện hơn cho các nhóm nội bộ chia sẻ.

Hình 2.5 Giao diện public và private của hai dự án trên hệ thống

2.6.2 Thực nghiệm push image lên Registry

Giờ đây nếu muốn push/pull các docker image sẽ nhấp vào dự án vừa tạo để xem thông tin gợi ý về việc push cho image.

docker tag SOURCE_IMAGE[:TAG] <ip>/library/REPOSITORY[:TAG]

docker push <IP máy chủ>/library/REPOSITORY[: TAG]

Chỉ cần thay đổi các tham số cần thiết là có thể gắn thẻ và push lên kho lưu trữ. Các tham số cần thiết có thể thấy khi chạy lệnh *docker images.*

/home/ubuntu/harbor# docker tag nginx: latest

18.140.220.206/public/nginx: latest

/home/ubuntu/harbor# docker images

REPOSITORY TAG IMAGE ID SIZE

18.140.220.206/public/nginx latest b52e0b094bc0 192MB

nginx latest b52e0b094bc0 192MB

/home/ubuntu/harbor# docker push 18.140.220.206/public/nginx: latest

Sau khi push thành công có thể vào trang web để kiểm tra như hình ảnh ở dưới tức là đề án đã thành công

Hình 2.6 Image đã thành công được push lên hệ thống

2.6.3 Thực nghiệm pull image từ Registry

Tiếp theo đây đề án sẽ thử pull một image từ trên hệ thống về máy và sử dụng liệt kê danh sách các image trong máy và thấy rằng đề án đã thành công pull image từ trên server về máy. Sau đó sử dụng lệnh kiểm tra : *docker images.*

pull huytamdepzai.net/private/private: latest

REPOSITORY TAG IMAGE ID SIZE

huytamdepzai.net/private/private latest b52e0b094bc0 192MB

2.6.4 Kiểm tra hoạt động và tính ổn định.

Để kiểm tra tính hoạt động của hệ thống ta có thực hiện kiểm tra nhật ký của hệ thống xem hệ thống có hoạt động bình thường không. Nhật ký của hệ thống được lưu trong đường dẫn /var/log/harbor. Trong thư mục này có những tệp nhật ký: *core.log, jobservice.log, portal.log, postgresql.log, proxy.log, redis.log, registry.log, registryctl.log, trivy-adapter.log.*

Có thể sử dụng lệnh *tail -f -n 100 <tên tài liệu>* để kiểm tra nhật kí trạng thái của hệ thống hiện tại, nếu nhật kí được ghi lại một cách đều đặn và không có lỗi xảy ra thì có thể kết luận rằng hệ thống đang hoạt động một cách bình thường.

Hình 2.7 Hình ảnh nhật ký core.log của hệ thống khi hoạt động bình thường

Tệp core.log lưu nhật ký của việc kiểm tra hoạt động của Core Service (UI, API, xác thực) để đảm bảo giao diện người dùng, API, và mã thông báo hoạt động ổn định, ghi nhận các lỗi nếu có. Như hình vẽ trên các dòng nhật ký hiện ra đang ở dạng "INFO" nếu lỗi thì nó sẽ được biểu hiện là "ERROR".

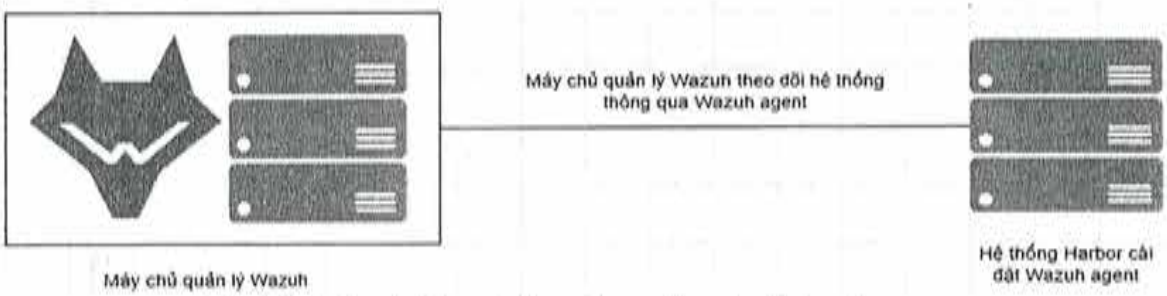
2.7 Giám sát hoạt động của hệ thống

2.7.1 Tổng quát về công cụ giám sát Wazuh

Wazuh, một nền tảng bảo mật mã nguồn mở, là giải pháp hiệu quả để giám sát và bảo vệ các môi trường Docker container. Với khả năng tích hợp chặt chẽ thông qua module Docker, Wazuh theo dõi log container, phát hiện hành vi bất thường như truy cập trái phép hay chạy container với quyền root. Có thể nói giám sát bảo mật là một thách thức lớn đối với mọi doanh nghiệp, đòi hỏi sự đầu tư đáng kể vào quy trình, công nghệ và con người. Hiện có nhiều công cụ giám sát bảo mật hàng đầu như Splunk, Rapid7 MSS... Tuy nhiên, chi phí cao của các giải pháp này thường không phù hợp với các doanh nghiệp vừa và nhỏ. Wazuh nổi lên như một giải pháp thay thế hiệu quả, đáp ứng các tiêu chí: chất lượng cao, miễn phí và dễ sử dụng, giúp các doanh nghiệp nhỏ hơn cũng có thể tiếp cận được công tác giám sát bảo mật toàn diện.

Wazuh là một nền tảng bảo mật mã nguồn mở và miễn phí. Nó theo dõi hệ thống trên các môi trường local, ảo hóa, container và cloud. Wazuh cung cấp các tính năng cần thiết để phát hiện các mối đe dọa phổ biến như scan hệ thống, tấn công dò tìm lỗ hổng.

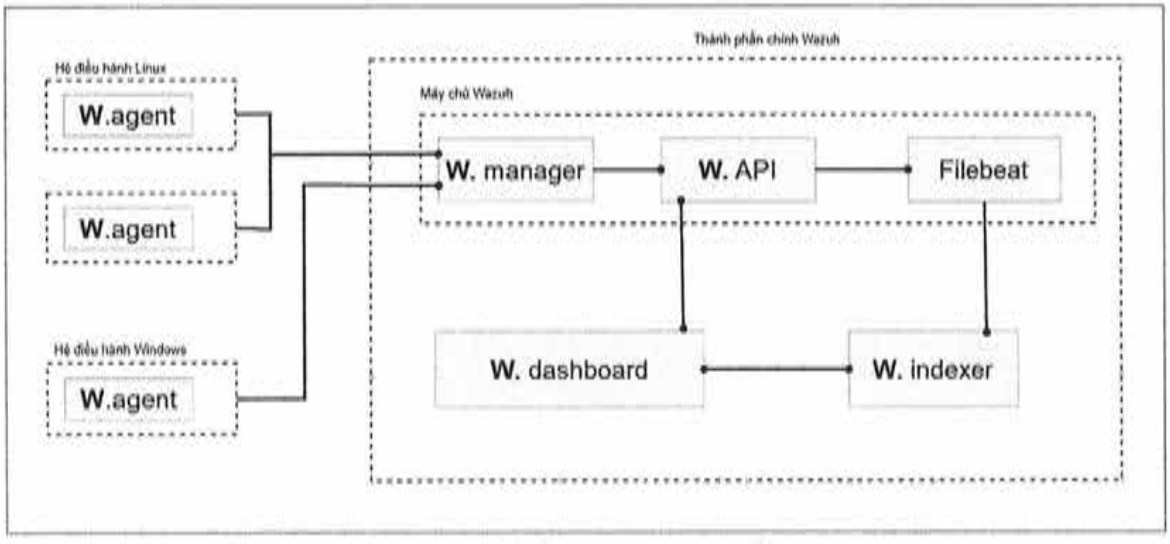
Wazuh cũng dễ dàng cài đặt và linh hoạt, có thể hoạt động ổn định trên các hệ thống với khoảng 2000 người dùng. Wazuh có khả năng tích hợp hiệu quả với các hệ thống thông báo phổ biến như Email, Slack và Discord.



Hình 2.8 Cơ chế hoạt động đơn giản của Wazuh

2.7.2 Các thành phần chính của Wazuh

Giải pháp Wazuh được xây dựng trên một kiến trúc phân tán, bao gồm một tác nhân (agent) duy nhất được triển khai trên các điểm cuối cần giám sát và ba thành phần trung tâm: Wazuh server (còn được gọi là Manager), Wazuh indexer (dựa trên Elasticsearch) và Wazuh dashboard (dựa trên Kibana).



Hình 2.9 Các thành phần chính của hệ thống giám sát Wazuh

Wazuh indexer: Đây là công cụ tìm kiếm và phân tích toàn diện, với khả năng mở rộng cao. Wazuh Indexer đóng vai trò là khu vực lưu trữ chính và hiển thị các cảnh báo được tạo ra bởi Wazuh Server.

Wazuh server: Được thiết kế để xử lý hàng trăm hoặc hàng nghìn dữ liệu đổ về cùng lúc từ các Agent

Wazuh dashboard: Đây là giao diện tương tác trực tiếp với người dùng, giúp trực quan hóa các dữ liệu và thông tin cảnh báo một cách rõ ràng.

Wazuh agents: được cài đặt trên các Endpoint như máy tính xách tay, máy chủ, PC, môi trường đám mây hoặc Docker. Chúng có nhiệm vụ cung cấp các thông tin cần thiết cho Wazuh Server để phát hiện các mối đe dọa.

2.7.3 Cài đặt công cụ giám sát Wazuh

Như đã nói ở phần giới thiệu Wazuh là một công cụ với khả năng cài đặt rất nhanh chỉ cần vài bước cơ bản đã có thể cài đặt thành công hệ thống. Để cài hệ thống ta phải làm các bước sau:

Bước 1: Tải xuống và chạy trợ lý cài đặt Wazuh trên máy chủ quản lý bằng câu lệnh sau.

curl -sO https://packages.wazuh.com/4.12/wazuh-install.sh && sudo bash ./wazuh-install.sh -a

Sau khi cài đặt thành công màn hình hiện tài khoản và mật khẩu để truy cập vào hệ thống.

https://<WAZUH_DASHBOARD_IP_ADDRESS>

User: admin

Password: <ADMIN_PASSWORD>

INFO: Installation finished.

Truy cập vào đường link: https://<WAZUH_DASHBOARD_IP_ADDRESS>.

Hình 2.10 Ảnh minh hoạ hệ thống đã cài đặt thành công

Bước 2: Cài đặt Wazuh agent trên các server cần theo dõi.

Cài đặt khoá GPG

curl -s https://packages.wazuh.com/key/GPG-KEY-WAZUH | gpg --no-default-keyring --keyring gnupg-ring:/usr/share/keyrings/wazuh.gpg --import && chmod 644 /usr/share/keyrings/wazuh.gpg

Thêm kho lưu trữ

echo "deb [signed-by=/usr/share/keyrings/wazuh.gpg] https://packages.wazuh.com/4.x/apt/ stable main" | tee -a /etc/apt/sources.list.d/wazuh.list

Cập nhật thông tin các gói

apt-get update

Triển khai Wazuh agent

WAZUH_MANAGER="<IP máy chủ wazuh đã cài ở bước 1>" apt-get install wazuh-agent

Khởi chạy dịch vụ Wazuh agent

systemctl daemon-reload

systemctl enable wazuh-agent

systemctl start wazuh-agent

Sau khi cài đặt thành công truy cập lại hệ thống sẽ thấy rằng hệ thống đã ghi nhận có 1 agent đang được theo dõi.

Hình 2.11 Hình ảnh minh hoạ tác nhân đã cài đặt thành công và đã được hệ thống theo dõi

2.8 Kết luận Chương II

Trong chương II, đề án đã tìm hiểu về việc xây dựng hệ thống Private Docker Registry, đặc biệt là thông qua hai công cụ chính là Docker Registry tiêu chuẩn và Harbor. Cả hai giải pháp đều mang đến những lợi ích riêng, với Docker Registry phù hợp cho những nhu cầu đơn giản và linh hoạt, còn Harbor nổi bật với tính năng bảo mật và quản lý mạnh mẽ hơn.

Đề án cũng đã thảo luận về yêu cầu hệ thống, bao gồm phần cứng và phần mềm cần thiết để triển khai Harbor. Việc xác định dung lượng lưu trữ và khả năng mở rộng là rất quan trọng để đảm bảo hệ thống hoạt động hiệu quả. Bên cạnh đó, cấu hình bảo mật thông qua RBAC và tích hợp Trivy để quét lỗ hổng bảo mật.

Cuối cùng, các bước cài đặt và cấu hình cơ bản cho Harbor đã được trình bày rõ ràng, cho thấy tính khả thi và sự đơn giản trong việc triển khai một Private Docker Registry. Ngoài ra, khả năng tích hợp Wazuh, một nền tảng giám sát bảo mật mã nguồn mở, đã được đề cập để tăng cường khả năng phát hiện và phản ứng với các mối đe dọa. Wazuh cung cấp khả năng giám sát nhật ký, phát hiện xâm nhập, phân tích hành vi trong thời gian thực từ đó cảnh báo kịp thời các hoạt động bất thường.

Nhìn chung, việc xây dựng một Private Docker Registry là một bước đi chiến lược cho các tổ chức trong việc quản lý và bảo vệ tài sản mã nguồn, đồng thời tối ưu hóa quy trình phát triển phần mềm.

CHƯƠNG III. TRIỂN KHAI GIẢI PHÁP ĐẢM BẢO AN TOÀN CHO HỆ THỐNG PRIVATE DOCKER REGISTRY KHI SỬ DỤNG HARBOR

3.1 Tổng quan về bảo mật Docker

Phần này đề án sẽ đi vào phân tích đánh giá các biện pháp bảo mật hiện có. Docker là một công cụ mạnh mẽ để container hóa ứng dụng, nhưng cũng đi kèm với nhiều rủi ro bảo mật. Dưới đây là phân tích chi tiết các biện pháp bảo.

Không chạy container dưới quyền root: Chạy container với quyền root có thể cho phép kẻ tấn công kiểm soát toàn bộ hệ thống nếu container bị hack. Chạy với tài khoản không phải root giới hạn thiệt hại và là một biện pháp bảo mật hiệu quả. Ví dụ: Thay vì chạy container với lệnh *docker run -u root nginx*, thì sẽ sử dụng tài khoản không phải root *docker run -u 1000:1000 nginx*. Trong Dockerfile, có thể thêm dòng USER 1000 để chạy ứng dụng với tư cách người dùng không đặc quyền [5].

Giới hạn tài nguyên container (CPU, RAM, I/O): Giới hạn tài nguyên ngăn chặn container chiếm dụng hết tài nguyên, gây ảnh hưởng đến các dịch vụ khác. Tuy nhiên, cần cấu hình phù hợp để đảm bảo ứng dụng hoạt động tốt. Ví dụ: Chạy container với giới hạn tài nguyên *docker run --memory="512m" --cpus="0.5" --blkio-weight=100 nginx*. Điều này đảm bảo container không sử dụng quá 512MB RAM, 0.5 CPU và có trọng số I/O thấp [5].

Kiểm tra lỗ hổng bảo mật trong image trước khi triển khai: Việc kiểm tra lỗ hổng bảo mật trước khi triển khai giúp phát hiện và khắc phục các vấn đề bảo mật tiềm ẩn, ngăn chặn các cuộc tấn công có thể xảy ra. Đây là một biện pháp cần thiết để đảm bảo image được sử dụng là an toàn. Trước khi triển khai image python:3.9, sử dụng công cụ Trivy để quét: trivy image python:3.9. Nếu phát hiện lỗ hổng CVE-2023-12345 trong thư viện pip, có thể cập nhật image hoặc cài đặt bản vá trước khi chạy container [5].

Tách biệt mạng, hạn chế port public: Tách biệt mạng và hạn chế port public giúp giảm nguy cơ tấn công trực tiếp từ bên ngoài và lây nhiễm chéo giữa các

container. Đây là biện pháp quan trọng để bảo vệ container khỏi các cuộc tấn công mạng. Ví dụ: Tạo mạng riêng cho container *docker network create test-network* và chạy container trong mạng này *docker run --network test-network nginx.* Chỉ mở port cần thiết *docker run -p 127.0.0.1:8080:80 nginx* [5].

3.2 Các mối đe dọa bảo mật với Private Docker Registry

3.2.1 Truy cập trái phép vào Registry

Truy cập trái phép xảy ra khi các cá nhân hoặc tổ chức không được phép có thể truy cập vào private registry, cho phép họ push/pull các image. Các cấu hình mặc định cho phép image truy cập công khai hoặc thiếu các quyền chi tiết, đặc biệt là việc không triển khai hoặc cấu hình sai kiểm soát truy cập dựa trên vai trò (RBAC), có thể mở ra cánh cửa cho kẻ tấn công. Ví dụ cấu hình RBAC không chặt chẽ, cho phép các vai trò hoặc người dùng không được phép truy cập vào registry, ví dụ, một vai trò "chỉ xem" có thể push image thay vì chỉ pull.

Kẻ tấn công có thể push các image độc hại vào registry, và khi triển khai, những image này có thể thực thi mã độc trên hệ thống host, gây ra mất dữ liệu, cài đặt các hình thức khai thác khác. Dữ liệu nhạy cảm của khách hàng hoặc công ty, tài sản trí tuệ, image độc quyền có nguy cơ bị đánh cắp. Mã độc có thể được chèn vào các image, được ký bằng chứng chỉ bị rò rỉ, sau đó được phân phối đến các hệ thống hạ nguồn, tạo ra rủi ro nghiêm trọng.

3.2.2 Tấn công MITM (Man-in-the-Middle)

Tấn công MITM xảy ra khi kẻ tấn công chặn và can thiệp vào liên lạc giữa máy khách (chẳng hạn, máy của nhà phát triển) và Private Registry Docker. MITM thường xuất hiện khi giao tiếp giữa máy khách và registry không được bảo vệ, ví dụ sử dụng HTTP thay vì HTTPS. Điều này cho phép kẻ tấn công đánh cắp thông tin đăng nhập, sửa đổi image trong quá trình truyền, hoặc chèn nội dung độc hại. Các nghiên cứu chỉ ra rằng MITM là mối đe dọa phổ biến trong môi trường containerized, đặc biệt khi liên lạc không được mã hóa.

Kẻ tấn công có thể chặn lưu lượng không được mã hoá nếu HTTPS/TLS không được cài đặt, kẻ tấn công cũng có thể dễ dàng chặn các giao tiếp giữa máy khách Docker và Registry, đọc hoặc sửa đổi dữ liệu. thậm chí là giả mạo tên miền chuyển hướng các yêu cầu registry hợp pháp đến một máy chủ độc hại là một kỹ thuật phổ biến trong các cuộc tấn công MITM.

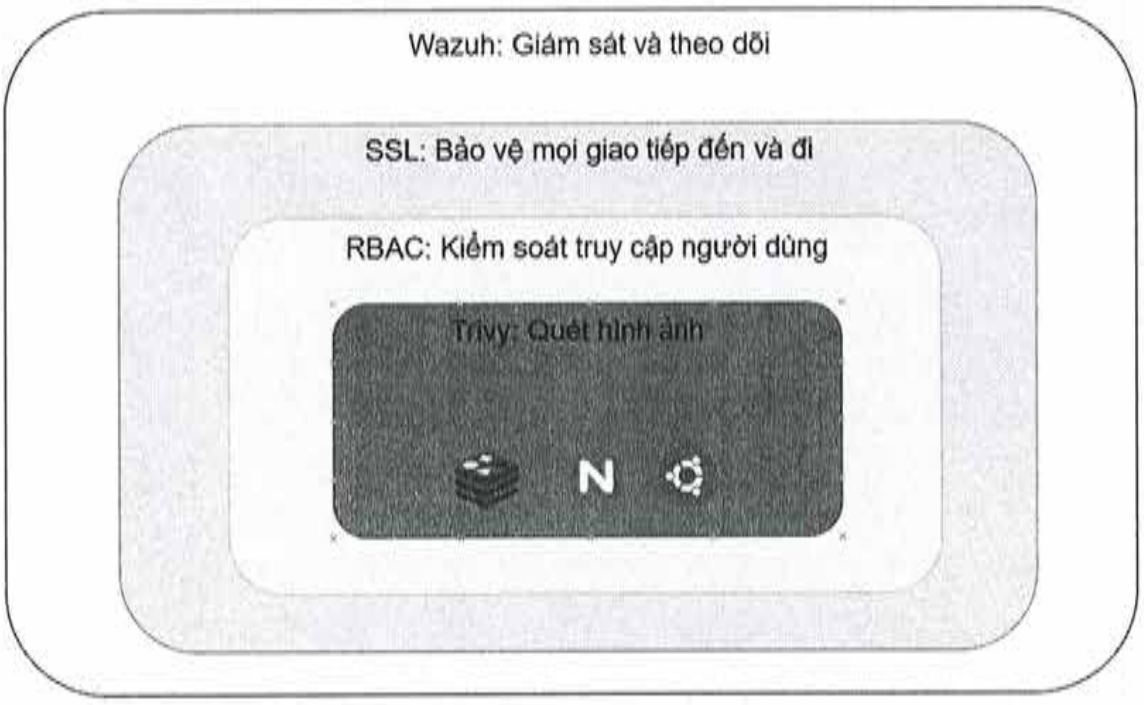
3.2.3 Lỗ hổng trong image

Các lỗ hổng trong image trở thành mối đe dọa nghiêm trọng nếu không quét định kỳ để phát hiện các điểm yếu bảo mật đã biết (CVE). Những image này có thể chứa lỗ hổng từ image lỗi thời hoặc ứng dụng chưa được vá, tạo nguy cơ bị khai thác khi triển khai. Nếu không kiểm tra image trước khi push lên registry hoặc sử dụng, tổ chức có thể vô tình triển khai image chứa các lỗ hổng nghiêm trọng, như SQL injection hoặc CVE đã biết. Điều này đặc biệt nguy hiểm trong môi trường sản xuất, nơi image được sử dụng rộng rãi.

Kẻ tấn công có thể truy cập hệ thống host, gây ra vi phạm dữ liệu, cài đặt mã độc, hoặc thực hiện tấn công DoS. Hậu quả có thể nghiêm trọng, đặc biệt khi các image được dùng trong các ứng dụng quan trọng, ảnh hưởng lớn đến hoạt động sản xuất. Việc không quét lỗ hổng image một cách thường xuyên và tự động sẽ làm tăng đáng kể rủi ro bảo mật. Nếu không có tự động hóa và tích hợp sớm trong quy trình CI/CD, các lỗ hổng có khả năng cao sẽ lọt vào môi trường sản xuất do tốc độ phát triển nhanh chóng, khiến việc kiểm tra thủ công trở nên không thực tế và kém hiệu quả.

3.3 Giải pháp bảo mật cho Private Docker Registry

Phần trên đề án đã phân tích các mối đe dọa đến hệ thống Docker Registry. Trong phần này, đề án sẽ tập trung vào các giải pháp bảo mật nhằm tăng cường an ninh, với bốn yếu tố cốt lõi tương ứng các mối đe dọa ở phần trên: quản lý quyền truy cập (RBAC), sử dụng SSL để mã hóa dữ liệu và ngăn chặn tấn công, sử dụng công cụ quét lỗ hổng như Trivy và cuối cùng là công cụ giám sát Wazuh.



Hình 3.1 Các lớp bảo vệ cho hệ thống

Thứ nhất, sử dụng công cụ quét lỗ hổng như Trivy mang lại khả năng phát hiện và ngăn chặn các rủi ro bảo mật trong container image. Trivy là một công cụ mạnh mẽ giúp phân tích từng lớp của container image để tìm kiếm các lỗ hổng bảo mật (CVE) trong thư viện, framework, hoặc hệ điều hành cơ sở. Công cụ này có thể dễ dàng tích hợp vào hệ thống Harbor hoặc pipeline CI/CD để tự động quét container image trước khi triển khai. Nhờ đó, Trivy giúp phát hiện các rủi ro bảo mật tiềm ẩn sớm, ngăn chặn việc triển khai các container không đạt chuẩn ra môi trường sản xuất.

Thứ hai, quản lý quyền truy cập bằng RBAC (Role-Based Access Control) là một trong những biện pháp quan trọng để đảm bảo chỉ những người dùng hoặc nhóm người dùng được phân quyền cụ thể mới có thể truy cập hoặc thực hiện các hành động trên Docker Registry. Việc triển khai RBAC cho phép người quản trị định nghĩa các vai trò như Admin (toàn quyền quản lý), Developer (push/pull image), và Viewer (chỉ xem hoặc pull image). Điều này giúp hạn chế quyền truy cập không cần thiết, giảm nguy cơ lạm dụng quyền và dễ dàng quản lý quyền theo từng vai trò, dự án hoặc nhóm người dùng. Nhờ đó, hệ thống sẽ được bảo vệ tốt hơn khỏi các hành vi truy cập trái phép.

Thứ ba, sử dụng SSL để mã hóa dữ liệu truyền tải và ngăn chặn tấn công là một yêu cầu bắt buộc đối với các Docker Registry. SSL/TLS đảm bảo rằng dữ liệu trao đổi giữa Docker Registry và các client được mã hóa, từ đó ngăn chặn các cuộc tấn công man-in-the-middle (MITM). Điều này giúp bảo vệ thông tin nhạy cảm như thông tin đăng nhập và nội dung image. Người quản trị cần triển khai SSL bằng cách sử dụng chứng chỉ SSL từ một tổ chức phát hành uy tín hoặc tự ký nếu hệ thống chỉ sử dụng nội bộ. Việc sử dụng SSL không chỉ tăng cường bảo mật mà còn đảm bảo tính toàn vẹn và bảo mật của dữ liệu khi truyền tải.

Thứ tư, triển khai công cụ giám sát và phân tích bảo mật như Wazuh là một giải pháp toàn diện để nâng cao khả năng phòng thủ của Private Docker Registry. Wazuh hoạt động như một nền tảng SIEM (Security Information and Event Management) và XDR (Extended Detection and Response), cung cấp khả năng thu thập, phân tích và tương quan các sự kiện bảo mật từ nhiều nguồn khác nhau. Trong bối cảnh Docker Registry, Wazuh có thể giám sát liên tục các hoạt động truy cập, thay đổi cấu hình, cũng như các hành vi bất thường trên máy chủ chứa Registry và các container liên quan.

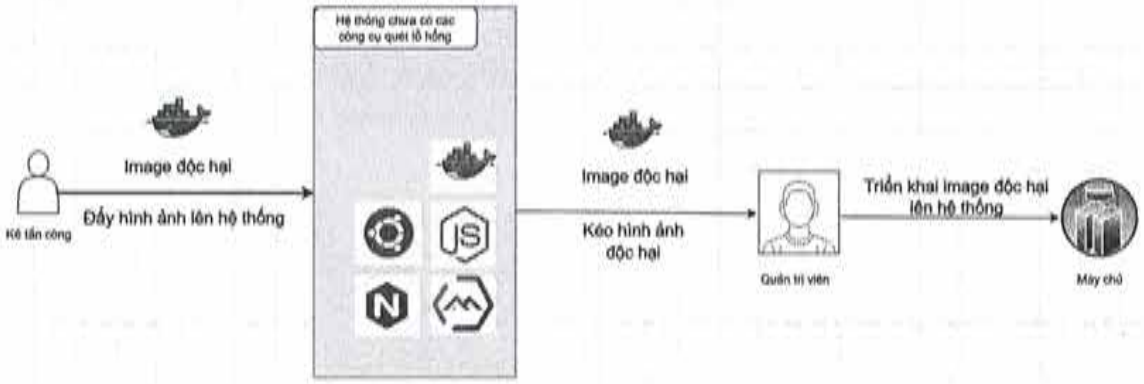
3.4 Quét lỗ hổng container image

3.4.1 Kịch bản tấn công khi chưa tích hợp các công cụ quét lỗ hổng

Kịch bản tấn công phổ biến là kẻ tấn công đưa mã độc vào một image, chẳng hạn như mã khai thác tiền mã hóa, mã thực hiện tấn công DDoS, hoặc phần mềm gián điệp, rồi tải image này lên các public registry. Nếu người dùng tải image này về và push lên Harbor mà không thực hiện quét bảo mật, mã độc có thể dễ dàng được triển khai trong môi trường sản xuất.

Khi container được chạy, mã độc có thể kích hoạt các hành vi nguy hiểm như sử dụng tài nguyên hệ thống để khai thác tiền mã hóa, gửi lưu lượng độc hại để thực hiện tấn công DDoS, hoặc lợi dụng quyền root của container để leo thang đặc quyền và chiếm quyền kiểm soát toàn bộ hệ thống. Điều này không chỉ làm hệ thống bị lạm

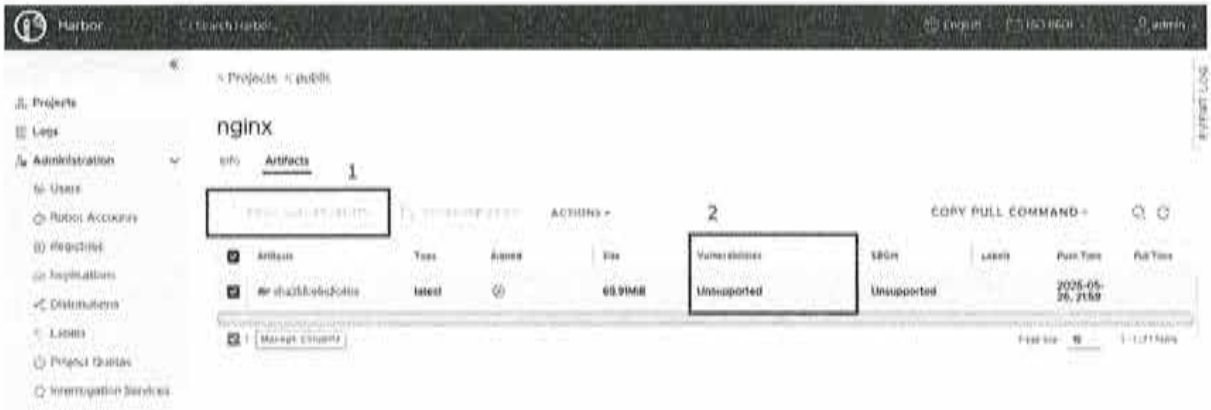
dụng tài nguyên mà còn có thể dẫn đến rò rỉ dữ liệu nhạy cảm hoặc phá hủy cơ sở hạ tầng.



Hình 3.2 Hình vẽ minh hoạ cách Hacker push image độc hại lên hệ thống

Hacker xây dựng image có chứa mã độc hoặc lỗ hổng, push lên hệ thống. Không có tool quét (Trivy) nên image không được phát hiện nguy hiểm. Người dùng hoặc CI/CD pull image này về triển khai vào môi trường sản xuất. Ứng dụng bị tấn công, nguy cơ rò rỉ dữ liệu, bị chiếm quyền hoặc phá hoại.

Khi chưa có các công cụ quét bảo mật tích hợp, Harbor sẽ không thể tự động kiểm tra các lỗ hổng bảo mật (vulnerability) hay các vấn đề tiềm ẩn khác trong các layer của image.



Hình 3.3 Hình ảnh hệ thống khi chưa được tích hợp các công cụ quét lỗ hổng

Ô số 1 trong hình vẽ chỉ ra rằng chức năng quét lỗ hổng bảo mật hiện tại chưa được tích hợp vào hệ thống. Tương tự, ô số 2 cho thấy hệ thống không hỗ trợ việc phát hiện các lỗ hổng tiềm ẩn đang tồn tại.

3.4.2 Tích hợp các công cụ như Clair, Trivy Aqua Security để phát hiện lỗ hổng trong image

Trong môi trường DevOps, việc kiểm tra lỗ hổng thủ công trở nên hạn chế. Do đó, cần có các giải pháp quét tự động và liên tục để xác định các lỗ hổng sớm trong vòng đời phát triển phần mềm. Điều này đảm bảo rằng chỉ những image an toàn và đáng tin cậy mới được triển khai. Hơn nữa, việc tích hợp các công cụ quét lỗ hổng giúp đáp ứng các yêu cầu tuân thủ của doanh nghiệp. Tự động hóa là yếu tố then chốt để đảm bảo an ninh container hiệu quả do tốc độ và số lượng triển khai lớn trong môi trường hiện đại. Việc quét tự động được tích hợp vào quy trình CI/CD cung cấp khả năng giám sát liên tục và ngăn ngừa lỗi.

Sau khi

Clair là một công cụ quét lỗ hổng mã nguồn mở khác, trước đây được sử dụng trong Harbor.

Khi sử dụng câu lệnh chạy để cài đặt hệ thống ./install.sh thêm lựa chọn Clair để Clair có thể được cài đặt cùng hệ thống

./install.sh --with-clair

Trivy là một công cụ quét lỗ hổng mã nguồn mở, thường là công cụ mặc định trong Harbor.

Khi sử dụng câu lệnh chạy để cài đặt hệ thống ./install.sh thêm lựa chọn Trivy để Trivy có thể được cài đặt cùng hệ thống

./install.sh --with-trivy

Aqua Security là một nhà cung cấp các giải pháp an ninh container, bao gồm cả mã nguồn mở (Trivy) và thương mại (Aqua CSP). Về cơ bản, Aqua Security không chỉ là một công cụ quét lỗ hổng đơn thuần như Trivy và Clair. Mặc dù Aqua Security có tích hợp khả năng quét lỗ hổng, nhưng nó là một nền tảng bảo mật container toàn diện hơn nhiều. Dưới đây là bảng so sánh giữa các công cụ dùng để phát hiện lỗ hổng image.

Bảng 3.1 Bảng so sánh giữa các công cụ quét bảo mật

Tính năng/Đặc điểm	Trivy	Clair	Aqua Security
Chức năng chính	Quét lỗ hổng bảo mật trong container images, file systems, repositories.	Quét lỗ hổng bảo mật trong container images.	Nền tảng bảo mật container toàn diện, bao gồm quét lỗ hổng, quản lý tuân thủ, bảo mật thời gian chạy.
Loại công cụ	Công cụ quét lỗ hổng mã nguồn mở.	Công cụ quét lỗ hổng mã nguồn mở.	Nền tảng bảo mật thương mại.
Độ chính xác	Cao, sử dụng nhiều nguồn dữ liệu.	Tốt, nhưng có thể cần cấu hình thêm để tối ưu.	Rất cao, tích hợp nhiều công nghệ và dữ liệu.
Tốc độ quét	Rất nhanh.	Chậm hơn Trivy.	Tùy thuộc vào cấu hình và quy mô quét.
Khả năng tích hợp	Tốt, dễ dàng tích hợp vào CI/CD.	Tốt, nhưng có thể phức tạp hơn.	Mạnh mẽ, tích hợp sâu vào quy trình DevOps.
Chi phí	Miễn phí (mã nguồn mở).	Miễn phí (mã nguồn mở).	Trả phí (thương mại).

3.4.3 Kết hợp quét bảo mật vào quy trình tích hợp liên tục/ triển khai liên tục (CI/CD)

Việc tích hợp bảo mật vào giai đoạn đầu của vòng đời phát triển phần mềm là rất quan trọng để xây dựng các ứng dụng an toàn. Các đoạn thông tin như nhấn mạnh lợi ích của việc tích hợp Harbor từ đó nâng cao tư thế bảo mật tổng thể.

Harbor cung cấp khả năng quét lỗ hổng tích hợp thông qua việc tích hợp các trình quét mã nguồn mở như Trivy và Clair. Các trình quét này thực hiện phân tích tĩnh các ảnh thùng chứa để xác định các lỗ hổng đã biết trong các gói hệ điều hành và các thư viện phụ thuộc của ứng dụng.

Dưới đây là hình ảnh khi tích hợp quét lỗ hổng bảo mật vào quá trình tích hợp và triển khai.

```
- name: Run Trivy vulnerability scanner
  uses: aquasecurity/trivy-action@7b7aa264d83dc58691451798b4d117d53d21edfe
  with:
    image-ref: 'docker.io/my-organization/my-app:${{ github.sha }}'
    format: 'template'
    template: '@/contrib/sarif.tpl'
    output: 'trivy-results.sarif'
    severity: 'CRITICAL,HIGH'
```

Hình 3.4 Cấu hình trivy trong github action

Đặt tên cho bước này là "Run Trivy vulnerability scanner" để dễ dàng theo dõi trong nhật ký của GitHub Actions.

uses: aquasecurity/trivy-action@7b7aa264d83dc58691451798b4d117d53d21edfe: Sử dụng hành động từ kho lưu trữ. Hành động này tích hợp công cụ quét lỗ hổng bảo mật Trivy vào quy trình làm việc.

3.4.4 Thực nghiệm bảo mật quét lỗ hổng trên một số thùng chứa image mẫu.

Cơ chế khi Trivy quét lỗ hổng bảo mật là khi tích hợp với Trivy để quét lỗ hổng bảo mật, Trivy hoạt động như một công cụ quét tĩnh để phát hiện các lỗ hổng trong container images. Trivy phân tích các lớp (layers) của container image để kiểm tra các gói phần mềm, thư viện, và các thành phần khác được cài đặt trong image. Trivy sử dụng cơ sở dữ liệu lỗ hổng để so sánh các thành phần trong container image với danh sách các lỗ hổng đã biết. Tiếp theo đây sẽ là thực nghiệm sử dụng Trivy để quét lỗ hổng trong các image.

Hình 3.5 Các bước để quét lỗ hổng của một image

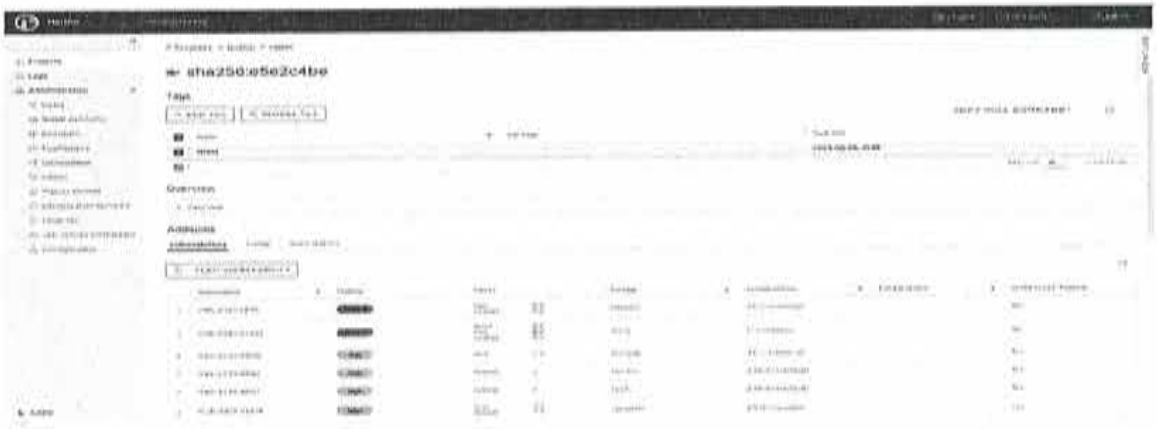
Bước 1 Chọn dự án muốn quét

Bước 2 Chọn chọn kho lưu trữ

Bước 3 Chọn image mà mình muốn quét

Bước 4 Chọn vào "SCAN VULNERABILITY"

Sau đó có thể chọn vào image đó và thấy kết quả sau khi quét gồm các lỗ hổng bảo mật được xếp ở các cấp độ khác nhau.

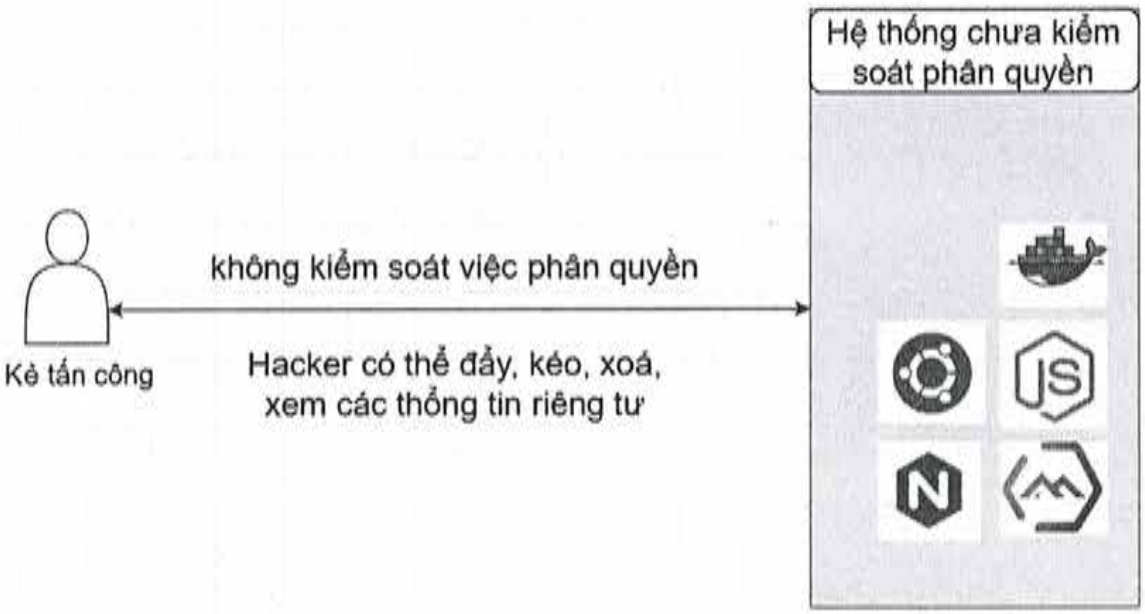


Hình 3.6 Giao diện hệ thống sau khi quét image

3.5 Xác thực và phân quyền người dùng

3.5.1 Kịch bản tấn công khi chưa phân quyền cho người dùng.

Các hệ thống quản lý image container như Harbor Docker Registry đóng vai trò quan trọng trong chuỗi phát triển và triển khai phần mềm. Tuy nhiên, nếu không cấu hình cơ chế kiểm soát truy cập dựa trên vai trò (RBAC) cho từng người dùng, hệ thống sẽ trở nên vô cùng dễ bị tổn thương. Khi đó, bất kỳ ai biết được địa chỉ registry đều có thể truy cập, thao tác và thực hiện các hành động nguy hiểm như tải lên image độc hại, tải xuống các image nhạy cảm hoặc thậm chí xóa toàn bộ repository. Việc thiếu RBAC không chỉ mở cửa cho hacker khai thác lỗ hổng bảo mật mà còn đe dọa nghiêm trọng đến tính an toàn và toàn vẹn của toàn bộ hệ thống ứng dụng.

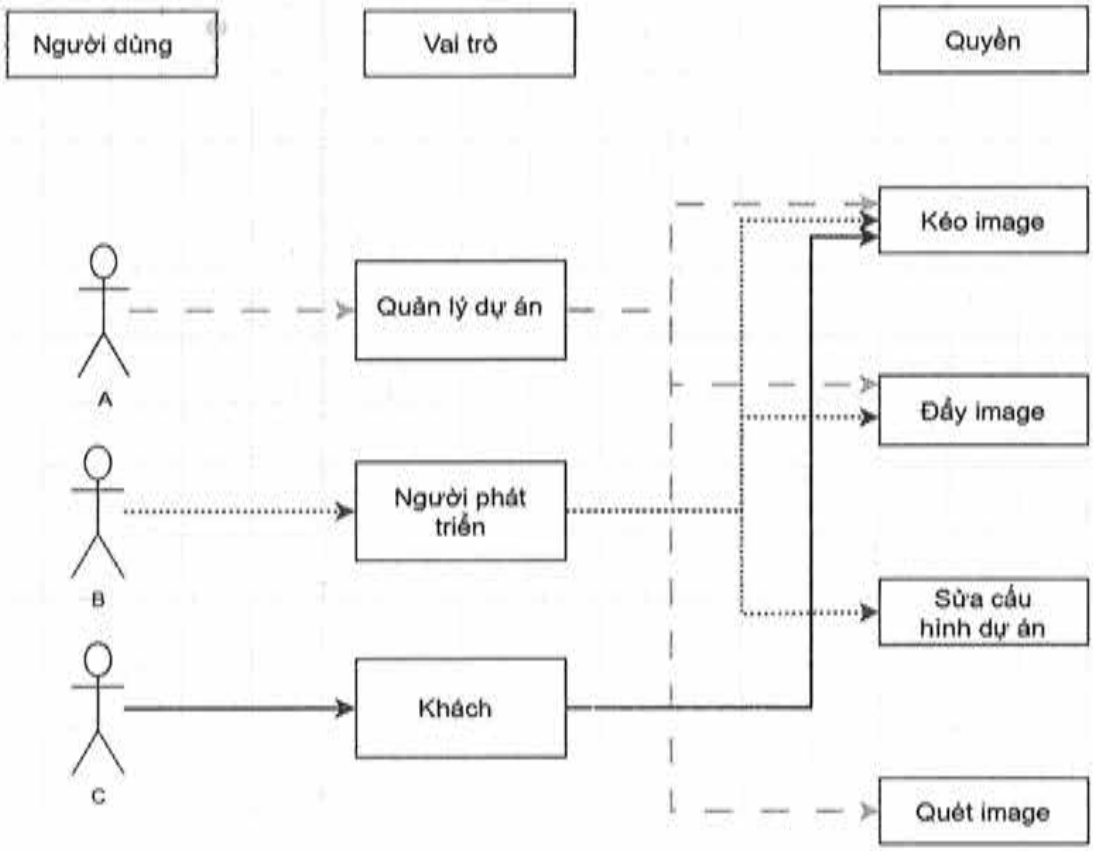


Hình 3.7 Hình vẽ minh hoạ cách kẻ tấn công tấn công hệ thống khi chưa phân quyền

Hacker hoặc người lạ biết đường dẫn đến Harbor Registry. Harbor không có RBAC (phân quyền từng user) ⇒ không kiểm soát được ai được làm gì. Hacker có thể push, pull, xóa repository hoặc image hoặc xem thông tin các Private Registry.

3.5.2 Thiết lập Kiểm soát truy cập dựa trên vai trò (RBAC) trong hệ thống.

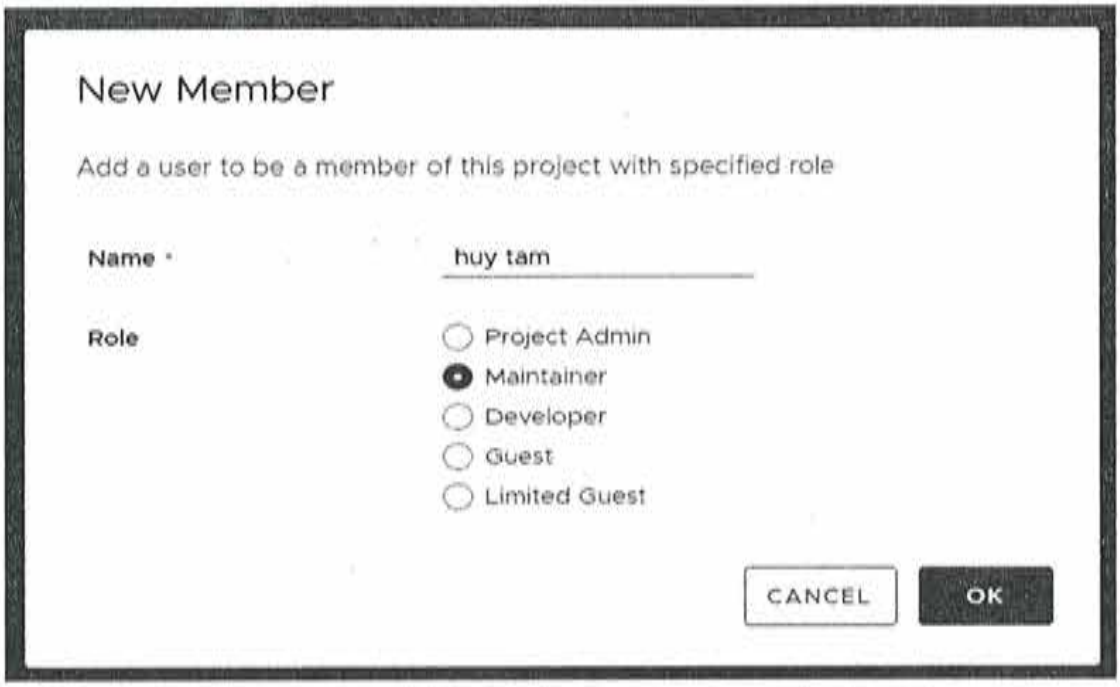
Một trong những tính năng nổi bật của Harbor là RBAC (Role-Based Access Control), cho phép quản lý quyền truy cập đến các tài nguyên trong Harbor một cách hiệu quả và linh hoạt. Nó không chỉ giúp tăng cường bảo mật bằng cách hạn chế quyền truy cập chỉ cho những đối tượng cần thiết, mà còn đơn giản hóa quá trình quản lý quyền thông qua việc gán vai trò. Với RBAC, có thể dễ dàng cấp quyền xem, pull, hoặc push image cho từng người dùng hoặc nhóm trong một dự án cụ thể, chỉ định người quản lý dự án với quyền hạn đầy đủ.



Hình 3.8 Hình vẽ minh họa quyền của từng người dùng dựa vào vai trò

Từ hình vẽ trên có thể thấy rằng người dùng A có vai trò là quản lý dự án với các quyền như là pull image, push image, quét image và xem cấu hình dự án. Với người dùng B là người phát triển có quyền pull, push image, xem cấu hình và sửa cấu hình. Tương tự với người dùng C. Sau khi được phân quyền người dùng giả sử người dùng C (với vai trò là khách) muốn push một image của họ lên thì sẽ không được vì vai trò của họ không cho phép làm điều đó.

Để biết rõ hơn về các các quyền mà người dùng có thể có khi đứng ở từng vai trò cụ thể tham khảo "Quyền của của thành viên dự án" trong [10].

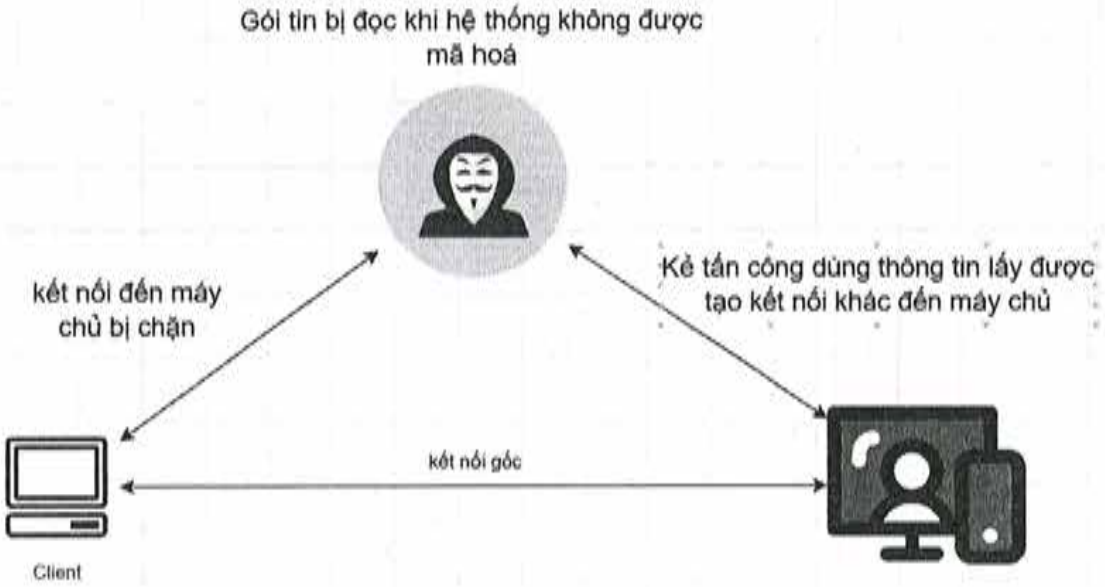


Hình 3.9 Hình vẽ minh hoạ chọn vai trò người dùng khi thêm người dùng trước khi thêm vào hệ thống

3.6 Mã hoá giao tiếp

3.6.1 Kịch bản tấn công trung gian khi chưa mã hoá giao tiếp sử dụng Wireshark.

Nhà phát triển gửi thông tin đăng nhập (tài khoản, mật khẩu) qua HTTP tới hệ thống. Kẻ tấn công sử dụng công cụ như Wireshark hoặc các proxy MITM (MITMproxy, ettercap...) để chặn và đọc gói tin. Toàn bộ thông tin xác thực đi qua mạng đều ở dạng có thể xem được, kẻ tấn công dễ dàng lấy được tài khoản. Kẻ tấn công dùng tài khoản vừa đánh cắp để truy cập, thao tác tùy ý như xoá, push, pull với hệ thống điều này là rất nguy hiểm.



Hình 3.10 Hình vẽ minh hoạ cách kẻ tấn công tấn công hệ thống khi gói tin không được mã hoá

Người dùng nhập tên tài khoản và mật khẩu, sau đó thực hiện đăng nhập vào Harbor thông qua lệnh *docker login*. Thông tin đăng nhập này được gửi trực tiếp từ máy của khách hàng tới máy chủ Harbor thông qua giao thức HTTP. Do không có SSL nên toàn bộ dữ liệu này đi qua mạng dưới dạng không mã hóa.

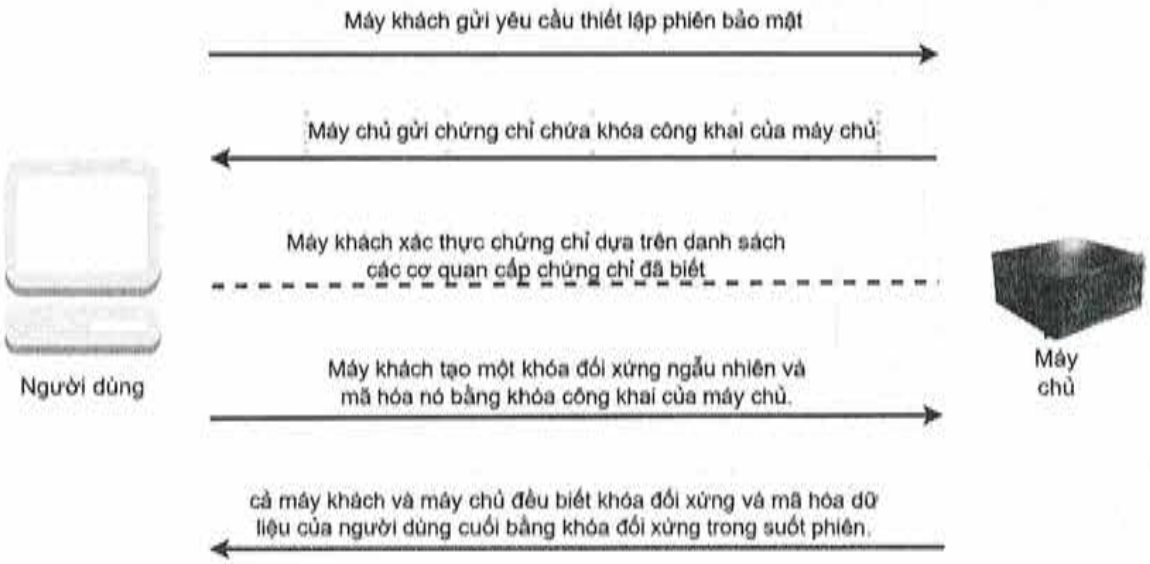
Trong cùng mạng, có một kẻ tấn công hoặc một thiết bị đã bị chiếm quyền kiểm soát đang chạy phần mềm Wireshark hoặc các công cụ bắt gói tin tương tự. Khi khách hàng gửi thông tin đăng nhập, Wireshark sẽ hiển thị rõ ràng tài khoản và mật khẩu ngay trong nội dung các gói tin HTTP.

Sau khi có được tài khoản và mật khẩu, kẻ tấn công tiến hành đăng nhập vào hệ với tư cách là nhà phát triển hoặc thậm chí là admin (nếu bắt được tài khoản admin). Từ đây, kẻ tấn công có thể thực hiện nhiều hành động nguy hiểm mà hệ thống không thể phát hiện như là push, pull, xóa image hoặc toàn bộ repository, gây gián đoạn hoặc phá hoại hệ thống của tổ chức.

Hình 3.11 Hình vẽ minh hoạ khi sử dụng công cụ như Wireshark để đọc tài khoản và mật khẩu của khách hàng khi hệ thống chưa được mã hoá.

3.6.2 Thiết lập kết nối mã hoá dữ liệu.

Mã hóa giao tiếp là một yếu tố quan trọng trong việc bảo vệ dữ liệu trên Internet, đặc biệt là khi truyền tải thông tin nhạy cảm giữa client và server. Việc sử dụng mã hóa giúp ngăn chặn các mối đe dọa như nghe lén và tấn công man-in-the-middle, đảm bảo rằng thông tin cá nhân và dữ liệu tài chính luôn được bảo vệ. Một trong những phương pháp phổ biến nhất để mã hóa giao tiếp là thông qua HTTPS, với sự hỗ trợ của hai giao thức bảo mật SSL (Secure Sockets Layer) và TLS (Transport Layer Security) [29].



Hình 3.12 Hình vẽ minh hoạ quá trình mã hoá giao tiếp

1. Máy khách gửi yêu cầu đến máy chủ để thiết lập một phiên bảo mật. Máy chủ phản hồi bằng cách gửi chứng chỉ số X.509 của mình đến máy khách.

2. Máy khách nhận được chứng chỉ số X.509 của máy chủ.

3. Máy khách xác thực máy chủ bằng cách sử dụng danh sách các cơ quan cấp chứng chỉ đã biết.

4. Máy khách tạo một khóa đối xứng ngẫu nhiên và mã hóa nó bằng khóa công khai của máy chủ.

5. Hiện tại, cả máy khách và máy chủ đều biết khóa đối xứng và có thể sử dụng quy trình mã hóa SSL để mã hóa và giải mã thông tin có trong yêu cầu của máy khách và phản hồi của máy chủ.

3.6.3 Thiết lập chứng chỉ SSL để bảo mật giao tiếp

Đầu tiên xoá thông tin đăng nhập được lưu lúc chưa dùng SSL trong folder /root/. docker. Sau khi xóa file cần restart lại Docker. Tiếp theo sẽ tạo thư mục mới tên là ssl dùng để chứa key và tạo CA.

/home/ubuntu/harbor/ssl# openssl req -newkey rsa:4096 -nodes -sha256 -keyout ca.key -x509 -days 365 -out ca.crt

Tạo một yêu cầu chứng chỉ csr

/home/ubuntu/harbor/ssl# openssl req -newkey rsa:4096 -nodes -sha256 -keyout 18.140.220.206.key -out 18.140.220.206.csr

Tạo chứng chỉ

/home/ubuntu/harbor/ssl# echo subjectAltName = IP:18.140.220.206 > extfile.cnf

/home/ubuntu/harbor/ssl# openssl x509 -req -days 365 -in 18.140.220.206.csr -CA ca.crt -CAkey ca.key -CAcreateserial -extfile extfile.cnf -out 18.140.220.206.crt

Sau khi tạo chứng chỉ thành công sẽ thiết lập nó vào hệ thống. Đầu tiên sẽ tiến hành tắt hệ thống sau đó sửa đổi tệp harbor.yml

hostname: 18.140.220.206

https:

port: 443

certificate: /home/ubuntu/harbor/ssl/18.140.220.206.crt

private_key: /home/ubuntu/harbor/ssl/18.140.220.206.key

Chú ý đặc biệt:

Định dạng cấu hình được sửa đổi trong tệp harbor.yml ở trên phải chính xác! "https" phải được viết ở đầu dòng và thụt lề của "port:443", "certificate" và "private key" phải nhất quán!

Nếu không, khi thực hiện lệnh cập nhật ". /prepare" bên dưới sẽ lỗi do định dạng cấu hình không đúng. Sau khi đã chắc chắn đúng chỉnh sửa file cấu hình sẽ tiến hành chạy file ./prepare để cập nhật lại cấu hình.

/home/ubuntu/harbor#. /prepare

Khởi động lại docker-compose

/home/ubuntu/harbor# docker compose up -d

Đăng nhập lại vào hệ thống thì sẽ thấy rằng có lỗi "*Error response from daemon: Get "https://18.140.220.206/v2/": tls: failed to verify certificate: x509: certificate signed by unknown authority*"

/home/ubuntu/harbor# docker login 18.140.220.206

Username: admin

*Password: ****

Error response from daemon: Get "https://18.140.220.206/v2/": tls: failed to verify certificate: x509: certificate signed by unknown authority

Lỗi trên xảy ra, giải pháp là:

Tình huống này thường xảy ra trong các chứng chỉ tự ký. Thông báo lỗi có nghĩa là cơ quan cấp chứng chỉ chưa được xác thực và không thể xác định được.

Giải pháp: Phải sao chép chứng chỉ Harbor của server vào mỗi máy khách và thực hiện thao tác uỷ quyền.

*root@ip-10-0-5-156:/home/ubuntu/harbor/ssl # cp * /etc/docker/certs.d/18.140.220.206/*

root@harbor-node harbor]# chmod 644 /etc/pki/ca-trust/extracted/pem/tls-ca-bundle.pem

root@ip-10-0-5-156:/cat /root/harbor/ssl/172.16.60.213.crt >> /etc/pki/tls/certs/ca-bundle.crt

root@ip-10-0-5-156:chmod 444 /etc/pki/ca-trust/extracted/pem/tls-ca-bundle.pem

Như vậy đề án đã thành công login vào hệ thống khi sử dụng lệnh. Sau khi khởi động lại docker-compose người dùng sẽ truy cập lại trang web để kiểm tra xem "https" đã hoạt động hay chưa. Sau khi vào trang Web mà thấy "https" đã hoạt động chứng tỏ rằng hệ thống đã áp dụng thành công SSL để tăng cường bảo mật.

Hình 3.13 Hình ảnh hệ thống sau khi đã cài đặt SSL

Khi sử dụng chứng chỉ SSL tự tạo trình duyệt thường hiển thị dấu gạch đỏ ngang chữ HTTPS. Lý do là chứng chỉ này không được cấp bởi một Cơ quan Chứng nhận uy tín, khiến trình duyệt không thể xác minh tính hợp lệ và danh tính của tên miền. Ở đề án lần này sẽ sử dụng chứng chỉ tự tạo vì vậy sau khi cài đặt hệ thống sẽ thấy dấu gạch đỏ hiển thị khi truy cập hệ thống.

3.6.4 Ngăn chặn thành công sau khi hệ thống đã cài đặt SSL

Sau khi cài đặt SSL thành công thử vào lại trang web và sử dụng Wireshark để theo dõi. Sau khi theo dõi thì có thể thấy rằng dữ liệu đã được mã hoá thành công. Điều này rất quan trọng trong thực tế, hiện tại dịch vụ của đề án đã có thể phòng chống những người có mục đích xấu.

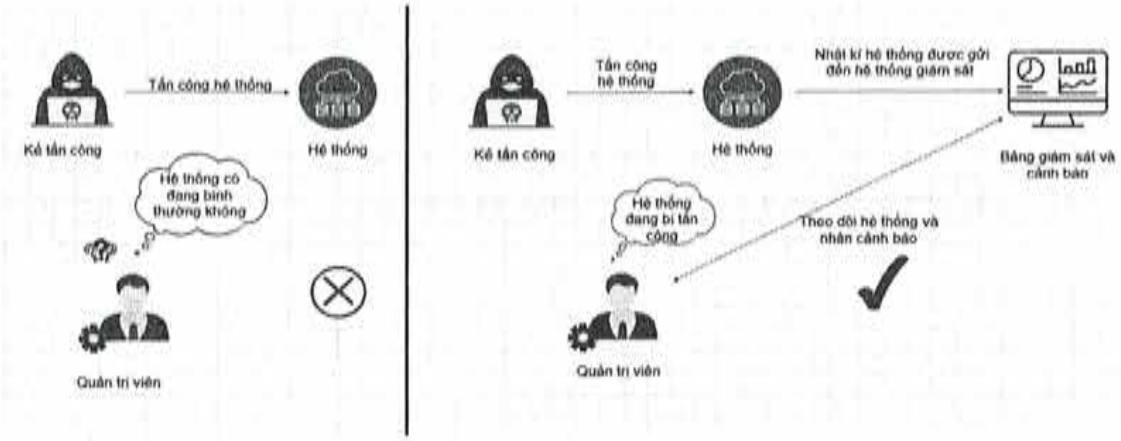
Hình 3.14 Hình ảnh dữ liệu đã được mã hoá sau khi cài đặt thành công SSL

Sau khi triển khai SSL, toàn bộ dữ liệu trao đổi trong hệ thống đều được mã hóa. Điều này đảm bảo tính bảo mật và toàn vẹn của dữ liệu, giúp các thông tin nhạy cảm không bị lộ lọt hoặc thay đổi trong quá trình truyền tải, góp phần bảo vệ quyền riêng tư của người dùng và uy tín của hệ thống.

3.7 Sử dụng Wazuh để theo dõi và cảnh báo khi có sự cố xảy ra

3.7.1 Khi hệ thống chưa có các công cụ theo dõi

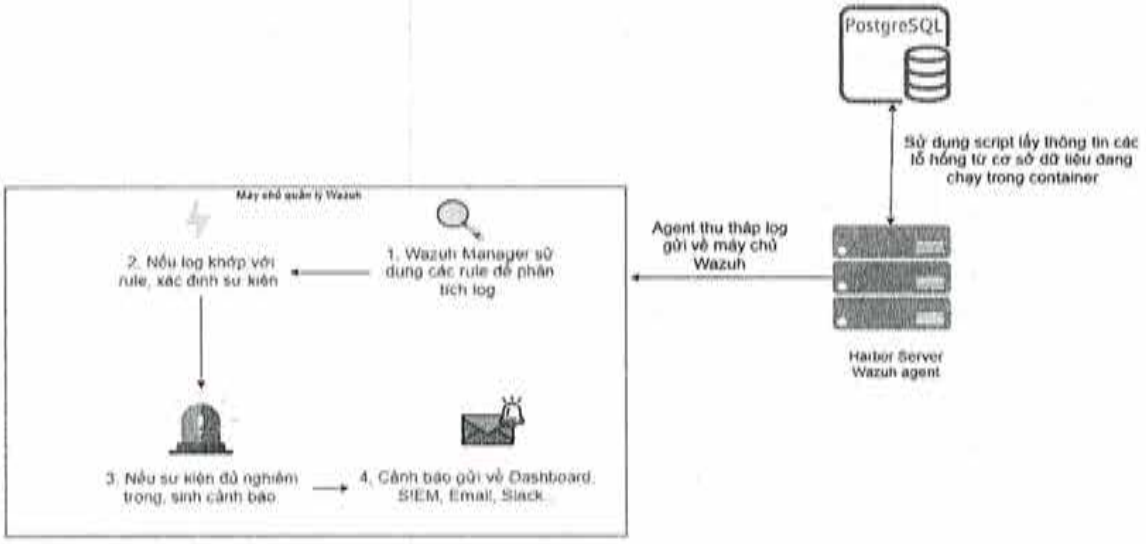
Khi hệ thống chưa được cài đặt các công cụ theo dõi như Wazuh. Các hoạt động bất thường như truy cập trái phép, cài đặt phần mềm độc hại hoặc thay đổi file hệ thống có thể diễn ra mà không bị phát hiện, dẫn đến nguy cơ dữ liệu bị đánh cắp hoặc hệ thống bị phá hoại mà quản trị viên không hề hay biết.



Hình 3.15 minh hoạ giữa việc có và không có các hệ thống giám sát

3.7.2 Cách Wazuh có thể phát hiện ra loại tấn công

Wazuh phát hiện các cuộc tấn công mạng bằng cách thu thập và phân tích dữ liệu từ hệ thống. Các tác nhân của Wazuh cài đặt trên máy chủ, máy trạm sẽ liên tục thu thập nhật ký hệ thống, theo dõi thay đổi tệp, hoạt động tiến trình. Dữ liệu này sau đó được gửi về máy chủ Wazuh, nơi nó được xử lý và phân tích bằng các bộ quy tắc. Những quy tắc này giúp Wazuh so sánh dữ liệu thu thập được với các hành vi tấn công đã biết. Wazuh còn giám sát tính toàn vẹn của tệp để cảnh báo khi có thay đổi trái phép và thực hiện phân tích hành vi để nhận diện những hoạt động bất thường so với mức hoạt động bình thường của hệ thống. Wazuh cung cấp cái nhìn toàn diện về bảo mật, giúp nhanh chóng phát hiện và ứng phó với các mối đe dọa.



Hình 3.16 Mô tả cách Wazuh phát hiện ra tấn công

Wazuh sử dụng tập luật để phân tích nhật ký (log) trên máy agent, phát hiện mối đe dọa và tạo cảnh báo khi phát hiện hành vi bất thường. Trước hết, chúng giúp phân tích nhật ký để xác định các mẫu hoặc hành vi đáng ngờ từ hệ thống, ứng dụng hay thiết bị mạng. Khi một sự kiện khớp với quy tắc, Wazuh sẽ tạo cảnh báo với mức độ nghiêm trọng từ 0 (bỏ qua) đến 16 (nghiêm trọng).

Luật Wazuh được viết trong tệp XML (thường là local_rules.xml):

<rule>: Định nghĩa quy tắc với id (100000-120000) và mức độ (0-16).

<if_sid> hoặc <if_group>: Liên kết luật cha hoặc nhóm luật.

<match> hoặc <regex>: Khớp mẫu log bằng biểu thức chính quy.

<description>: Mô tả luật.

<group>: Phân loại luật (syslog, sshd, pci_dss).

Vậy là đã có cái nhìn tổng quan về cách Wazuh phát hiện ra tấn công và cách cấu hình các luật, tiếp sau đây sẽ thực hiện cấu hình cho Wazuh để phát hiện ra các cuộc tấn công trước đã đề cập trước đó.

Cấu hình cho Wazuh phát hiện khi image có lỗ hổng.

Đầu tiên sau khi sử dụng các công cụ quét lỗ hổng để quét các image thì kết quả hiện tại đang được lưu trong cơ sở dữ liệu của Harbor. Chính vì vậy sẽ cần viết một đoạn mã, đầu tiên khai báo các giá trị cần thiết DB_CONTAINER (tên container chứa cơ sở dữ liệu của Harbor), DB_USER, DB_NAME, DB_TABLEE, COLUMNS (thông tin kết nối của cơ sở dữ liệu sau khi thực hiện vào container đó), OUTPUT_FILE (kết quả được lấy trong cơ sở dữ liệu sẽ được lưu vào file vulnerable.log), cuối cùng là câu lệnh sẽ thực thi vào container đang chạy cơ sở dữ liệu và lấy dữ liệu từ trong cơ sở dữ liệu với đầu ra là tệp vulnerable.log.

```
#!/bin/bash
DB_CONTAINER="harbor-db"
DB_USER="postgres"
DB_NAME="registry"
DB_TABLE="scan_report"
```

```
OUTPUT_FILE="vulnerable.log"
COLUMNS="report"
docker exec -i $DB_CONTAINER bash -c "psql -U $DB_USER -d
$DB_NAME -c \"\\copy (SELECT $COLUMNS FROM $DB_TABLE) TO
STDOUT\"" > $OUTPUT_FILE
```

Đoạn mã này có nhiệm vụ lấy dữ liệu từ trong cơ sở dữ liệu và ghi vào file vulnerable.log. file log sẽ có dạng:

```
{"generated_at":"2025-06-
11T03:26:53.338505716Z","scanner":{"name":"Trivy","vendor":"Aqua
Security","version":"v0.62.1"},"severity":"Critical","vulnerabilities":[]}.
```

Sẽ cần lập lịch cho đoạn mã này chạy và cập nhật dữ liệu mỗi phút. Chỉ cần cài đặt nó mỗi phút.

```
* * * * * /bin/bash /path/Vulnerable.sh
```

Tiếp theo sẽ cần thêm dòng sau để để Wazuh agent có thể gửi nhật ký đến Wazuh manager /var/osses/etc/osses.conf.

```
<localfile>
 <log_format>json</log_format>
 <location>/path/vulnerable.log</location>
 </localfile>
```

Sau khi thêm cần chạy lại Wazuh agent để cập nhật cấu hình mới: systemctl restart wazuh-agent.

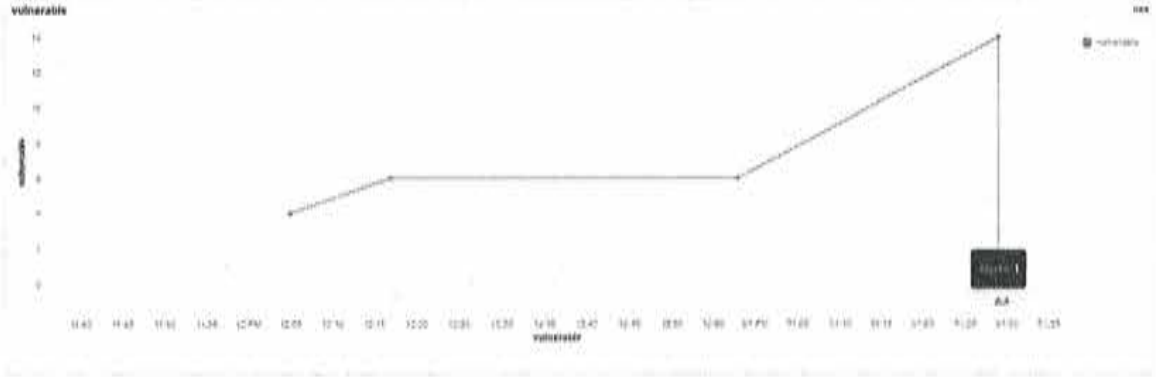
Trên máy chủ quản lý Wazuh sẽ tiến hành cấu hình các tập luật để xác định mức độ nghiêm trọng, các điều kiện chính, mô tả của nó.

```
<group name="serverity">
 <rule id="100002" level="10">
  <decoded_as>json</decoded_as>
  <field name=" severity">Critical</field>
  <description>severity log</description>
 </rule>
```

</group>

Sau khi thành công tiến hành chạy lại dịch vụ và tải lại trang giao diện của Wazuh: *systemctl restart wazuh-manager*.

Đây là những gì sẽ hiển thị trên bảng điều khiển của Wazuh. Có thể có một cái nhìn trực quan khi theo dõi nó. Chỉ cần quá một ngưỡng nhất định thì Wazuh sẽ hiện cảnh báo được mô tả rằng bởi những cột tia màu đỏ báo hiệu rằng số image đang bị lỗ hổng vượt quá ngưỡng chi tiết ở bên dưới hình như sau:



Hình 3.17 Bảng điều khiển Wazuh hiển thị cảnh báo

Cấu hình cho Wazuh phát hiện có người đăng nhập vào hệ thống không có SSL

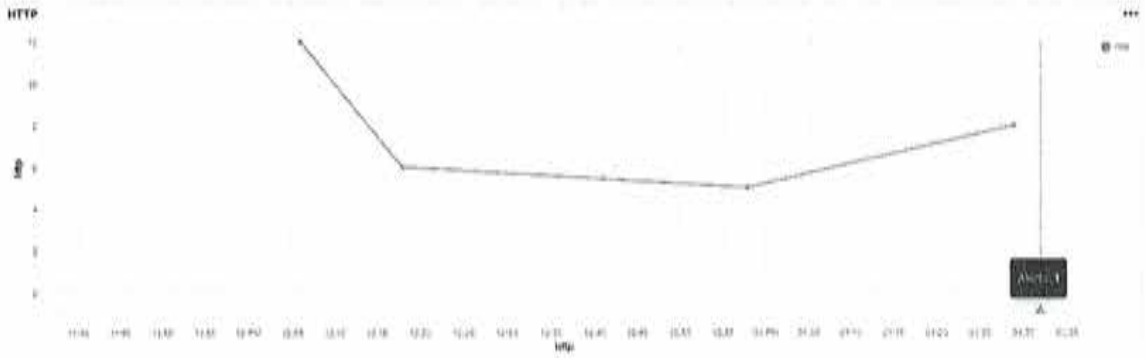
Cơ chế để Wazuh phát hiện ra người dùng không sử dụng HTTPS là: Thứ nhất đầu tiên để Wazuh phát hiện ra người dùng khi không sử dụng HTTPS để đăng nhập vào hệ thống thì chúng ta cần phải có nhật kí (log) được đẩy lên Wazuh. Tệp nhật kí */var/log/harbor/proxy.log* lưu lại lịch sử trang web khi sử dụng HTTP hoặc HTTPS. Tiếp theo đó chúng ta sẽ tiến hành cấu hình các tập luật cho wazuh nhận biết rằng khi có dòng log nào mà có thông tin "*http://18.140.220.206*" chứng tỏ người dùng đang không sử dụng HTTPS. Nếu người dùng sử dụng HTTPS thì log sẽ hiện ra là "*https://18.140.220.206*" những thông tin này hoàn toàn được lưu lại trong tệp nhật kí. Tiếp theo đây ta sẽ tiến hành cấu hình nó.

Máy chủ quản lý Wazuh cũng cần cấu hình các luật sau khi cấu hình xong cũng cần chạy lại Wazuh để cập nhật cấu hình mới.

<group name="web,attack,">

```
<rule id="100100" level="10">
 <match>http://18.140.220.206</match>
 <description>[Harbor] log in no https</description>
</rule>
</group>
```

Tiếp theo sẽ chuyển đến giao diện Wazuh để xem kết quả hiển thị trên bảng điều khiển. Hình vẽ mô tả tổng số người truy cập vào hệ thống mà không sử dụng HTTPS và cảnh báo được hiển thị lên khi số người dùng vượt quá ngưỡng.



Hình 3.18 Mô tả tổng người dùng đăng nhập không có HTTPS

Cấu hình cho Wazuh phát hiện có người dùng được trao quyền.

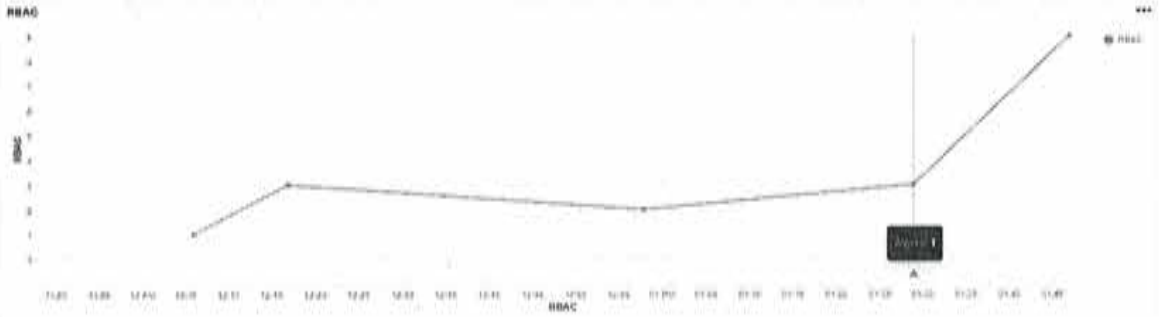
Tương tự ở trên chỉ cần truy xuất vào cơ sở dữ liệu để lấy dữ liệu và đưa về dạng json. Dữ liệu sau khi đưa về sẽ có dạng như sau.

```
{"user_id":1,"username":"admin","sysadmin_flag": true}
```

Lúc này chỉ cần thêm cấu hình các luật của máy chủ quản lý Wazuh để xác định mức độ nghiêm trọng.

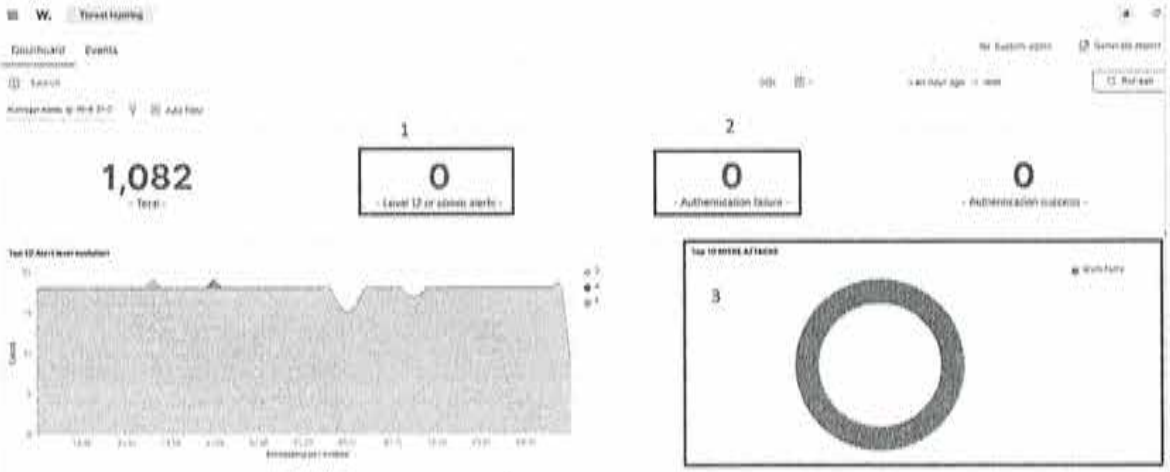
```
<group name="serverity,">
 <rule id="100008" level="10">
 <decoded_as>json</decoded_as>
 <field name="sysadmin_flag">true</field>
 <description>severity user permission</description>
 </rule>
</group>
```

Sau khi cấu hình xong tiến hành chạy lại dịch vụ wazuh và xem kết quả trên được hiển thị trên bảng điều khiển Wazuh.



Hình 3.19 Số lượng người dùng được tạo với quyền admin vượt quá ngưỡng

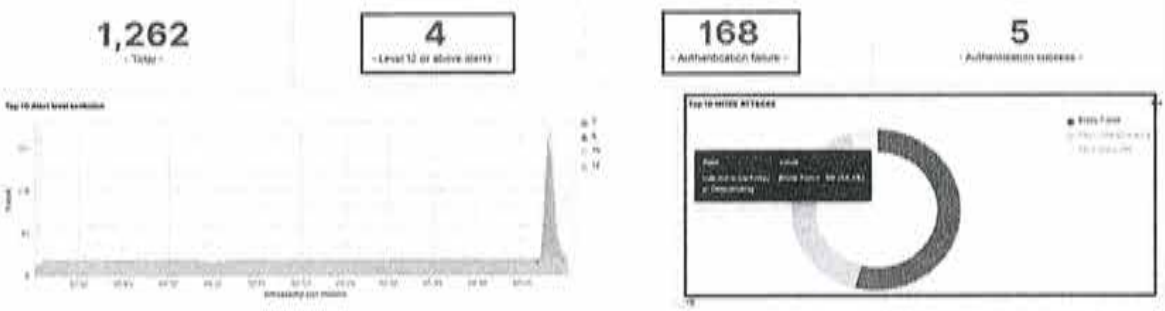
Tiếp theo sau ta sẽ thử một cách tấn công khác và xem trạng thái hiện tại hệ thống và sau khi đã sử dụng các cách tấn công để có thể thấy được Wazuh phát hiện và cảnh báo như thế nào.



Hình 3.20 Minh hoạ hệ thống khi đang ở trạng thái bình thường

- *Brute-force attack*: là một kỹ thuật tấn công mạng mà kẻ xấu sử dụng để cố gắng đăng nhập vào hệ thống bằng cách thử liên tục hàng nghìn, thậm chí hàng triệu, tổ hợp tên người dùng và mật khẩu khác nhau cho đến khi tìm ra bộ thông tin đăng nhập đúng. Đề án sẽ sử dụng hydra để tấn công:

hyra -l <Username> -P <password_list.txt> < Server_IP> ssh.



Hình 3.21 Giao diện monitor khi bị tấn công bằng Brute force

Nhìn vào hình vẽ trên có thể thấy rằng trong thời gian ngắn có khoảng 180 yêu cầu được gửi đến hệ thống và hệ thống ghi nhận xác thực không thành công tăng vọt hiện là 168 thành công chỉ có 5, mức độ của hành động tấn công trên mức cảnh báo là 4 lần, biểu đồ phân tích cũng đã chỉ ra phương pháp tấn công chính là Brute force với 99 lần. Ta cũng có thể xem chi tiết hơn bằng cách xem nhật kí được ghi lại bằng cách vào mục Discovery của Wazuh Explore.

Tương tự như vậy Wazuh có thể phát hiện ra các cuộc tấn công như SQL injection [22].

Hình 3.22 Hình ảnh minh hoạ nhật ký Wazuh phân loại tấn công Brute force

3.7.3 Thiết lập cảnh báo khi hệ thống bất thường

Sau khi đã tiến hành phân loại mức độ nghiêm trọng theo từng cấp độ, tiếp theo đây sẽ tiến hành cài đặt cảnh báo dựa trên những cấp độ đó. Wazuh có thể hát hiện các sự kiện đã biết hoặc các mối đe dọa được định nghĩa trước. Điều này cho phép nó nhận diện và cảnh báo ngay lập tức khi có bất kỳ sai lệch đáng kể nào so với

hành vi chuẩn, từ đó giúp phát hiện sớm các dấu hiệu của tấn công, lỗi cấu hình, hoặc các vấn đề tiềm ẩn.

Cảnh báo email Wazuh không hỗ trợ máy chủ SMTP có xác thực, chẳng hạn như Gmail. Tuy nhiên, có thể gửi những email này qua một máy chủ chuyển tiếp như Postfix, có thể tham khảo các bước cài đặt [21]. Điều chú ý nhất là cấu hình gửi email và cài đặt ngưỡng gửi cảnh báo tại đường dẫn */var/ossec/etc/ossec.conf*.

```
<ossec_config>
  <global>
  <email_notification>yes</email_notification>
  <email_from><USERNAME>@gmail.com</email_from>
  <email_to><RECEIVER_EMAIL></email_to>
</global>
<alerts>
   <log_alert_level>3</log_alert_level>
   <email_alert_level>8</email_alert_level>
  </alerts>
</ossec_config>
```

Thẻ <log_alert_level>đặt mức độ nghiêm trọng tối thiểu để kích hoạt cảnh báo, thẻ <email_alert_level>đặt mức độ nghiêm trọng tối thiểu cho cảnh báo để tạo thông báo qua email có thể cài đặt nó theo ngưỡng mà muốn.

Hình 3.23 Hình vẽ minh hoạ cảnh báo đã được gửi về email

Sau khi cài đặt mỗi khi hệ thống ghi nhận một hành động có mức độ nghiêm trọng từ 8 trở lên sẽ gửi về email để cảnh báo người quản trị hệ thống.

3.8 Kết luận Chương III

Trong chương III, đề án đã phân tích và triển khai các giải pháp bảo mật cho hệ thống Private Docker Registry khi sử dụng Harbor. Những biện pháp bảo mật như không chạy container dưới quyền root, giới hạn tài nguyên và kiểm tra lỗ hổng bảo mật trước khi triển khai đã được nhấn mạnh như là những hoạt động cần thiết để bảo vệ hệ thống khỏi các mối đe dọa tiềm ẩn.

Việc thiết lập các cơ chế kiểm soát quyền truy cập không chỉ giúp hạn chế rủi ro từ các hành vi trái phép mà còn nâng cao khả năng quản lý tài nguyên trong hệ thống. Các mối đe dọa như truy cập trái phép và tấn công MITM cũng đã được phân tích, cho thấy sự cần thiết của việc áp dụng các biện pháp bảo mật hiệu quả.

Đề án đã đề xuất các giải pháp bảo mật cụ thể, bao gồm việc sử dụng RBAC, SSL để mã hóa dữ liệu và tích hợp các công cụ quét lỗ hổng như Trivy. Những giải pháp này giúp đảm bảo rằng chỉ những image an toàn mới được triển khai vào môi trường sản xuất, từ đó giảm thiểu nguy cơ xâm nhập và lỗ hổng bảo mật. Bên cạnh đó, việc sử dụng Wazuh để theo dõi và cảnh báo khi có sự cố xảy ra sẽ giúp quản trị viên phát hiện sớm các hoạt động bất thường và phản ứng kịp thời.

Việc triển khai các giải pháp bảo mật hiệu quả cho Private Docker Registry là rất cần thiết để bảo vệ tài sản mã nguồn và dữ liệu nhạy cảm. Sự kết hợp giữa các biện pháp bảo mật chặt chẽ, công cụ giám sát mạnh mẽ và quy trình phát triển phần mềm an toàn sẽ tạo ra một môi trường an toàn, giúp tổ chức duy trì uy tín và độ tin cậy trong mắt khách hàng và đối tác.

CHƯƠNG IV ĐÁNH GIÁ HIỆU SUẤT VÀ KHẢ NĂNG MỞ RỘNG CỦA PRIVATE DOCKER REGISTRY KHI SỬ DỤNG HARBOR

4.1 Các tiêu chí đánh giá hiệu suất

4.1.1 Thời gian push/pull container image.

Hiện này chưa có một chuẩn nào dùng để đánh giá hiệu suất của Docker Registry. Nhưng để đánh giá một cách khách quan hiệu suất của Docker Registry ta có thể xem xét một vài tiêu chí như thời gian push/pull các image lên hệ thống. Các tiêu chí này giúp định lượng và so sánh hiệu suất trong các điều kiện và cấu hình khác nhau [33].

Push Time: Đây là khoảng thời gian cần thiết để một client tải một image container lên registry. Thời gian push có thể bị ảnh hưởng bởi kích thước của image, tốc độ mạng giữa client và registry, và loại lưu trữ.

Pull Time: Đây là khoảng thời gian cần thiết để một client tải một image container từ registry xuống. Thời gian pull ảnh hưởng đến tốc độ triển khai ứng dụng. Các yếu tố ảnh hưởng đến thời gian kéo bao gồm kích thước image, vị trí của registry, và số lượng layer trong image. Tương tự như thời gian push/pull trung bình cũng có xu hướng tăng khi số lượng client đồng thời tăng lên.

4.1.2 Hiệu năng khi số lượng image tăng lên

Số lượng image và kho lưu trữ trong Harbor Registry có thể có tác động đáng kể đến hiệu suất hoạt động của nó.

Tác động đến thao tác Push: Với số lượng image lớn, thao tác push có thể bị ảnh hưởng. Mặc dù Harbor được thiết kế để xử lý khối lượng lớn, nhưng việc có quá nhiều image có thể làm tăng tải cho thành phần core của Harbor, đặc biệt khi có nhiều thao tác push đồng thời.

Tác động đến thao tác Pull: Số lượng image lớn cũng có thể ảnh hưởng đến hiệu suất khi pull. Khi một khách hàng yêu cầu một image, registry cần tìm kiếm và truy xuất các lớp cần thiết. Mặc dù các registry hiện đại sử dụng các cơ chế bộ nhớ đệm và đánh vị trí để tối ưu hóa quá trình này, nhưng với số lượng image khổng lồ, thời gian tìm kiếm và truy xuất có thể tăng lên.

Yếu tố lưu trữ: Số lượng image lớn đồng nghĩa với yêu cầu về dung lượng lưu trữ lớn. Hiệu suất của server lưu trữ (ví dụ: ổ đĩa cục bộ hoặc dịch vụ lưu trữ đám mây) sẽ trực tiếp ảnh hưởng đến hiệu suất tổng thể của registry.

4.2 Thử nghiệm và đánh giá

So sánh hiệu suất trong các trường hợp khác nhau

Đầu tiên xem thời gian push và và pull khi có ít image trên kho lưu trữ với hệ điều hành là ubuntu 22.04. Giả sử có những image sau và tôi sẽ lần lượt push chúng lên kho lưu trữ:

Bảng 4.1 Thông tin của các image sẽ thử nghiệm

REPOSITORY	TAG	IMAGE ID	SIZE
node	latest	b1e1dcf10eb9	1.13GB
golang	latest	5669eb73fcac	853MB
postgres	latest	76e3e031d245	438MB
alpine	latest	aded1e1a5b37	7.83MB
nginx	latest	53a18edff809	192MB

Bảng 4.2 Thời gian push image lên các loại lưu trữ khác nhau

Lần	Tên image được push lên hệ thống	Thời gian push image lên kho lưu trữ đám mây (S3)	Thời gian push image lên kho lưu trữ cục bộ
1	Node	1m55.112s	1m28.118s

2	Golang	0m56.438s	0m47.763s
3	Postgres	0m31.272s	0m27.928s
4	Alpine	0m7.595s	0m0.984s
5	Nginx	0m16.128s	0m8.298s

Số lượng image ít và nhiều

Bảng trên là lượt lần lượt thời gian push của image, nhưng có một điều đặc biệt là các image được push lên trước có các lớp trùng với image được push lên sau thì nó sẽ được tận dụng các lớp đó mà không cần phải push lên kho lưu trữ tiếp nữa. Ví dụ như lần push thứ 2 khi đề án push image golang lên kho lưu trữ sau khi push image node thì mất 0m47.763s. Nếu mà chỉ push image golang lên kho mà không có các image lúc trước đấy thì sẽ mất nhiều thời gian hơn do các lớp của image golang chưa có sẵn trên hệ thống.

Như vậy nếu có kho lưu trữ đủ lớn và đường mạng ổn định thì việc push các image mới lên sẽ nhanh hơn khi push image đó lần đầu lên hệ thống.

Lưu trữ trên địa cục bộ và lưu trữ đám mây

Khi sử dụng lưu trữ địa phương, các image được lưu trữ trực tiếp trên máy chủ chạy Harbor, mang lại hiệu suất cao và dễ dàng quản lý. Ngược lại, lưu trữ đám mây, như Amazon S3, cung cấp khả năng mở rộng và tính linh hoạt, cho phép người dùng lưu trữ khối lượng lớn image mà không cần phải đầu tư vào hạ tầng vật lý. Bằng cách thay đổi cấu hình trong file harbor.yml, người dùng có thể dễ dàng chuyển đổi giữa hai phương thức lưu trữ này. Dưới đây là một cấu hình đơn giản của trong việc thay đổi phương thức lưu trữ, chỉ cần thêm nó vào trong cấu hình "storage_service" trong file harbor.yml.

```
s3:
  accesskey: xxx
  secretkey: xxx
  region: ap-southeast-1
```

bucket: testhuytam

secure: true

rootdirectory: /harbor

Hình 4.1 Hình ảnh các image được lưu trữ trên kho lưu trữ khi sử dụng kho lưu trữ đám mây

Đo lường hiệu suất khi có nhiều người truy cập

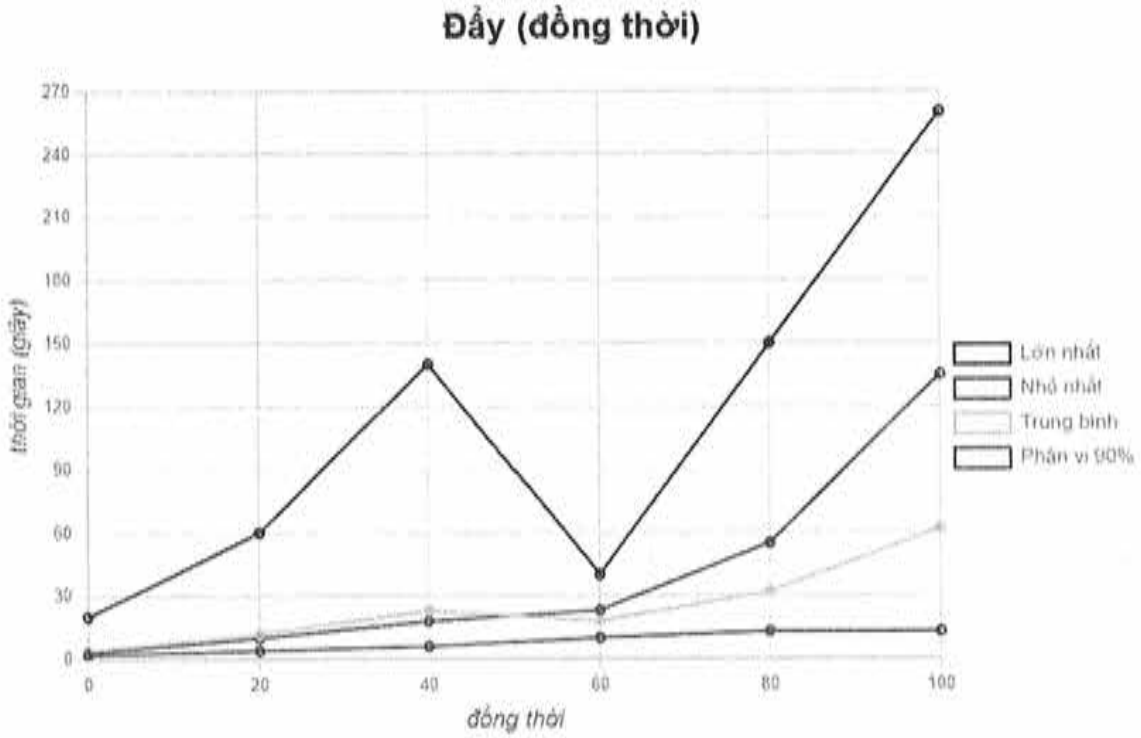
Việc nhiều người dùng truy cập đồng thời vào Harbor Registry để push image có thể gây ra những tác động đáng kể đến hiệu suất. Khi số lượng người dùng đồng thời tăng lên, hệ thống phải xử lý một lượng lớn yêu cầu, điều này có thể dẫn đến nghẽn mạng, tăng tải CPU và bộ nhớ trên máy chủ Harbor trên máy chủ lưu trữ. Thời gian push tăng lên khi số lượng người dùng đồng thời tăng lên. Hiệu suất có thể bị ảnh hưởng nghiêm trọng trong các tình huống có tính đồng thời cao. Để xử lý tình huống này, các biện pháp như tăng max_connections của Postgres và tăng số lượng instance của thành phần core của Harbor đã được đề xuất.

Dưới đây là một thực nghiệm đã được tiến hành để đo giá trị thời gian của một registry trong trường hợp khi số lượng người dùng đồng thời tăng lên [33]. Thí nghiệm trên được thực hiện trong môi trường như sau: Hệ điều hành (Ubuntu-trusty x86_64), 256GB RAM, 12 CPU, 3,6 TB lưu trữ, bang thông (bandwidth) là 10G [33].

Khi số lượng đồng thời tăng lên thời gian push của image cũng tăng lên. Thực nghiệm đo giá trị của thời gian tối đa, tối thiểu, trung bình, và phân vị 90% (thời gian phản hồi của 90% yêu cầu đạt được).

Bảng 4.3 Giá trị Tối đa, Tối thiểu, Trung bình của các giá trị thời gian push phụ thuộc vào tham số "đồng thời" [33].

Đồng thời	Tối đa	Tối thiểu	Trung Bình	Phân vị 90%
1	18.23183703	2.014497995	2.852927562	2.120845795
10	51.36455989	4.625913858	6.886669915	4.924576068
30	143.376904	14.23889208	20.4385057	14.57682798
50	45.15124679	21.27197409	24.59056571	24.24201851
100	254.9175169	20.78799295	66.44495539	133.36117



Hình 4.2 Đồ thị cho thấy tham số "thời gian push" phụ thuộc vào tham số "đồng thời" [33].

Để đảm bảo Harbor có thể xử lý một số lượng lớn người dùng đồng thời một cách hiệu quả, có thể áp dụng các chiến lược mở rộng sau:

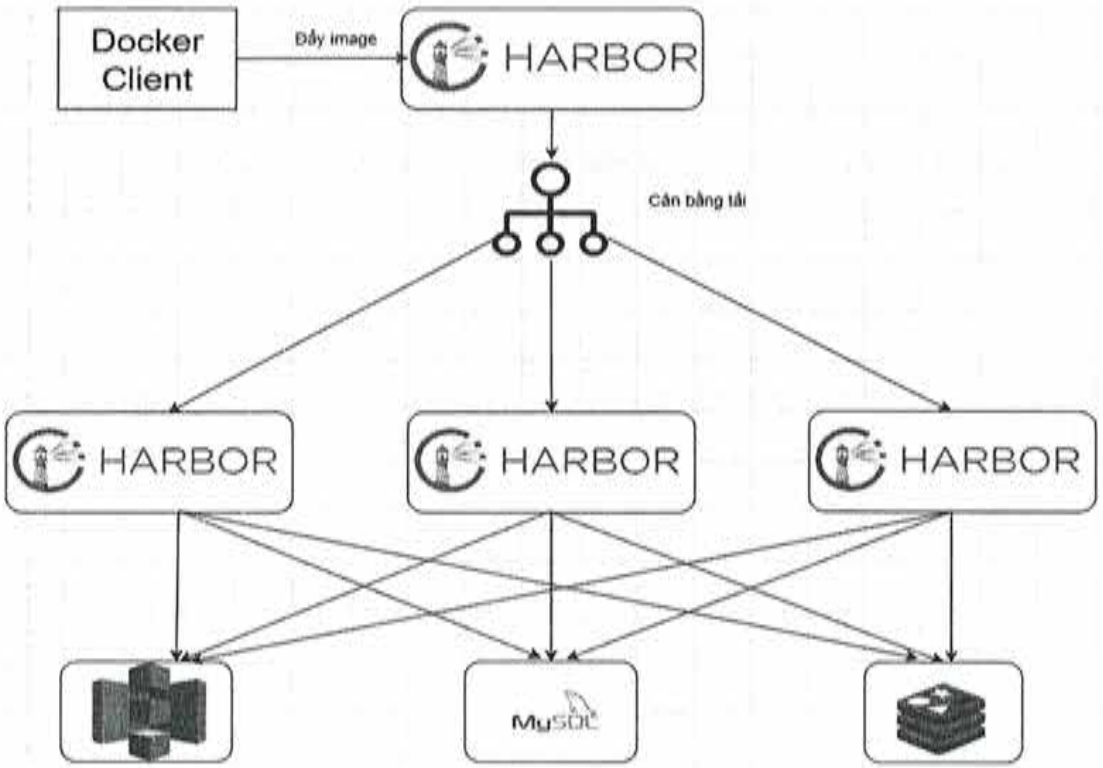
Tối ưu hóa cơ sở hạ tầng cơ bản: Đảm bảo rằng cơ sở hạ tầng mạng, lưu trữ và tính toán có đủ tài nguyên để đáp ứng nhu cầu của số lượng lớn người dùng đồng thời. Điều này có thể bao gồm việc sử dụng mạng có băng thông cao hơn, máy chủ lưu trữ hiệu suất cao và máy chủ có đủ CPU và bộ nhớ.

Ước tính nhu cầu về số lượng người dùng đồng thời: Dựa trên phân tích lưu lượng truy cập web hoặc các mẫu sử dụng dự kiến, ước tính số lượng người dùng đồng thời có thể truy cập hệ thống và lên kế hoạch về tài nguyên và khả năng mở rộng cho phù hợp.

Việc truy cập đồng thời có tác động đáng kể đến hiệu suất của Harbor, dẫn đến thời gian push tăng lên. Mở rộng Harbor theo chiều ngang bằng cách tăng số lượng máy chủ của các thành phần và đảm bảo đủ tài nguyên cơ sở hạ tầng là rất quan trọng để duy trì hiệu suất dưới tải lượng người dùng cao.

4.3 Giải pháp mở rộng:

Tính sẵn sàng cao là yếu tố then chốt đối với các hệ thống trong môi trường sản xuất để đảm bảo dịch vụ liên tục và ngăn ngừa thời gian chết. Các thành phần không trạng thái không lưu trữ bất kỳ trạng thái phiên nào, do đó có thể dễ dàng được thay thế hoặc nhân rộng mà không ảnh hưởng đến tính nhất quán của hệ thống.



Hình 4.3 Sơ đồ minh hoạ sử dụng bản sao để tăng tính sẵn sàng

Khi triển khai giải pháp này trong môi trường sản xuất thực tế, có ba vấn đề chính cần lưu ý. Việc lựa chọn loại lưu trữ chia sẻ phù hợp là rất quan trọng. Harbor hiện hỗ trợ các bộ lưu trữ phụ trợ như AWS S3. Liên quan đến cơ sở dữ liệu đa phiên bản Harbor, chỉ cần tách riêng cơ sở dữ liệu và triển khai độc lập bên ngoài. Nhiều phiên bản Harbor có thể cùng sử dụng một cơ sở dữ liệu bên ngoài, và tính khả dụng cao của cơ sở dữ liệu này có thể được đảm bảo bằng các giải pháp HA cho MySQL.

4.4 Kết luận Chương IV

Trong chương IV, đề án đã đánh giá hiệu suất và khả năng mở rộng của Private Docker Registry. Để thực hiện điều này, đề án đã thiết lập các tiêu chí đánh giá rõ ràng, bao gồm thời gian push/pull container image, cũng như hiệu suất hoạt động khi số lượng image tăng lên. Qua các thử nghiệm, thấy rằng thời gian push/pull ảnh hưởng rất lớn bởi kích thước của image và cấu hình lưu trữ. Điều này nhấn mạnh tầm quan trọng của việc chọn lựa phương thức lưu trữ phù hợp, giữa lưu trữ cục bộ và đám mây, để tối ưu hóa hiệu suất.

Bên cạnh đó, đề án cũng đã thảo luận về tác động của việc nhiều người dùng truy cập đồng thời đến hiệu suất của Harbor. Vì vậy, việc tối ưu hóa cơ sở hạ tầng và ước tính nhu cầu sử dụng là rất cần thiết để đảm bảo khả năng mở rộng trong môi trường sản xuất.

Cuối cùng, chương này đã giới thiệu các giải pháp mở rộng, đặc biệt là việc sử dụng bản sao để tăng tính khả dụng cho hệ thống. Như vậy, việc tối ưu hóa hiệu suất và khả năng mở rộng cho Private Docker Registry không chỉ giúp cải thiện quy trình phát triển phần mềm mà còn nâng cao tính sẵn sàng và độ tin cậy của dịch vụ.

KẾT LUẬN

1 Tóm tắt kết quả đạt được

Sau quá trình nghiên cứu và triển khai đề tài "Nghiên cứu giải pháp đảm bảo an toàn cho hệ thống Private Docker Registry của tổ chức". Cụ thể, đề tài đã xây dựng thành công một hệ thống Private Docker Registry an toàn và hiệu quả, đáp ứng nhu cầu quản lý image container nội bộ cho các tổ chức và doanh nghiệp.

Đóng góp nổi bật đầu tiên là việc hệ thống hóa kiến thức nền tảng về Docker và công nghệ container, giúp làm rõ các khái niệm, vai trò, ưu điểm và thách thức khi so sánh với mô hình máy ảo truyền thống. Đây là cơ sở quan trọng để triển khai giải pháp một cách đúng đắn và có định hướng.

Thứ hai, đề án đã xây dựng và triển khai thành công hệ thống Private Docker Registry sử dụng Harbor là một giải pháp mạnh mẽ với nhiều tính năng bảo mật và quản lý. Toàn bộ quá trình từ xác định yêu cầu hệ thống, cài đặt, cấu hình, tích hợp với quy trình phát triển phần mềm, đến vận hành đều được trình bày rõ ràng, minh chứng cho tính khả thi của mô hình.

Đặc biệt, đề án đã tập trung vào các biện pháp bảo mật toàn diện:

- Áp dụng công cụ Trivy để quét và phát hiện lỗ hổng trong image container.
- Thiết lập RBAC nhằm phân quyền truy cập chi tiết, giảm thiểu rủi ro truy cập trái phép.
- Mã hóa giao tiếp bằng SSL, giúp bảo vệ dữ liệu khỏi các cuộc tấn công trung gian (MITM).
- Tích hợp hệ thống giám sát an ninh Wazuh, cung cấp khả năng thu thập nhật ký, phát hiện xâm nhập, phân tích bất thường và cảnh báo thời gian thực.

Trong đó, Wazuh đóng vai trò quan trọng trong việc giám sát liên tục và bảo vệ hệ thống, giúp phát hiện và phản ứng kịp thời với các mối đe dọa như truy cập trái phép, thay đổi cấu hình, hoặc hành vi bất thường.

Bên cạnh đó, đề tài cũng đánh giá hiệu suất, khả năng mở rộng của hệ thống Private Docker Registry, chỉ ra tầm quan trọng của tối ưu hóa hạ tầng, lựa chọn phương thức lưu trữ phù hợp và khả năng đáp ứng khi có nhiều người dùng đồng thời.

Với những đóng góp trên, đề tài không chỉ hoàn thành mục tiêu đề ra mà còn đưa ra một mô hình giải pháp có tính ứng dụng cao, góp phần quan trọng vào việc bảo vệ tài sản mã nguồn, nâng cao hiệu suất làm việc, và xây dựng nền tảng bảo mật bền vững cho các hệ thống sử dụng Docker trong doanh nghiệp.

2 Hướng phát triển

Trong tương lai, việc phát triển hệ thống Private Docker Registry có thể tập trung vào việc tích hợp sâu rộng hơn vào hệ sinh thái DevOps, đặc biệt là các quy trình tự động hóa triển khai phần mềm thông qua pipeline CI/CD (Continuous Integration/Continuous Deployment). Sự tích hợp này sẽ cho phép tự động hóa các bước kiểm thử, quét lỗ hổng, và triển khai image container, từ đó tối ưu hóa quy trình phát triển, giảm thời gian đưa sản phẩm ra thị trường, và tăng cường tính nhất quán giữa các môi trường phát triển và sản xuất. Ngoài ra, việc nâng cấp hệ thống với các tính năng bảo mật tiên tiến có thể mang lại mức độ bảo mật cao hơn. Đồng thời, việc mở rộng khả năng sẵn sàng và triển khai trên các đám mây lai (hybrid cloud) sẽ đáp ứng nhu cầu của các doanh nghiệp lớn, đảm bảo tính sẵn sàng cao và khả năng phục hồi khi gặp sự cố. Hướng phát triển này không chỉ tối ưu hóa quy trình làm việc mà còn tăng cường kiểm soát chặt chẽ các thành phần phần mềm, bảo vệ dữ liệu nhạy cảm, và hỗ trợ doanh nghiệp thích nghi với các xu hướng công nghệ mới trong tương lai.

DANH MỤC TÀI LIỆU THAM KHẢO

Tài liệu, giáo trình:

[1]. https://www.practical-devsecops.com/container-security-risks/ ,truy cập ngày 05/04/2025.

[2]. Experimental Analysis of Security Attacks for Docker Container Communications (Haneul Lee, Soonhong Kwon and Jong-Hyouk Lee).

[3]. https://iterasec.com/blog/container-security-vulnerabilities/ ,Truy cập ngày 20/02/2025.

[4]. An analysis of security vulnerabilities in container images for scientific data analysis (Bhupinder Kaur, Mathieu Dugre, Aiman Hanna and Tristan Glatard).

[5]. https://cheatsheetseries.owasp.org/ , truy cập ngày 23/02/2025.

[6]. https://github.com/wagoodman/dive , truy cập ngày 30/03/2025.

[7]. https://danaepp.com/finding-api-secrets-in-hidden-layers-within-docker-containers , truy cập ngày 30/03/2025.

[8]. https://sysdig.com/learn-cloud-native/what-is-a-dos-attack/ , truy cập ngày 05/04/2025.

[9]. https://security.snyk.io/vuln/SNYK-GOLANG-GITHUBCOMDOCKERDISTRIBUTIONREGISTRYAPIV2-5885037, truy cập ngày 05/04/2025.

[10]. https://goharbor.io/docs/main/administration/managing-users/user-permissions-by-role/, truy cập ngày 05/04/2025.

[11]. https://docs.docker.com/reference/cli/docker/container/run/#set-ulimits-in-container---ulimit , truy cập ngày 05/04/2025.

[12]. https://github.com/goharbor/harbor/releases, truy cập ngày 05/05/2025.

[13]. https://sysdig.com/blog/2022-cloud-native-security-usage-report/ , truy cập ngày 05/05/2025.

[14]. Docker Official Documentation, (2023). "Docker Registry", Docker Official Documentation, [truy cập ngày 04tháng 12 năm 2024]. https://docs.docker.com/registry/.

[15]. Harbor, (2023). "Harbor", Harbor Documentation, [truy cập ngày 04 tháng 12 năm 2024]. https://goharbor.io/.

[16]. Nigel Poulton, (2020). Docker Deep Dive, 2nd Edition, Packt Publishing, 326 trang.

[17]. Liz Rice, (2020). Container Security, O'Reilly Media, Inc, 1-120

[18]. Ian Miell, Aidan Hobson Sayers, Docker in Practice (2019). Manning Publications UK, 19-45.

[19]. https://3ac.vn/tam-giac-bao-mat-cia-tinh-bao-mat-tinh-toan-ven-tinh-san-sang-la-gi/, truy cập ngày 13/05/2025.

[20]. https://www.upguard.com/blog/defense-in-depth, truy cập ngày 17/05/2025.

[21]. https://documentation.wazuh.com/current/user-manual/manager/alert-management.html#smtp-server-with-authentication. , truy cập ngày 18/05/2025.

[22]. https://documentation.wazuh.com/current/proof-of-concept-guide/detect-web-attack-sql-injection.html , truy cập ngày 20/05/2025.

[23]. https://owasp.org/www-project-docker-top-10/., truy cập ngày 20/05/2025.

[24]. https://docs.docker.com/get-started/docker-overview/,truy cập ngày 25/05/2025.

[25]. https://github.com/aquasecurity/trivy, truy cập ngày 26/05/2025.

[26]. https://www.aquasec.com/cloud-native-academy/docker-container/docker-registry/, truy cập ngày 22/05/2025.

[27]. Tasneem Salah, M. Jamal Zemerly, Chan Yeob Yeu, Mahmoud Al-Qutayri (2017). Performance comparison between container-based and VM-based services, IEEE.

[28]. Mauro Conti, Nicola Dragoni, Viktor Lesyk (2016) A Survey of Man In The Middle Attacks, IEEE, 2027-2051.

[29]. Sirikarn Pukkawanna, Gregory Blanc, Joaquin Garcia-Alfaro, Youki Kadobayashi, Herve Debar (2014). Classification of SSL Servers based on their SSL Handshake for Automated Security Assessment, IEEE.

[30]. https://phoenixnap.com/kb/set-up-a-private-docker-registry, truy cập 30/05/2025.

[31]. https://arnold101.hashnode.dev/docker-private-registry-for-corporates, truy cập ngày 30/05/2025.

[32]. https://www.cloudthat.com/resources/blog/securing-container-workflows-with-a-private-docker-registry, truy cập ngày 30/05/2025.

[33]. https://docs.openstack.org/developer/performance-docs/test_results/container_repositories/registry2/ , truy cập ngày 02/06/2025.

BẢN CAM ĐOAN

Tôi xin cam đoan đã thực hiện kiểm tra mức độ tương đồng nội dung đề án qua phần mềm kiểm tra tài liệu một cách trung thực và đạt kết quả mức độ tương đồng 05 % toàn bộ nội dung đề án tốt nghiệp. Bản đề án tốt nghiệp kiểm tra qua phần mềm là bản cứng đề án tốt nghiệp đã nộp để bảo vệ trước hội đồng. Nếu sai tôi xin chịu các hình thức kỷ luật theo quy định hiện hành của Học viện.

Hà Nội, ngày 28 tháng 07 năm 2025 HỌC

VIÊN CAO HỌC

(Ký và ghi rõ họ tên)

Nguyễn Huy Tâm

KiểmTraTàiLiệu

BÁO CÁO KIỂM TRA TRÙNG LẶP

Thông tin tài liệu

Tên tài liệu:	Nguyễn Huy Tâm_Nghiên cứu giải pháp đảm bảo an toàn cho hệ thống Private Docker Registry của tổ chức
Tác giả:	Nguyễn Huy Tâm
Điểm trùng lặp:	5
Thời gian tải lên:	21:48 28/07/2025
Thời gian sinh báo cáo:	21:50 28/07/2025
Các trang kiểm tra:	111/111 trang

Kết quả kiểm tra trùng lặp

Có 5% nội dung trùng lặp

Có 95% nội dung không trùng lặp

Có 0% nội dung người dùng loại trừ

Có 0% nội dung hệ thống bỏ qua

Nguồn trùng lặp tiêu biểu

123docz.net viblo.asia infohub.delltechnologies.com

Học viên
(Ký và ghi rõ họ tên)

Nguyễn Huy Tâm

Người hướng dẫn khoa học
(Ký và ghi rõ họ tên)

Đinh Thị Thúy Hằng

BỘ KHOA HỌC VÀ CÔNG NGHỆ
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

BÁO CÁO GIẢI TRÌNH
SỬA CHỮA, HOÀN THIỆN ĐỀ ÁN TỐT NGHIỆP

Họ và tên học viên: Nguyễn Huy Tâm

Chuyên ngành: HTTT

Khóa: 2023 đợt 2

Tên đề tài: Nghiên cứu giải pháp đảm bảo an toàn cho hệ thống Private Docker Registry của tổ chức

Người hướng dẫn khoa học: TS. Đinh Trường Duy

Ngày bảo vệ: 19/07/2025

Các nội dung học viên đã sửa chữa, bổ sung trong đề án tốt nghiệp theo ý kiến đóng góp của Hội đồng chấm đề án tốt nghiệp:

TT	Ý kiến hội đồng	Sửa chữa của học viên
1	Chỉnh sửa lỗi soạn thảo, lỗi ngữ pháp, chính tả,	Học viên đã rà soát, chỉnh sửa các lỗi soạn thảo, các lỗi ngữ pháp.
2	Chỉnh sửa lỗi thuật ngữ kỹ thuật, tài liệu tham khảo.	Học viên đã rà soát, chỉnh sửa các lỗi về trích dẫn tài liệu, về các thuật ngữ như kéo/đẩy, tấn công vũ lực.
3	Làm rõ cơ chế quét lỗ hổng bảo mật của Harbor	Tiếp thu góp ý của Hội đồng, tác giả đã bổ sung thêm cơ chế quét lỗ hổng tại mục 3.4.4 trong chương 3.
4	Mô tả chi tiết môi trường đánh giá hiệu suất của Private Docker Registry khi sử dụng Harbor	Tiếp thu góp ý của Hội đồng, tác giả đã bổ sung thêm mô tả môi trường đánh giá hiệu suất khi đo lường hiệu suất có nhiều người truy cập tại mục 4.2 trong chương 4.
5	Đặc điểm của tổ chức có gì khác biệt so với các thực thể khác khi triển khai áp dụng Private Docker Registry.	Tiếp thu góp ý của Hội đồng, tác giả đã bổ sung thêm các đặc điểm của tổ chức tại mục mở đầu.

6	Cơ chế để công cụ Wazuh phát hiện ra những người dùng đăng nhập vào hệ thống mà không sử dụng https.	Tiếp thu góp ý của Hội đồng, tác giả đã bổ sung cơ chế Wazuh phát hiện ra người dùng đăng nhập vào hệ thống tại mục cấu hình phát hiện có người đăng nhập vào hệ thống không có SSL tại mục 3.7.2 trong chương 3.

Hà Nội, ngày tháng năm 2025

Ký xác nhận của

CHỦ TỊCH HỘI ĐỒNG CHẤM ĐỀ ÁN	THƯ KÝ HỘI ĐỒNG	NGƯỜI HƯỚNG DẪN KHOA HỌC	HỌC VIÊN
PGS.TSKH Hoàng Đăng Hải	TS. Đào Ngọc Phong	TS. Đinh Trường Duy	Nguyễn Huy Tâm

BỘ KHOA HỌC VÀ CÔNG NGHỆ
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

BIÊN BẢN
HỌP HỘI ĐỒNG CHẤM ĐỀ ÁN TỐT NGHIỆP THẠC SĨ

Căn cứ quyết định số Quyết định số 1098/QĐ-HV ngày 26 tháng 06 năm 2025 của Giám đốc Học viện Công nghệ Bưu chính Viễn thông về việc thành lập Hội đồng chấm đề án tốt nghiệp thạc sĩ. Hội đồng đã họp vào hồi..9....giờ.10..phút, ngày 19 tháng 07 năm 2025 tại Học viện Công nghệ Bưu chính Viễn thông để chấm đề án tốt nghiệp thạc sĩ cho:

Học viên: **Nguyễn Huy Tâm**

Tên đề án tốt nghiệp: **Nghiên cứu giải pháp đảm bảo an toàn cho hệ thống Private Docker Registry của tổ chức**

Chuyên ngành: **Hệ thống thông tin** Mã số: **8480104**

Các thành viên của Hội đồng chấm đề án tốt nghiệp có mặt: ...05.../ 05

TT	HỌ VÀ TÊN	TRÁCH NHIỆM TRONG HĐ	GHI CHÚ
1	**PGS.TSKH. Hoàng Đăng Hải**	Chủ tịch	
2	**TS. Đào Ngọc Phong**	Thư ký	
3	**TS. Bùi Hải Phong**	Phản biện 1	
4	**PGS.TS. Trần Nguyên Ngọc**	Phản biện 2	
5	**PGS.TS. Hoàng Xuân Dậu**	Uỷ viên	

Các nội dung thực hiện:

1. Chủ tịch Hội đồng điều khiển buổi họp. Công bố quyết định của Giám đốc Học viện Công nghệ Bưu chính Viễn thông về việc thành lập Hội đồng chấm đề án tốt nghiệp thạc sĩ.
2. Người hướng dẫn khoa học hoặc thư ký đọc lý lịch khoa học và các điều kiện bảo vệ đề án tốt nghiệp của học viên. (có bản lý lịch khoa học và kết quả các môn học cao học của học viên kèm theo).
3. Học viên trình bày tóm tắt đề án tốt nghiệp.
4. Phản biện 1 đọc nhận xét (có văn bản kèm theo)
5. Phản biện 2 đọc nhận xét (có văn bản kèm theo)
6. Các câu hỏi của thành viên Hội đồng:

a) Cơ chế quét lỗ hổng như thế nào
b) Mối tương tác giữa Kyverno
c) Triển khai cho tổ chức có đặc trưng gì
d) Wazuh kiểm soát https như thế nào

7. Trả lời của học viên:

+ Việc quét đạo văn cần sử dụng thực hiện
+ Đề cương đề án và sẽ hoàn chỉnh thêm để làm rõ
+ Tỷ lệ trùng lặp cần cao hơn về bảo mật
+ Tiếp thu, bổ sung và kiểm soát lại https

8. Thư ký đọc nhận xét về quá trình thực hiện đề án tốt nghiệp của học viên *(có văn bản kèm theo).*

9. Hội đồng họp riêng:

- Bầu Ban kiểm phiếu:
  1. Trưởng Ban kiểm phiếu: Đào Ngọc Phong
  2. Uỷ viên Ban kiểm phiếu: Bùi Hải Phong
  3. Uỷ viên Ban kiểm phiếu: Trần Nguyên Ngọc
- Hội đồng chấm đề án tốt nghiệp bằng bỏ phiếu kín.
- Ban kiểm phiếu làm việc:
- Trưởng Ban kiểm phiếu báo cáo kết quả kiểm phiếu *(có Biên bản họp Ban kiểm phiếu kèm theo)*
- **Điểm trung bình của đề án tốt nghiệp:** 8.0

Kết luận:

1. Các nội dung cần chỉnh sửa, hoàn thiện sau bảo vệ đề án tốt nghiệp:

+ Tiếp thu đầy đủ ý kiến của hội đồng và phản biện
+ Rà soát lại việc dịch các thuật ngữ kỹ thuật
+ chỉnh sửa lại định dạng, trình bày

2. Đề nghị Học viện công nhận (hoặc không) và cấp bằng (hoặc không) thạc sĩ cho học viên:
Công nhận và cấp bằng

3. Đề án tốt nghiệp có thể phát triển thành đề tài nghiên cứu cho
NCS.....Không.....

Buổi làm việc kết thúc vào.....9h55..... cùng ngày.

Chủ tịch	**Thư ký**
PGS.TSKH. Hoàng Đăng Hải	TS. Đào Ngọc Phong

CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM

Độc lập - Tự do - Hạnh phúc

NHẬN XÉT CỦA CÁN BỘ PHẢN BIỆN

ĐỀ ÁN TỐT NGHIỆP

Họ và tên cán bộ chấm phản biện: Trần Nguyên Ngọc

Học hàm, học vị: Phó Giáo sư, Tiến sĩ

Đơn vị công tác: Học viện Kỹ thuật quân sự

Công tác chuyên môn: Quản lý, nghiên cứu và giảng dạy

NHẬN XÉT ĐỀ ÁN THẠC SĨ

Đề tài: Nghiên cứu giải pháp bảo đảm an toàn cho hệ thống Private Docker Registry của tổ chức

Học viên thực hiện: Nguyễn Huy Tâm

Đề án được kết cấu gồm 4 chương: Tổng quan về Docker và Private Docker Registry; Xây dựng hệ thống Private Docker Registry; Triển khai giải pháp đảm bảo an toàn cho hệ thống Private Docker Registry khi sử dụng Harbor; Đánh giá hiệu suất và khả năng mở rộng của giải pháp đã đề xuất ở chương 3.

Đề án được trình bày trong 73 trang tham khảo 33 nguồn tài liệu khác nhau.

Nội dung nghiên cứu của đề án là vấn đề có tính ứng dụng cao khi tập trung vào bài toán bảo mật cho hệ thống Docker trong bối cảnh các image Docker nếu không kịp thời cập nhật sẽ dễ dàng bị tấn công khai thác.

Đề án có bố cục tương đối hợp lý, tuy nhiên phần trình bày cần rà soát các thuật ngữ ví dụ "Kéo/đẩy" "Brute-force" dịch là "tấn công bằng vũ lực" là chưa hợp lý, đặc biệt đến cách trích dẫn các tài liệu tham khảo. Các nội dung

** Lưu ý: Bản nhận xét phải đánh máy, không viết tay.*

liên quan đến câu lệnh cài đặt như ở trang 31,32 có thể nghiên cứu đưa vào phần phụ lục, trong nội dung đề án chỉ cần mô tả kiến trúc tổng thể của hệ thống và các mối quan hệ giữa các thành phần.

Một số phần liên quan đến kiến thức cơ bản như khi mã hóa SSL thì không quan sát được bằng Wireshak có thể không cần đưa vào đề án thạc sỹ. Phần đánh giá hiệu suất đề án chủ yếu tham khảo đến nguồn [33] vì vậy chưa làm rõ được đóng góp của tác giả.

Đề nghị học viên làm rõ thêm các vấn đề sau:

- Đặc điểm của tổ chức có gì khác biệt so với các thực thể khác khi triển khai áp dụng Private Docker Registry?

- Cơ chế để công cụ Wazuh phát hiện ra những người dùng đăng nhập vào hệ thống mà không sử dụng https là gì?

Mặc dù còn tồn tại một số điểm cần trao đổi như trên, tuy nhiên trong phạm vi nghiên cứu học viên Nguyễn Huy Tâm đã hoàn thành nhiệm vụ được giao, tôi đồng ý để học viên bảo vệ trước hội đồng./.

Hà Nội, Ngày 9 tháng 7 năm 2025.

NGƯỜI CHẤM PHẢN BIỆN

Trần Nguyên Ngọc

** Lưu ý: Bản nhận xét phải đánh máy, không viết tay.*

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM

Độc lập – Tự do – Hạnh phúc

BẢN NHẬN XÉT ĐỀ ÁN TỐT NGHIỆP THẠC SĨ

(Dùng cho người phản biện)

Tên đề tài đề án tốt nghiệp: Nghiên cứu giải pháp đảm bảo an toàn cho hệ thống Private Docker Registry của tổ chức

Chuyên ngành: Hệ thống thông tin

Mã chuyên ngành: 8480104

Họ và tên học viên: Nguyễn Huy Tâm

Họ và tên người nhận xét: Bùi Hải Phong

Học hàm, học vị: Tiến sĩ

Chuyên ngành: Khoa học máy tính

Cơ quan công tác: Trường đại học Kiến Trúc Hà Nội

Số điện thoại: 0915594033 E-mail: phongbh@hau.edu.vn

NỘI DUNG NHẬN XÉT

I/ Cơ sở khoa học và thực tiễn, tính cấp thiết của đề tài:

Docker là một nền tảng hỗ trợ người dùng triển khai các gói phần mềm nhanh chóng, hiệu quả thay vì triển khai ứng dụng lên các máy chủ phức tạp. Đề tài có cơ sở khoa học công nghệ, phân tích được những lợi ích của Docker, các rủi ro của việc triển khai Docker không an toàn. Qua đó, đề tài triển khai các giải pháp đảm bảo an toàn cho hệ thống Private Docker Registry. Đề tài có tính thực tiễn cao.

II/ Nội dung của đề án tốt nghiệp, các kết quả đã đạt được:

Đề tài gồm 75 trang, 4 chương, có đầy đủ danh mục bảng, hình vẽ, bố cục đề tài hợp lý. Đề tài trình bày cơ sở lý thuyết về Docker, Private Docker Registry. Đề tài xây dựng Private Docker Registry, triển khai giải pháp đảm bảo an toàn cho hệ thống sử dụng Harbor.

Đề tài đã đánh giá hiệu năng của Private Docker Registry sử dụng Harbor. Tuy vậy, đề tài nên phân tích rõ hơn các kết quả đạt được trong chương 4.

III/ Những vấn đề cần giải thích thêm:

Đề tài xem xét, làm rõ các nội dung sau:
- Làm rõ cơ chế quét lỗ hổng bảo mật của Harbor?
- Mô tả chi tiết môi trường đánh giá hiệu suất của Private Docker Registry khi sử dụng Harbor trong chương 4?

IV/ Kết luận:

Đồng ý cho phép học viên bảo vệ đề án tốt nghiệp.

Đồng ý (hoặc không đồng ý) cho phép học viên bảo vệ đề án tốt nghiệp.

Ngày.....tháng....7....năm 2025

NGƯỜI NHẬN XÉT

Bùi Hải Thắng