soát truy cập là điều cần thiết. Tuy nhiên, danh sách này không thể được coi là bất biến và cả danh sách đối tượng truy cập cũng như các chính sách kiểm soát truy cập đều cần phải được cập nhật thường xuyên.

Vì thế, sự linh hoạt trong việc triển khai các phiên bản mới của microservice cùng với các chính sách bảo mật mới trở thành điểm mấu chốt. Thay vì cố gắng cập nhật trực tiếp các container đang chạy, ta có phương pháp tốt hơn là tạo ra các container mới chứa mã nguồn và chính sách bảo mật mới nhất. Mặc dù điều này có thể gây ra một số sự không nhất quán giữa các phiên bản container, nhưng lại giúp đảm bảo tính nhất quán và bảo mật hơn cho hệ thống.

Bản chất phân tán của microservices làm cho việc chia sẻ ngữ cảnh người dùng khó khăn hơn.

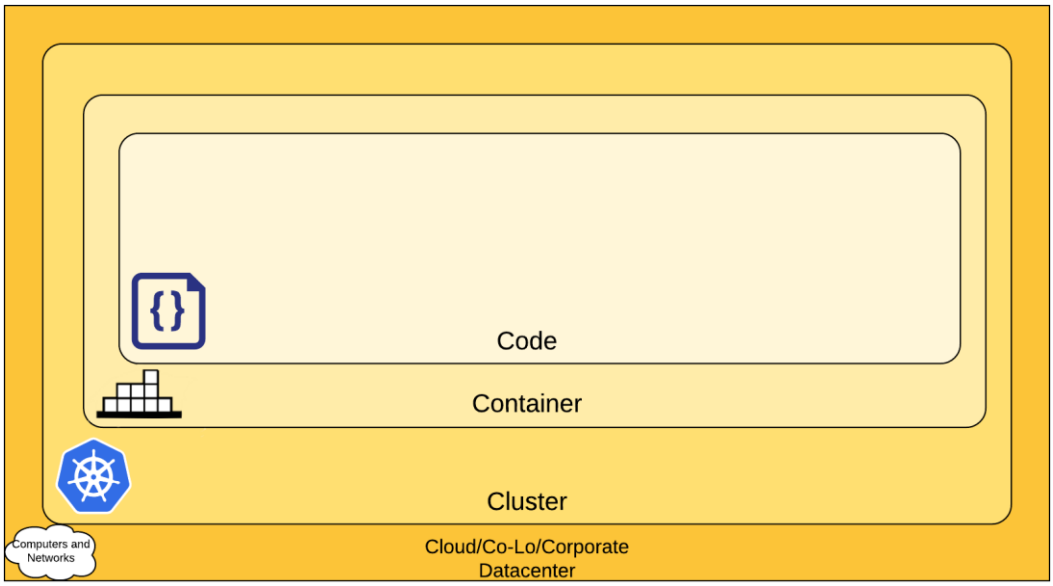
Trong kiến trúc monolithic, tất cả thông tin, ngữ cảnh và dữ liệu của người dùng (User-context) được chia sẻ liền mạch giữa các thành phần bên trong. Tuy nhiên, trong môi trường với nhiều dịch vụ hoạt động như hệ thống microservices, việc chia sẻ thông tin người dùng yêu cầu chúng ta phải đưa ra một phương thức rõ ràng và bảo mật để truyền dữ liệu người dùng từ microservice này sang microservice khác. Thách thức ở đây không chỉ đơn thuần là việc truyền dữ liệu mà còn là việc xây dựng sự tin cậy giữa các microservices, để mỗi "microservice nhận" có thể chấp nhận và xử lý ngữ cảnh của người dùng được gửi từ phần còn lại của hệ thống.

Vấn đề quan trọng khác là đảm bảo rằng thông tin người dùng được truyền giữa các microservices không bị thay đổi một cách trái phép. Điều này đòi hỏi việc xây dựng các biện pháp bảo mật như sử dụng JSON Web Tokens (JWT) để ký và mã hóa thông tin người dùng. JWT là một giải pháp phổ biến trong việc xác minh và bảo vệ dữ liệu người dùng trong môi trường phân tán, giúp đảm bảo tính toàn vẹn và đáng tin cậy của thông tin được chuyển tiếp giữa các microservices.

1.3.2. Các vấn đề an toàn của Kubernetes

Ở phần 1.2 chúng ta đã biết được Kubernetes là gì, tại sao nó được tạo ra, cũng như các thành phần cơ bản của nó. Tiếp đến ta sẽ phân tích tổng quan về các vấn đề an toàn mà hệ thống kubernetes sẽ gặp phải.

Để có cái nhìn tổng thể hơn về những vấn đề an toàn bảo mật của k8s ta không thể không nhắc đến mô hình Security 4C của các hệ thống Cloud Native. Mô hình 4C này chia hệ thống thành 4 lớp khác nhau đó chính là : Cloud, Clusters, Containers, và Code. Cách tiếp cận này sẽ giúp ta tăng cường bảo vệ theo chiều sâu (Defend in Depth) và cũng được nhiều người coi là phương thức chuẩn nhất, hiệu quả nhất để có thể bảo mật cho hệ thống phần mềm [9].



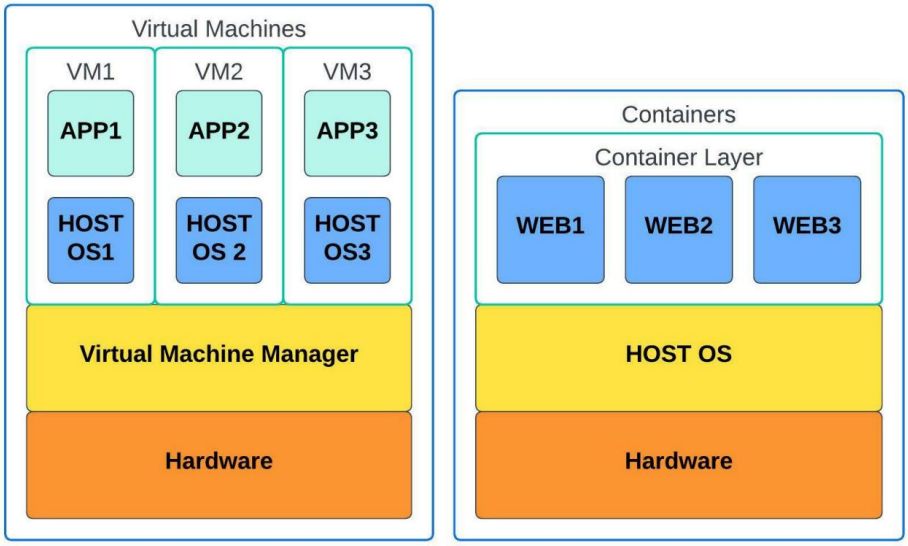
Hình 1.11: Mô hình Security Cloud, Cluster, Container và Code

Mỗi lớp của mô hình này sẽ được xây dựng dựa trên lớp ngoài kế tiếp nó. Lớp Code được hưởng lợi từ việc bảo vệ tốt các lớp trước nó như Container, Cluster, Cloud. Nhưng Code layer cũng là lớp cuối cùng trong mô hình bảo mật Cloud Native. Nếu ta chỉ giải quyết các vấn đề bảo mật ở lớp Code, mà các lớp trên không đáp ứng được các tiêu chuẩn bảo mật thì cũng không đảm bảo độ an toàn của hệ thống. (Cloud trong mô hình này là nhắc đến các nhà cung cấp dịch vụ Cloud, nhưng cũng có thể hiểu ở đây là Datacenter, là các máy chủ chạy cụm K8S chúng ta). Ta sẽ đi vào phân tích từng phần cụ thể.

An toàn bảo mật ở mức HostOS container

Container là một công nghệ ảo hóa nhẹ, cho phép chạy các ứng dụng độc lập với nhau trên cùng một máy chủ. Container có thể được triển khai nhanh chóng, dễ dàng và linh hoạt, giúp tăng hiệu quả và tiết kiệm chi phí. Tuy nhiên, container cũng

đặt ra những thách thức về bảo mật, đòi hỏi sự chú ý đến các lớp bảo mật khác nhau, từ ứng dụng, container engine, network, storage cho đến host OS.



Hình 1.12: Mô hình so sánh giữa Virtual Machines và Containers

HostOS là phần mềm tương tác với phần cứng cơ bản và là lớp bảo mật đầu tiên cần được quan tâm từ góc độ phần mềm. Nó là nền tảng để chạy container engine và các container. Nếu nó bị xâm nhập hoặc bị lợi dụng, toàn bộ hệ thống container có thể bị ảnh hưởng. HostOS có thể có bề mặt tấn công rộng hoặc hẹp tùy thuộc vào loại hệ điều hành và triết lý kỹ thuật đằng sau nó. Các hệ điều hành dành cho container thường có bề mặt tấn công nhỏ hơn và được tối ưu hóa cho container. Chúng chỉ bao gồm các thành phần thiết yếu để chạy container engine và các container, loại bỏ các thành phần không cần thiết hoặc nguy hiểm. Ví dụ một số loại HostOS như CoreOS, RancherOS, Atomic Host hay Ubuntu Core.

Các hệ điều hành này có thể bị ảnh hưởng bởi các lỗ hổng của các thành phần như kernel, thư viện hệ thống và tiện ích hệ thống. Tất cả các thành phần này đều cần được cập nhật thường xuyên, không chỉ là kernel. Các cập nhật nên được kiểm tra kỹ lưỡng trước khi áp dụng để tránh gây ra sự cố hoặc không tương thích với các container. Các cập nhật nên được áp dụng theo lịch trình định kỳ hoặc khi có thông báo khẩn cấp.

HostOS cũng có thể bị xâm nhập bởi người dùng đăng nhập vào hệ thống để quản lý các ứng dụng hoặc container. Các phiên đăng nhập nên được giám sát và

kiểm tra khi cần thiết, giới hạn quyền sudo cho một số người dùng xác định và loại bỏ người dùng root. Người dùng nên tuân theo các nguyên tắc nhỏ nhất (least privilege) và tách biệt (separation of duties) khi làm việc với các container.

Ngoài ra HostOS có thể bị lộ dữ liệu nhạy cảm nếu container được cấu hình sai để truy cập vào các tập tin hoặc thư mục của host. Container chỉ nên chạy với các tập hợp quyền truy cập tệp hệ thống tối thiểu. Các container nên được cô lập với nhau và với host OS bằng cách sử dụng các tính năng bảo mật của container engine như namespaces, cgroups, capabilities, seccomp, SELinux, AppArmor hoặc TOMOYO và cũng nên được kiểm tra định kỳ để phát hiện các hoạt động bất thường hoặc độc hại.

Tóm lại, host OS là một lớp bảo mật quan trọng cho container. Nó cần được chọn, cấu hình, cập nhật và giám sát một cách cẩn thận để bảo vệ các container khỏi các mối đe dọa. Nó cũng cần được phối hợp với các lớp bảo mật khác để tạo ra một hệ thống container an toàn và tin cậy.

Rủi ro liên quan đến Kubernetes và quản trị cụm K8S

Hệ thống cụm Kubernetes có thể gặp phải các rủi ro an ninh liên quan đến quyền truy cập quản trị, các truy cập trái phép, sự phân tách mạng của các container trong cụm [11].

Quyền truy cập quản trị là khả năng điều khiển cụm k8s, bao gồm cả các container và các tài nguyên khác. Nhiều nền tảng quản lý container được thiết kế dựa trên giả định rằng người dùng sẽ đóng vai trò là quản trị viên hệ thống. Điều này khá bình thường trong các hệ thống phức tạp, nơi mà quyền hạn được mở rộng để rất nhiều người có thể tham gia quản lý và triển khai ứng dụng. Tuy nhiên cụm kubernetes quản lý container thường điều hành nhiều ứng dụng khác nhau, mỗi ứng dụng lại thuộc về một đội nhóm, dự án khác nhau. Do đó, để có thể tùy chỉnh và xác định quyền hạn cụ thể cho từng đội nhóm, từng dự án ta cũng cần sử dụng các biện pháp cần thiết sau đây:

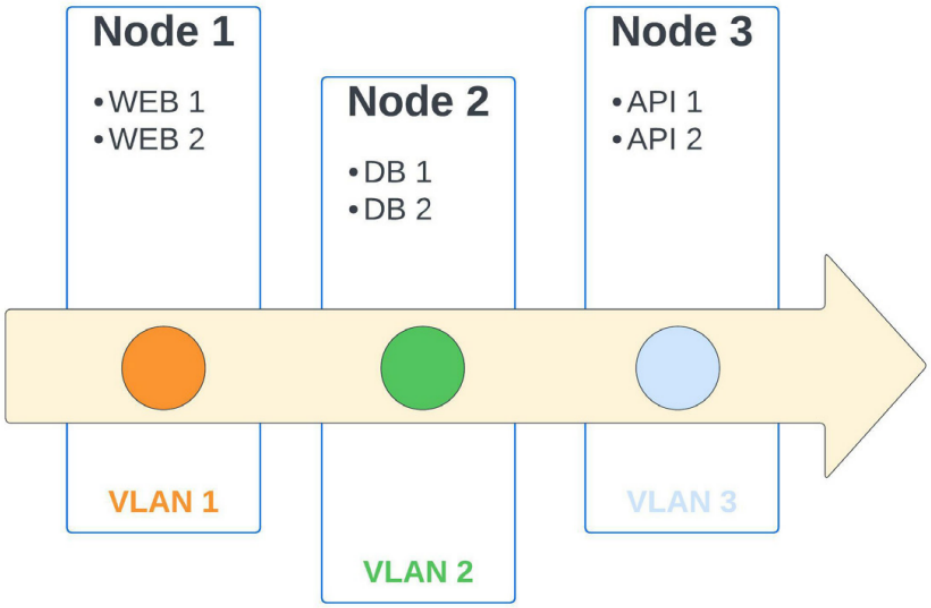
- Mô hình quyền hạn tối thiểu (Least Privileged Model)

- Tối thiểu hóa việc phân phối token truy cập có quyền đặc biệt (privileged tokens)
- Triển khai mô hình kiểm soát truy cập dựa trên vai trò (Role-Based Access Control Model)
- Cấm truy cập vào các thông tin bí mật (secrets)

Việc chỉ định một vai trò cụ thể (Role) và tạo một liên kết vai trò tương ứng (RoleBinding) cho tài khoản đảm của các team riêng biệt đảm bảo rằng chỉ có tài nguyên API hợp lệ được truy cập khi cần thiết và bởi người dùng hợp pháp.

Tiếp đến là các truy cập trái phép. Truy cập trái phép là khả năng tiếp cận với các container hoặc các tài nguyên khác mà không có sự cho phép hợp lệ. Nếu kẻ tấn công có được truy cập này, họ có thể đọc hoặc sửa đổi dữ liệu nhạy cảm, chiếm quyền kiểm soát các container hoặc khai thác các lỗ hổng bảo mật. Để ngăn chặn truy cập trái phép, các biện pháp bảo mật như mã hóa giao tiếp, xác minh danh tính và kiểm soát truy cập dựa trên thuộc tính nên được áp dụng.

Một rủi ro nữa phải kể đến chính là phân tách mạng. Phân tách mạng là khả năng ngăn chặn hoặc giới hạn sự giao tiếp giữa các container hoặc giữa các container và các dịch vụ bên ngoài. Luồng dữ liệu giữa các container có thể so sách như luồng dữ liệu giữa các node. Trong Kubernetes điểm khác biệt, và cũng là trở ngại lớn nhất là ta phải xem xét rằng các POD trên Node A có khả năng tương tác hoặc giao tiếp với nhiều POD trên Node B hay không. Mạng trong Kubernetes mặc định là phẳng (flat). Điều này có nghĩa là, khi không có các điều khiển bổ sung nào được thiết lập, bất kỳ container (workload) nào cũng có thể giao tiếp với các container khác mà không bị ràng buộc. Những kẻ tấn công khai thác container hay PODs đang hoạt động có thể tận dụng hành vi mặc định này để thăm dò mạng nội bộ, di chuyển đến các container đang hoạt động khác, hoặc gọi các API riêng.

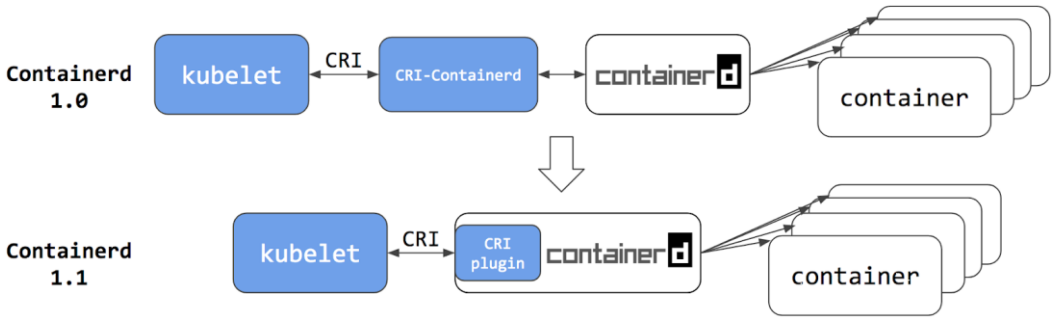


Hình 1.13: Sự phân tách mạng giữa các thành phần

Để bảo vệ phân tách mạng, các biện pháp bảo mật như tạo ra các mạng riêng cho từng container hoặc nhóm container, áp dụng Network Policy và sử dụng các dịch vụ proxy cũng nên được áp dụng.

Rủi ro bảo mật liên quan đến Container

Như ta đã biết Container là thành phần nhỏ nhất trong cấu trúc của một cụm K8S, và nó ảnh hưởng trực tiếp đến tầng ứng dụng. Để các container này có thể hoạt động được Kubernetes phải sử dụng Container runtime. Container runtime là một daemon service trong Linux để quản lý, tạo, khởi chạy, dừng và hủy các container đồng thời nó cũng quản lý lưu trữ (storage) và mạng cho container.



Hình 1.14: Cấu trúc của Container Runtime Interface trong hệ thống Kubernetes

Như đã thấy ở ảnh trên runtime là điểm kết nối giữa container và hệ điều hành. Một container bị xâm phạm có thể tạo điều kiện cho kẻ tấn công, tác nhân đe dọa thực hiện các hành vi di chuyển, mở rộng phạm vi quyền hạn từ bên trong (lateral movements), xâm phạm các container khác thậm chí là hệ điều hành bên ngoài. Người ta thường gọi đó là kiểu tấn công thoát khỏi container (container escape). Có hai nguyên nhân chính khiến kiểu tấn công này có thể xảy ra:

- Cấu hình không an toàn
- Các lỗ hổng phần mềm trong runtime

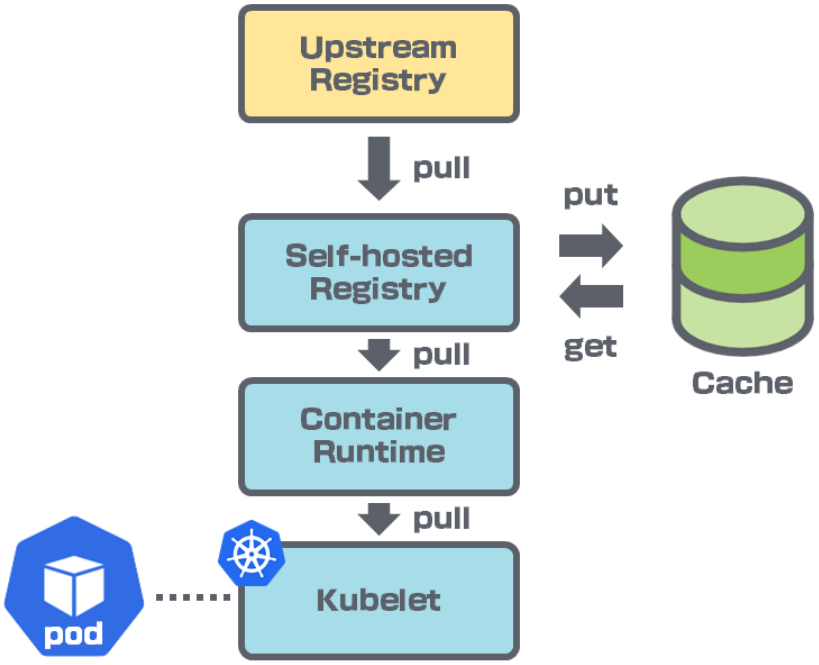
Một số ví dụ về lỗ hổng trong container runtime có thể kể đến gần đây như CVE-2022-29162 của Runc hay lỗ hổng CVE-2022-23648 của Containerd.

Trong container ta cũng cần quan tâm đến luồng lưu lượng mạng của container (Network traffic). Nó là luồng dữ liệu giữa các container và giữa các container với mạng bên ngoài. Nếu network traffic không được mã hóa hoặc kiểm soát, nó có thể bị nghe trộm, thay đổi hoặc chặn đứng bởi kẻ xâm nhập.

Tầng logic ứng dụng chạy trong container cũng nên được quan tâm đến. Ở tầng này, logic code, ứng dụng của chúng ta có thể chứa các lỗ hổng an ninh hoặc mã độc, dẫn đến việc bị lộ, mất, thất thoát dữ liệu, hoặc bị chiếm quyền điều khiển bởi kẻ xâm nhập.

Các vấn đề an toàn bảo mật của Image ứng dụng

Tiếp theo để các container có thể chạy được Image của các container là điều không thể thiếu. Trong kubernetes việc khởi tạo một POD mới luôn phải gắn với quá trình tải xuống (pull) các image.

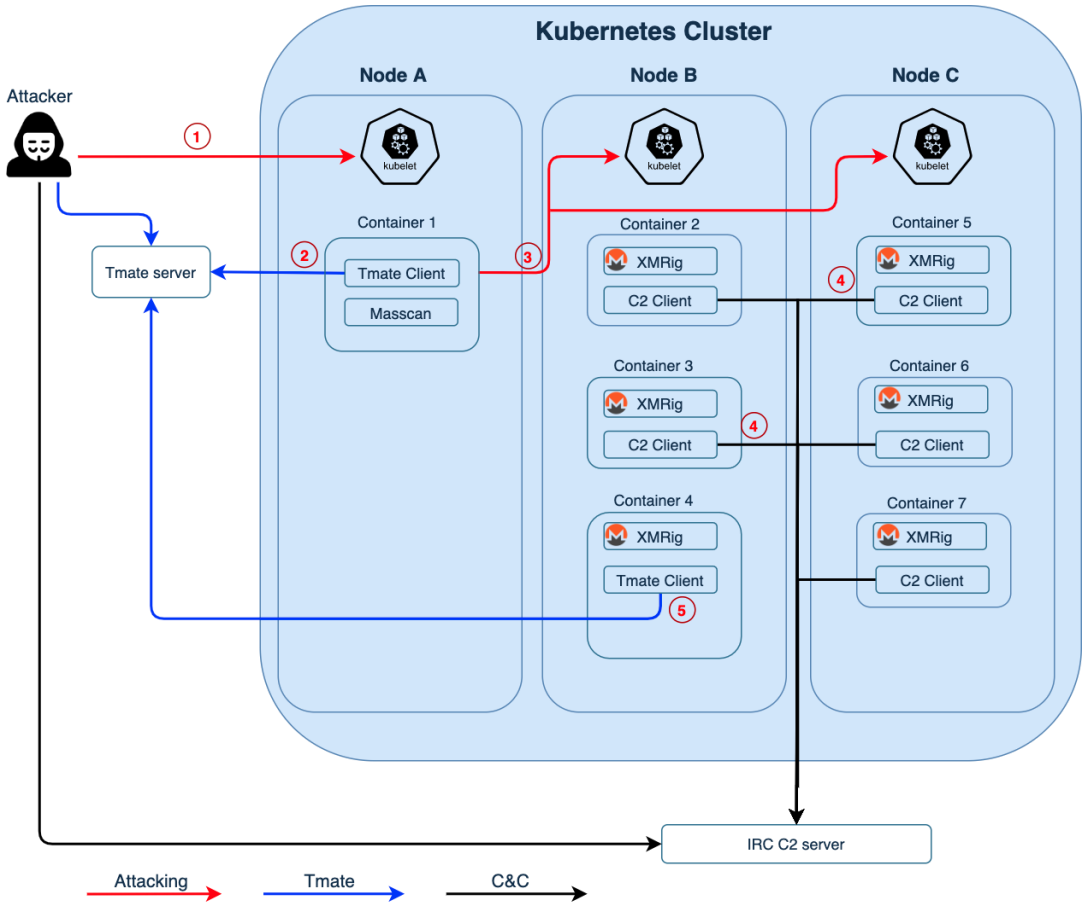


Hình 1.15: Cách thức Pull image về để khởi tạo POD trong K8S

Và vì vậy vấn đề an toàn của các Image này cũng là điều chắc chắn phải nhắc đến. Đầu tiên là những lỗ hổng an ninh có thể tồn tại trong các Image container do sử dụng các gói phần mềm cũ, không được cập nhật hoặc chứa mã độc. Những lỗ hổng này có thể cho phép kẻ xâm nhập khai thác và chiếm quyền điều khiển container hoặc truy cập dữ liệu bí mật.

Trong quá trình sử dụng và khởi chạy các POD, sai sót trong việc thiết lập các tham số hoặc tùy chọn cho các Image container, có thể gây ra các vấn đề về an ninh, ví dụ như thiết lập các policy cho phép chạy Image container bằng quyền root.

Các image không được tin tưởng, không được xác minh nguồn gốc, cũng như các image có chứa mã độc hoàn toàn có thể được tải về qua các kho lưu trữ image công khai hoặc các bên thứ ba không uy tín. Vào ngày 22 tháng 4 năm 2022 Vbulletin cũng đã công bố một tài liệu báo cáo về cuộc tấn công của nhóm hacker TeamNTN vào các hệ thống K8S, tiến hành thay thế các POD bằng các mã độc đào tiền ảo.



Hình 1.16: Cuộc tấn công và cụm K8S để cài đặt mã độc đào coin

Bảo mật Registry (Kho lưu trữ images container)

Và cuối cùng là một phần quan trọng không kém, tuy không nằm trong hệ thống K8S nhưng nhưng nó ảnh hưởng trực tiếp đến việc lưu trữ các Image container được pull và sử dụng trong hệ thống K8S, đó chính là Registry. Có hai loại kho chứa container chính là:

- Kho chứa công khai (public registries): Được sử dụng bởi các cá nhân hoặc team muốn nhanh chóng và thuận tiện trong việc sử dụng. Tuy nhiên, điều này có thể mang lại nhiều vấn đề bảo mật phức tạp hơn như các bản vá lỗi và kiểm soát truy cập.
- Kho chứa riêng tư (private registries): Cung cấp các cơ chế bảo mật cho việc lưu trữ image container do doanh nghiệp sử dụng, có thể được lưu trữ từ xa hoặc tại chỗ.

Các registry chỉ nên cho phép kết nối qua các kênh an toàn để thực hiện các thao tác Push (đẩy Image lên Registry) và Pull (kéo Image về server) giữa các điểm

cuối tin cậy thông qua các cơ chế mã hóa trong quá trình truyền. Các kho chứa công khai thường đã có các tính năng này, nhưng với các kho chứa riêng tư, ta cần phải tự thiết lập các biện pháp bảo mật.

Một điều cần lưu ý khi sử dụng các kho chứa riêng tư cho công ty đó chính là việc xác thực và ủy quyền. Công ty hoặc tổ chức cần phải xác định rõ ràng xem ai có quyền truy cập vào registry và ai thì không. Việc các team có thể push đè image container của nhau là điều hoàn toàn có thể xảy ra, và điều đó ảnh hưởng nghiêm trọng đến vấn đề an toàn của hệ thống kho chứa nói riêng và cho cụm K8S nói chung. Việc sử dụng OAuth hay RBAC là hết sức quan trọng trong việc đảm bảo an toàn cho kho lưu trữ image của container.

1.4. Kết luận chương 1

Chương 1 đã cung cấp một cái nhìn tổng quan về hai khía cạnh quan trọng trong việc triển khai và quản lý ứng dụng phân tán: kiến trúc Microservice và kiến trúc Kubernetes. Với việc giới thiệu về kiến trúc Microservice, chúng ta đã tìm hiểu về khái niệm, kiến trúc cơ bản, cũng như lợi ích và hạn chế của Microservice. Điều này đã giúp chúng ta nắm rõ về cách mà việc phân tách ứng dụng thành các dịch vụ nhỏ và độc lập có thể mang lại hiệu quả trong phát triển và quản lý.

Tiếp theo, việc giới thiệu về kiến trúc Kubernetes đã cung cấp cái nhìn sâu hơn về khái niệm cơ bản của Kubernetes, cách mà kiến trúc của nó được cấu thành và các thành phần quan trọng trong hệ thống. Chương này cũng giúp chúng ta hiểu rõ hơn về cách mà Kubernetes đảm bảo việc triển khai, quản lý và mở rộng các ứng dụng phân tán.

Ngoài ra, phần 1.3 của chương đã tập trung vào các vấn đề an toàn của cả Microservice và Kubernetes. Chúng ta đã tìm hiểu về những rủi ro và thách thức mà các kiến trúc này mang lại trong việc bảo mật và đảm bảo tính toàn vẹn của hệ thống.