TẠP CHÍ KHOA HỌC & CÔNG NGHỆ ĐẠI HỌC DUY TÂN
DTU Journal of Science and Technology 3(58) (2023) 12-22
TRƯỜNG ĐẠI HỌC DUY TÂN
DUYTAN UNIVERSITY

Xây dựng hệ thống tích hợp liên tục nội bộ sử dụng công cụ nguồn mở Jenkins và Gitlab

Building an internal continuous integration system using two open source tools JENKINS and GITLAB

Nguyễn Kim Tuấn, Nguyễn Trí Tâm*, Trần Hữu Minh Đăng, Hoàng Phi Cường
Nguyen Kim Tuan, Nguyen Tri Tam*, Tran Huu Minh Dang, Hoang Phi Cuong

Trường Khoa học máy tính, Đại học Duy Tân, Đà Nẵng, Việt Nam
School of Computer Science, Duy Tan University, 550000, Da Nang, Viet Nam

(Ngày nhận bài: 05/4/2023, ngày phản biện xong: 22/4/2023, ngày chấp nhận đăng: 13/5/2023)

Tóm tắt

Trong bài báo này, chúng tôi đề xuất và triển khai một hệ thống "tích hợp liên tục" nội bộ, dựa trên hai công cụ mã nguồn mở Jenkins và Gitlab, có tính đến yếu tố đảm bảo an toàn cho các máy chủ trong hệ thống. Trong hệ thống đề xuất, chúng tôi sử dụng kết hợp chức năng của firewall và chức năng của reverse proxy để bảo vệ chính Jenkins server và giảm thiểu rủi ro cho server này trước các cuộc tấn công vào lỗ hổng CVE-2021-44228, có thể tồn tại trong các plugin của Jenkins. Hệ thống này có tính thực tế cao, và nó có thể được áp dụng để "ngay lập tức" bảo vệ các server dịch vụ khi mà lỗ hổng trong nó đã được phát hiện nhưng bản vá lỗi tương ứng chưa được tìm thấy hoặc điều kiện cập nhật bản vá chưa cho phép.

Từ khóa: Tích hợp liên tục; chuyển giao liên tục; tường lửa bảo mật; phần mềm nguồn mở; CVE-2021-44228.

Abstract

In this paper, we propose and implement an internal "continuous integration" system, based on two open-source tools Jenkins and Gitlab, taking into account the safety factor for servers in the system. In the proposed system, we use a combination of firewall function and reverse proxy function to protect Jenkins server itself and reduce the risk of this server against attacks on vulnerability CVE-2021-44228, may exist in plugins of Jenkins. This system is highly practical, and it can be applied to "immediately" protect service servers when a vulnerability in it has been discovered but the corresponding patch has not been found or the condition to update the patch is not allowed yet.

Keywords: Continuous Integration; Continuous Delivery; CVE-2021-44228; Firewalls; Jenkin; Gitlab;

1. Giới thiệu

Tích hợp liên tục (CI: Continuous Integration) và Phân phối liên tục (CD: Continuous Delivery) là hai quy trình cốt lõi trong phương pháp phát triển phần mềm DevOps, một phương pháp phát triển phần mềm hiện đại, có nhiều ưu việt so với phương pháp phát triển phần mềm truyền thống và được triển khai ở hầu hết các doanh nghiệp, các nhóm phát triển phần mềm hiện nay. Quy trình

Tác giả liên hệ: Nguyễn Trí Tâm, Khoa Kỹ thuật Mạng máy tính & Truyền thông, Trường Khoa học Máy tính, Đại học Duy Tân, Đà Nẵng, Việt Nam
Email: nguyentritam@dtu.edu.vn

CI và quy trình CD được xem là hai mặt của một vấn đề, nhưng chúng được sử dụng kết hợp với nhau để làm cho quá trình phát triển một sản phẩm phần mềm, từ lúc một phiên bản mới của mã nguồn được đưa ra (commit) đến khi sản phẩm mới nhất được triển khai (deploy), trở nên đơn giản, nhanh chóng và hiệu quả hơn. Thông tin chi tiết, đầy đủ, liên quan đến DevOps, CI và CD, không thuộc phạm vi của nghiên cứu này, nên chúng tôi không trình bày chi tiết ở đây, nó có thể được tìm thấy ở [1-3].

Hạ tầng mạng máy tính, để triển khai các dịch vụ liên quan, là cần thiết để một doanh nghiệp, hoặc một nhóm, phát triển phần mềm có thể triển khai các quy trình CI, CD một cách tự động vào chu kỳ phát triển phần mềm theo hướng tiếp cận DevOps của họ. Các máy chủ thực hiện chức năng CI server và CD server đóng vai trò quan trọng trong hạ tầng mạng này, vì thế, chúng cần được bảo vệ để có khả năng chống lại các cuộc tấn công vào chính nó, hoặc vào dịch vụ đang được triển khai trên nó. Hạ tầng mạng này có thể triển khai trên không gian Internet (dựa trên các dịch vụ Cloud) hoặc trong mạng nội bộ (Intranet/Extranet), tùy thuộc vào yêu cầu, điều kiện và quy mô của mỗi doanh nghiệp phát triển phần mềm. Hạ tầng mạng CI/CD trong mạng Internet hay trong mạng nội bộ đều có những ưu điểm, nhược điểm riêng. Trong bài báo này, chúng tôi đề xuất một hạ tầng mạng chỉ đảm bảo cho sự tự động của quy trình CI trong mạng nội bộ, nó được gọi là "Hệ thống Tích hợp liên tục nội bộ", vì chúng tôi quan tâm đến tốc độ, tính sẵn sàng và sự chủ động trong công tác triển khai bảo mật của nó.

Hai thành phần chính của mọi hệ thống tích hợp liên tục là server thực hiện chức năng tích hợp liên tục (CI server) và server đóng vai trò lưu trữ và quản lý mã nguồn (Repository server). Nếu các server này bị "đánh sập" bởi kẻ tấn công thì hệ thống này không những phải dừng hoạt động mà dữ liệu lưu trữ trong nó có thể bị khai thác, bị đánh cắp một cách có chủ đích. Mối nguy này là tiềm ẩn, vì các server này cho dù được đặt trên không gian Internet hoặc bên trong mạng nội bộ đều phải có tính chia sẻ, tính sẵn sàng và tính đa kết nối đồng thời.

Ngoài ra, kẻ tấn công không chỉ tấn công vào chính các CI server hay Repository server mà còn tìm cách khai thác các lỗ hổng bảo mật có thể tồn tại trong các phần mềm được sử dụng để xây dựng các server này. Như vậy, đảm bảo an toàn cho CI server và Repository server là cần thiết, có nhiều kỹ thuật và công cụ khác nhau có thể được sử dụng cho nhiệm vụ này, chúng tôi chọn cách sử dụng firewall cho hệ thống được đề xuất trong bài báo này.

Trong hệ thống của chúng tôi, firewall không những thực hiện nhiệm vụ bảo vệ chính CI server (sử dụng Jenkins) và Repository server (sử dụng Gitlab) mà còn có thể ngăn chặn các cuộc tấn công khai thác lỗ hổng bảo mật CVE-2021-44228 tồn tại trong một số plugin của công cụ mã nguồn mở Jenkins. Trong trường hợp này, chúng tôi muốn giới thiệu một cách khác để "bịt" một lỗ hổng tồn tại trong một server dịch vụ, ở đây là Jenkins, mà chưa cần, hoặc không cần, sử dụng bản vá tương ứng. Tất nhiên, sử dụng bản vá để "bịt" lỗ hổng là giải pháp thường được lựa chọn nhất, nhưng trong điều kiện bản vá chưa sẵn sàng hoặc thời điểm thực hiện việc "vá" lỗ hổng chưa cho phép thì việc sử dụng firewall để "bịt" tạm thời lỗ hổng được phát hiện là một đề xuất nên được xem xét để triển khai.

2. Một số kiến thức liên quan

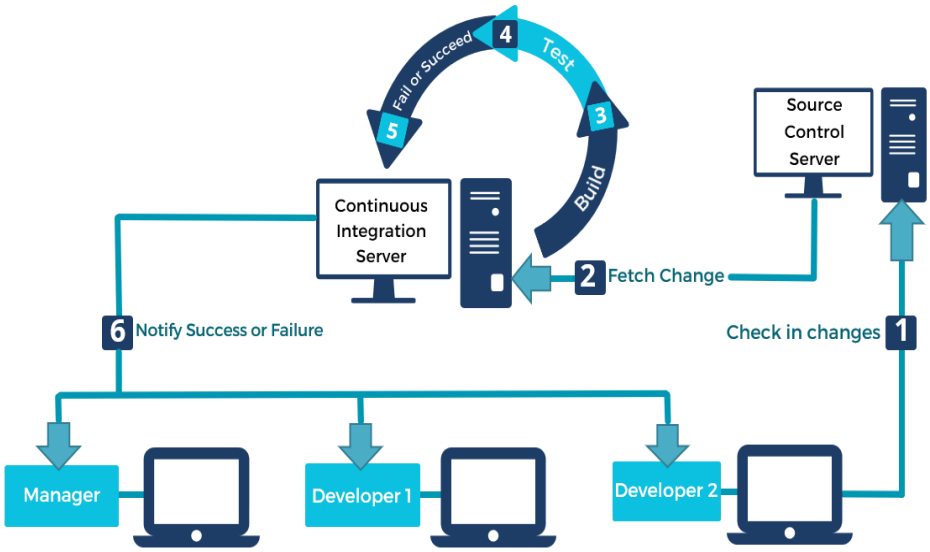
2.1. Hệ thống tích hợp liên tục dựa trên Jenkins và Gitlab

Trong phần này, chúng tôi chỉ trình bày những thông tin cơ bản nhất liên quan đến một hệ thống tích hợp liên tục và những chức năng chính của hai công cụ mã nguồn mở Jenkins và Gitlab. Lý do chọn sử dụng Jenkins và Gitlab để xây dựng hệ thống đề xuất cũng được đề cập ở đây. Thông tin chi tiết về những chủ đề này đã được đề cập ở [4-7].

2.1.1. Về hệ thống tích hợp liên tục

Hình 2.1 dưới đây cho thấy các thành phần chính, và nguyên lý hoạt động, của một một hệ thống CI nội bộ. Trong đó, Continous Integration Server (CI server) đóng vai trò điều khiển quy trình tích hợp liên tục và làm cho quy trình này hoạt động một cách tự động. i) Source Control Server (Repository server) chịu trách nhiệm lưu trữ và quản lý các tập tin mã nguồn, mà nó nhận được từ Manager và/hoặc từ các Developer. Server này thực hiện một giải pháp tin cậy để tập trung và duy trì những sự thay đổi mã nguồn được tạo ra bởi các Developer (1: Check-in Changes). ii) CI server liên tục, hoặc theo định kỳ, phát hiện có sự thay đổi code trên Repository server, thực hiện việc nhận phiên bản code mới nhất từ Repository server (2: Fetch changes) để thực hiện những công việc được giao của nó (3: Build; 4: Test; 5: Fail or Succeed), và rồi gửi kết quả (6: Notify success or failure) về lại cho các thành viên của dự án, Manager và Developer.

Như vậy, duy trì sự kết nối ở mức cao, cả về tốc độ và sự sẵn sàng, giữa CI server, Repository server, Manager computers, Developer computers là yêu cầu cần đặt ra với hệ thống này. Hệ thống đề xuất của chúng tôi đặc biệt quan tâm đến tính sẵn sàng của CI server và Repository server.



Hình 2.1: Các thành phần và nguyên lý hoạt động của một hệ thống CI nội bộ

2.1.2. Về lý do chọn Jenkin để xây dựng CI server

Hiện trên thị trường đã có sẵn nhiều công cụ hỗ trợ xây dựng CI server cho hệ thống CI/CD, như Buddy, TeamCity, Jenkins, GoCD, CodeShip, v.v... Theo tìm hiểu của chúng tôi, công cụ mã nguồn mở Jenkins có nhiều ưu điểm hơn và được sử dụng rộng rãi hơn, so với các sản phẩm cùng loại, bởi: Jenkins được phát triển từ Java nên tính tương thích đa nền tảng của nó là rất cao; số lượng plugin của Jenkins lớn và đa dạng về chức năng; người dùng Jenkins nhận được sự hỗ trợ rất lớn từ cộng đồng những nhà phát triển; người quản trị Jenkins có thể thực hiện các thao tác cấu hình trên nó thông qua cả giao diện giả GUI và câu lệnh điều khiển; v.v... Đây là lý do mà chúng tôi sử dụng Jenkins để xây dựng CI server cho hệ thống đề xuất.

Vấn đề đảm bảo an toàn cho sự hoạt động của Jenkins server cũng được Jenkins chú trọng, nó cho phép người quản trị thực hiện các tùy chọn về bảo mật sẵn có, như Access Control; CSRF Protection; Controller Isolation; Content Security Policy; v.v... một cách dễ

dàng. Tuy nhiên, trong nghiên cứu này, chúng tôi chỉ đề cập đến phương án bảo vệ server Jenkins, và biện pháp đối phó với lỗ hổng CVE-2021-44228 của Jenkins, theo hướng tiếp cận sử dụng Firewall.

2.1.3. Về lý do chọn Gitlab để xây dựng Repository server

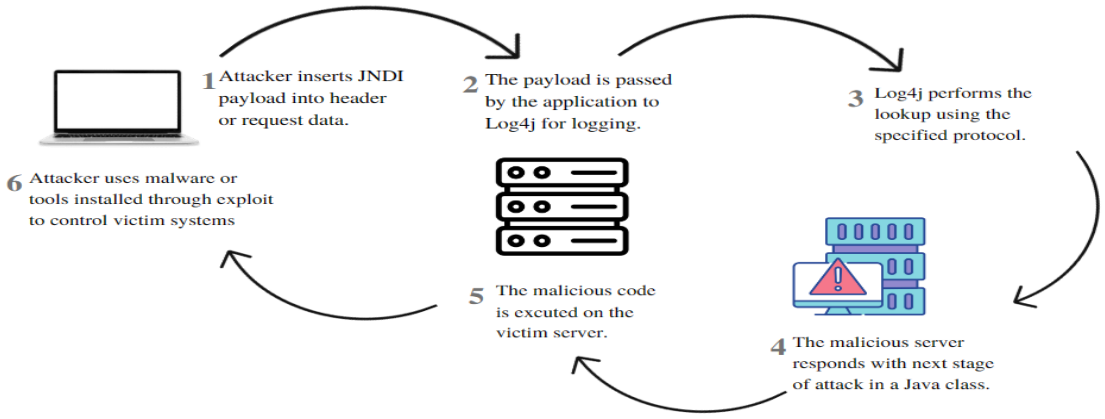
Gitlab cũng là công cụ mã nguồn mở, nó thường được sử dụng để xây dựng các server lưu trữ, quản lý mã nguồn (Source code repository server) cho hệ thống CI/CD. Chúng ta có thể sử dụng một trong các sản phẩm sẵn có sau đây để thay thế cho Gitlab, như Git, Github, Apache Subversion, Bitbucket Server, Team Foundation Server, v.v..., vì tất cả chúng đều cung cấp các sự hỗ trợ cơ bản mà một repository server cần có, như hỗ trợ cơ chế theo dõi sự thay đổi của một tập tin (source code files) được lưu trữ; hỗ trợ cơ chế cho phép nhiều thành viên (developers) truy xuất source code file một cách đồng thời; hỗ trợ cơ chế giải quyết xung đột khi đồng thời có nhiều developer cùng muốn thay đổi nội dung của một source code file; hỗ trợ một giao diện web để developer thấy được sự thay đổi giữa hai phiên bản source code, phiên bản trước đó và phiên bản vừa được commit, một cách trực quan; v.v...

Trong hệ thống đề xuất, chúng tôi sử dụng Gitlab để xây dựng Repository server vì những ưu điểm nổi trội của nó, như tính miễn phí không giới hạn, khả năng tích hợp với Git, cho phép kiểm tra bảo mật động, cho phép dự án được thực hiện theo kế hoạch lập sẵn, v.v... Đặc biệt, GitLab cho phép hạn chế quyền truy cập vào các dự án và xem trạng thái tuân thủ của từng người tham gia; Trong phạm vi của nghiên cứu này, chúng tôi không đề cập đến những vấn đề bảo mật liên quan đến Gitlab, điều này sẽ được làm rõ trong một công bố của chúng tôi trong tương lai.

2.2. Lỗ hổng bảo mật CVE-2021-44228

Lỗ hổng Log4Shell [8], được Chen Zhaojun của Alibaba phát hiện vào năm 2021 và được định danh là CVE-2021-44228, thuộc nhóm lỗ hổng bảo mật phần mềm, tồn tại trong Log4j2. Log4j2 là một thư viện mã nguồn mở dựa trên ngôn ngữ Java, nó hỗ trợ việc ghi lại (log) các massage lỗi, và những thông tin liên quan, phát sinh từ các chương trình ứng dụng Java. Log4j2 được dùng khá phổ biến, đặc biệt, được tích hợp trong máy chủ web Apache.

Lỗ hổng CVE-2021-44228 được xem là một trong những lỗ hổng nghiêm trọng nhất được phát hiện, tính đến thời điểm hiện tại, nó cho phép kẻ tấn công thực hiện một cuộc tấn công từ xa để chiếm quyền điều khiển mục tiêu dễ bị tấn công, thường là một thiết bị trên Internet, nếu thiết bị đó đang chạy một phiên bản của Log4j2, mà không cần xác thực. Để thực hiện tấn công này, kẻ tấn công chỉ cần gửi một yêu cầu mà trong đó có chứa mã khai thác (payload), thông qua một trong các giao thức LDAP, RMI, DNS, CORBA, v.v..., đến máy chủ sử dụng Log4j2, để kích hoạt lỗ hổng trong Log4j2. Khi đó, thông qua một trong các dịch vụ do kẻ tấn công kiểm soát, máy chủ sẽ tạo một yêu cầu (request) trên JNDI (Java Naming and Directory Interface) để gửi lại cho kẻ tấn công. Nhận được yêu cầu này, kẻ tấn công sẽ trả về một đường dẫn đến tệp Java class được lưu trữ từ xa, tệp này sau đó sẽ được phía máy bị tấn công chèn vào luồng xử lý và cho phép kẻ tấn công có thể thực thi mã từ xa (RCE: Remove Code Execute) tùy ý. Hình 2.2 cho thấy hoạt động của tấn công vào lỗ hổng bảo mật CVE-2021- 44228.



Hình 2.2. Hoạt động của tấn công vào lỗ hổng bao mật CVE-2021-44228

Như vậy, tác hại to lớn tiềm ẩn trong lỗ hổng bảo mật Log4Shell là đã rõ, nhưng rất không may là thư viện Log4j được sử dụng trong khá nhiều plugin của Jenkins (Log4j không tồn tại trong core của Jenkins), như Audit-log; bootstraped-multi-test-results-report; cmakebuilder; cucumber-reports; v.v... Chính vì điều này mà công tác đảm bảo an toàn cho các CI server sử dụng Jenkins trong các hệ thống CI/CD luôn được người quản trị hệ thống đặc biệt quan tâm.

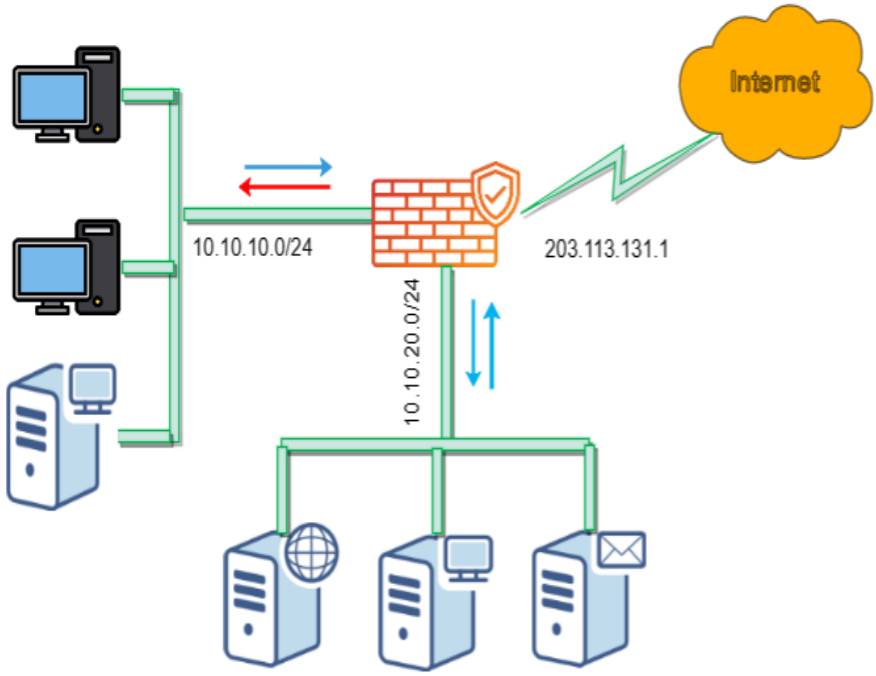
Hiện có nhiều giải pháp được đưa ra để đối phó với lỗ hổng bảo mật CVE-2021-44228 như: Cập nhật phiên bản mới nhất của thư viện Log4j (v2.17.0); vô hiệu hóa chức năng tìm kiếm trong message; xóa lớp JndiLookup khỏi classpath; sử dụng bảng rules trong firewall lọc gói để giới hạn sự kết nối đến các server, các dịch vụ có sử dụng thư viện Log4j; v.v... Vì kẻ tấn công có thể chèn mã khai thác lỗ hổng Log4j vào bất kỳ thành phần nào HTTP headers và HTTP body nên trong nghiên cứu này chúng tôi sử dụng firewall lọc gói vừa để bảo vệ chính CI server vừa giảm thiểu rủi ro do lỗ hổng Log4Shell tồn tại trong server này gây ra.

2.3. Vai trò của Firewall lọc gói trong bảo mật mạng

Thông tin chi tiết về hoạt động của firewall và firewall lọc gói được chúng tôi trình bày ở [9-10], do đó, trong phần này, chúng tôi chỉ trình bày lại những đặc trưng của loại firewall này.

Firewall lọc gói dựa vào chính sách truy cập mạng, được thiết kế dưới dạng Bảng luật lọc gói, và các thông tin liên quan chứa trong dòng traffic, thường là IP Address, Protocols và Port number, để quyết định cho phép một dòng traffic, hay một gói tin (packet), nào đó có được phép đi qua nó hay không. Bảng luật lọc gói được người quản trị an ninh mạng thiết lập và cài đặt vào firewall, nó là cơ sở để Bộ lọc gói ra quyết định cho phép gói tin đến firewall có được đi qua nó, để hướng đến đích của gói tin, hay không.

Hình 2.3 là sơ đồ minh họa mạng doanh nghiệp, trong đó có sử dụng firewall lọc gói như là một cổng kết nối mạng, mọi gói tin từ mạng Internet vào mạng bên trong và từ mạng bên trong ra Internet đều phải đi qua firewall.



Hình 2.3. Sơ đồ mạng với sự có mặt của firewall lọc gói

Chính sách truy cập mạng của mạng doanh nghiệp này là như sau: Chỉ có các gói tin từ Internet chỉ được đi vào vùng mạng 192.168.2.0/24 - vùng DMZ của mạng doanh nghiệp - không được phép đi vào vùng mạng 10.10.10.0/24, vùng mạng người dùng bên trong. Tuy nhiên, mọi dòng traffic từ vùng mạng người dùng và vùng mạng DMZ đều có thể đi ra mạng Internet.

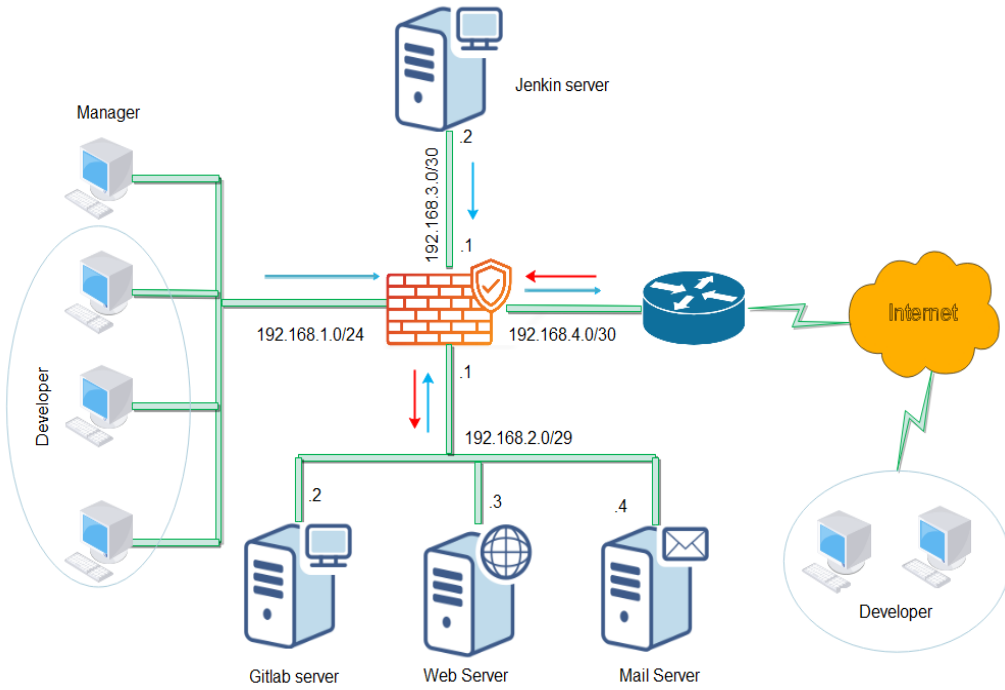
Trong nghiên cứu này, chúng tôi sử dụng công cụ Iptables kết hợp với công cụ Nginx (Reverse proxy) để xây dựng các firewall cho hệ thống đề xuất [11], nó vừa bảo vệ các server bên trong mạng, vừa có thể ngăn chặn tấn công vào lỗ hổng Log4j.

Xin nhắc lại rằng, Reverse proxy server là một server đóng vai trò cầu nối giữa các máy client và các máy server. Server này kiểm soát các request được gửi từ các client, và điều phối những request này tới một server phù hợp, để các request đó được xử lý. Khi server xử lý xong, sẽ trả về response cho Reverse proxy, và Reverse proxy có trả về response đó cho client gửi request.

3. Hệ thống tích hợp liên tục nội bộ đề xuất

3.1. Sơ đồ hệ thống

Trong phần này, chúng tôi đề xuất một hệ thống tích hợp liên tục nội bộ, có tính đến yếu tố đảm bảo an toàn cho CI server và Repository server. Hình 3.1 là sơ đồ mạng của hệ thống này.



Hình 3.1: Sơ đồ hệ thống tích hợp liên tục nội bộ đề xuất

Trong sơ đồ này: Máy Jenkins server đóng vai trò CI Server; Máy Gitlab server đóng vai trò Server Repository; Web server và Mail server là hai thành phần cơ bản của một mạng Intranet; Router được sử dụng cho việc kết nối mạng nội bộ vào Internet; Firewall được sử dụng cho việc bảo vệ các server, bảo vệ vùng mạng bên trong và thực hiện chức năng của một Reverse proxy server cho http, https và một số giao thức khác (phục vụ cho việc ngăn chặn tấn công vào lỗ hổng CVE-2021-44228).

Vùng mạng chứa Web server và Mail server thường được gọi là vùng DMZ. Thông thường, mọi kết nối từ Internet vào các server trong DMZ là được phép, do đó, Gitlab server được đặt ở DMZ là với mục đích cho phép developer kết nối và commit mã nguồn lên Server Repository ngay cả khi họ ở bên ngoài mạng nội bộ. Người dùng trên Internet, kể cả Manager và Developer, đều không được phép truy cập trực tiếp đến Jenkins server, đây là lý do mà Jenkins server được đặt ở một vùng mạng riêng, để nó được bảo vệ bởi cả Router và Firewall.

3.2. Cấu hình hệ thống

Yêu cầu cấu hình đối với hệ thống đề xuất bao gồm:

- Thiết lập địa chỉ IP cho từng máy tính và thiết bị trong mạng theo đúng lược đồ địa chỉ IP đã định (Hình 3.1). Máy tính của các Developer được chỉ định nhận địa chỉ IP động (một DHCP server bên trong mạng nội bộ sẽ đảm nhận việc phân phối địa chỉ IP động này).

- Cài đặt Jenkins, và các plugin cần thiết, lên máy tính được chọn làm Jenkins server, máy này phải được cài đặt Java trước đó, và rồi thiết lập các thông số cấu hình cho server này. Chú ý chọn port dịch vụ (service port number) phù hợp cho Jenkins (Port number mặc định của Jenkins là 8080, tuy nhiên chúng ta có thể thay đổi port này sau khi đã xong cài đặt Jenkins).

- Cài đặt Gitlab, và các plugin liên quan, lên máy tính được chọn làm Repository server, và rồi thiết lập các thông số cấu hình cho server này. Nhớ lưu lại thông tin liên quan đến tài khoản quản trị Gitlab được tạo ra trong quá trình cài đặt để đổi mật khẩu hoặc đăng nhập

Gitlab để thực hiện chức năng của người quản trị hệ thống sau này. Port number mặc định của Gitlab là 80, tuy nhiên chúng ta có thể thay đổi port này sau khi đã xong cài đặt Gitlab. Dịch vụ FirewallD trên Gitlab sẽ được tắt, vì chúng tôi sử dụng một firewall riêng, firewall IpTables kết hợp Nginx.

- Kết nối Jenkins server với Gitlab server: Trước hết phải tạo một cặp khóa xác thực, để từ Jenkin server có thể kết nối được đến Gitlab server.

- Kết nối Jenkins server với Web server: Trước hết phải tạo một cặp khóa xác thực, để có thể tạo sự kết nối từ Jenkin server đến Web server.

Với sơ đồ mạng của hệ thống đề xuất, việc cấu hình cho firewall Iptables, và thiết lập bảng luật lọc gói cho nó, là rất quan trọng. Phải thực hiện đúng bước cấu hình này thì "Chính sách truy cập mạng", hay còn gọi là "Chính sách an ninh mạng", của hệ thống mới có hiệu lực. Khi đó, Firewall sẽ dựa vào các bảng luật lọc gói để thực hiện được chức năng bảo vệ mạng, bảo vệ các server và đặc biệt là ngăn chặn các cuộc tấn công vào lỗ hổng bảo mật CVE-2021-44228 trên Jenkins server.

Bảng 3.1: Bảng luật lọc gói trên firewall Iptables

Order	IP Source	Port Source	IP Destination	Port Destination	Allow/Deny
1	Internet-dev	Any	192.168.2.2	80, 443	Allow
2	Any	Any	192.168.2.3	80, 443	Allow
3	Any	Any	192.168.2.4	587	Allow
4	Manager's IP	Any	192.168.3.1	8080	Allow
5	192.168.3.2	Any	192.168.2.2	22	Allow
6	192.168.1.0/24	Any	192.168.2.2	80, 443	Allow
7	192.168.1.0/24	Any	Internet	Any	Allow
8	192.168.2.4/24	Any	192.168.1.0/24	Any	Allow
9	192.168.2.0/29	Any	Internet	Any	Allow
10	Any	Any	Any	Any	Deny

- Chính sách truy cập mạng được thiết kế dưới dạng bảng luật (Bảng 3.1), và sẽ được cấu hình thành bộ lọc trên firewall Iptable. Firewall sẽ dựa vào bảng luật này để chỉ cho phép các kết nối từ Internet hướng đến các server trong vùng DMZ là được đi vào mạng. Đặc biệt, để kết nối đến Gitlab server thì dòng traffic mang bởi giao thức HTTPS và port đích là 443 thì mới được Firewall cho qua. Điều này cũng với các kết nối xuất phát từ bên trong mạng nội bộ.

Cuối cùng, cài đặt Nginx lên máy chạy firewall Iptables, vô hiệu chức năng Default virtual host của Nginx và rồi triển khai chức năng Reverse proxy trên nó (bằng cách sửa file cấu hình nginx.conf). Các luật sau đây cần được thiết lập trên Nginx để yêu cầu nó theo dõi và chặn các dòng traffic mà payload của nó chứa nội dung khai thác lỗ hổng Log4j trên Jenkins server.

Sau đây là một số lệnh lọc của bộ luật:

```
server {
[...]
   ## Block Log4j
   set $block_log4j 0;
   if ($query_string ~ "\$\{jndi\:") {
      set $block_sql_injections 1;
```

```
    }
    if ($query_string ~ "${jndi:ldap:") {
        set $block_sql_injections 1;
    }
    if ($query_string ~ "\$\{jndi\:(ldap[s]?|rmi|dns|nis|iiop|corba|nds|http):/[\/]?[^\n]+") {
        set $block_sql_injections 1;
    }
[...]
    if ($block_sql_injections = 1) {
        return 403;
    }
```

Trên đây chỉ là một số lệnh lọc cơ bản, cho mục tiêu chặn tấn công vào lỗ hổng Log4j, của bộ luật cần thiết lập trên Ngnix. Dễ thấy, nếu trong payload của traffic định hướng đến Jenkins server, được mạng bởi giao thức http hoặc https, có chứa một trong các chuỗi ký tự "jndi\:", "jndi:ldap:", "jndi\:(ldap[s]?|rmi|)",… thì Ngnix sẽ trả về mã lỗi 403: Trang web bị chặn truy cập.

3.3. Hoạt động của hệ thống

Trong phần này, chúng tôi không bàn về hoạt động của hệ thống tích hợp liên tục đề xuất, mà chỉ bàn về hoạt động của firewall lọc gói, kết hợp reverse proxy, trong hệ thống. Cách mà firewall lọc gói được sử dụng để bảo vệ hệ thống mạng nội bộ bên trong và các server trong hệ thống đề xuất; và cách mà reverse proxy server ngăn chặn những cuộc tấn công vào lỗ hổng Log4j được trình bày ở đây. Cụ thể như sau:

Với bảng luật lọc gói đã được thiết lập, xét một cách tổng quát, firewall chỉ cho phép những dòng kết nối định hướng đến vùng DMZ đi qua nó. Tuy nhiên, để đến được Gitlab server thì dòng traffic phải được mang bởi giao thức https và cổng đến là 443. Nhờ điều này mà các developer cả bên trong mạng nội bộ, lẫn bên ngoài Internet đều có thể kết nối và commit mã nguồn đến Gitlab server. Firewall cũng chỉ cho phép sự kết nối và trao đổi dữ liệu trực tiếp giữa Jenkins server và Gitlab server. Dòng traffic mang message, giao thức https và port 8080, từ Jenkins server đến các máy tính của manager và developer cũng được firewall cho phép đi qua.

Mọi dòng traffic, chính xác là các request, định hướng đến Jenkins server, được mang bởi giao thức http hoặc https đều bị kiểm soát bởi Nginx server. Server này chặn lại các dòng traffic mà payload của nó chứa nội dung phục vụ cho mục đích khai thác lỗ hổng Log4j có thể tồn tại trong các plugin mà Jenkins đang sử dụng. Nginx proxy dựa vào tập luật đã được thiết lập để thực hiện điều này. Như vậy, Jenkins server không chỉ được bảo vệ để chống lại những cuộc tấn công có thể vào chính server này, mà được bảo vệ để chống lại những cuộc tấn công vào lỗ hổng Log4j có thể tồn tại trong các plugin của nó.

4. Thảo luận về hệ thống đề xuất

Hệ thống tích hợp liên tục nội bộ được đề xuất và triển khai trong nghiên cứu này đã đảm bảo những yêu cầu cơ bản của mạng Intranet của các doanh nghiệp làm phần mềm, định hướng phát triển sản phẩm phần mềm theo phương pháp DevOps, áp dụng quy trình CI/CD.

Vì Gitlab server đặt trong mạng nội bộ nên tốc độ và tính sẵn sàng của hệ thống là khá cao. Điều này cũng góp phần giảm chi phí cần cho hoạt động của hệ thống và tiết kiệm được không gian làm việc của doanh nghiệp (vì Manager và Developer có thể làm việc tại nhà, kết nối với Gitlab server và/hoặc Jenkins server

thông qua Internet); Toàn bộ hệ thống bên trong, đặc biệt là Jenkins server và Gitlab server, được bảo vệ bởi Firewall và Reverse proxy. Mọi sự truy cập từ Internet vào mạng nội bộ đều được kiểm soát, chỉ có những dòng traffic định hướng vào vùng DMZ thì mới được đi qua Firewall. Người dùng trên Internet, mà không phải là manager hoặc developer của một dự án phần mềm nào đó của doanh nghiệp thì khó có thể kết nối được đến Gitlab server; Chỉ có kết nối hai chiều giữa Jenkins server và Gitlab server là được phép. Tức là, mọi tấn công nhằm vào Jenkins server, thành phần trọng yếu của hệ thống CI, đều khó có thể vượt qua được sự kiểm soát và ngăn chặn của Firewall, Reverse proxy. Máy tính của Manager và Developer chỉ có thể nhận message từ Jenkins server, nó không được tạo kết nối trực tiếp đến server này.

Đặc biệt, những dòng traffic có chủ đích khai thác lỗ hổng Log4Shell thì không thể đi qua firewall, nó bị reverse proxy chặn lại theo các lệnh lọc đã được thiết lập trong bộ. Đây là nhiệm vụ quan trọng của firewall, nó cũng là đóng góp chính của chúng tôi trong nghiên cứu này.

Tuy nhiên, để cài đặt, cấu hình và vận hành hệ thống này thì đòi hỏi doanh nghiệp phải có nhân sự có chuyên môn trong lĩnh vực quản trị mạng, an ninh mạng và kiện toàn hệ thống mạng.

5. Kết luận

Trong nghiên cứu này, chúng tôi đã đề xuất và cài đặt thành công một hệ thống mạng tích hợp liên tục nội bộ với ba yếu tố đặc trưng: i) Cả CI server và Repository server đều đặt trong mạng nội bộ nên tốc độ và tính sẵn sàng của hệ thống luôn được đảm bảo ở mức cao; ii) Đảm bảo an toàn cho CI server và Repository server theo hướng tiếp cận sử dụng firewall lọc gói kết hợp với reverse proxy server và iii) Firewall không chỉ thực hiện chức năng bảo vệ mạng nội bộ, bảo vệ CI server, bảo vệ Repository server chống tại tấn công từ bên ngoài mà còn bảo vệ Jenkins server khi lỗ hổng CVE-2021-44228 tồn tại trong nó chưa được vá kịp thời.

Chúng tôi sử dụng các công cụ nguồn mở Jenkins, Gitlab, IpTables và Nginx cho mô hình đề xuất, nhưng những đặc trưng của mô hình này có tính tổng quát cao nên khi chúng ta sử dụng các công cụ nguồn mở tương tự để triển khai mô hình thì tính thực tế và tính khả thi của mô hình vẫn được đảm bảo.

Tài liệu tham khảo

[1] Mojtaba S., Muhammad A. B. and Liming Z., *"Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices"*, IEEE Access, vol. 5, 2017, DOI: 10.1109/ACCESS.2017.2685629.

[2] Donca C., Stan O. P., Misaros M., Gota D. And Miclea L., "*Method for Continuous Integration and Deployment Using a Pipeline Generator for Agile Software Projects",* Sensors (Basel), vol. 22, no. 12, 2022, DOI: 10.3390/s22124637.

[3] Sree P. K., Rajkumar P.*, "Survey on Continuous Integration, Deployment and Delivery in Agile and DevOps Practices"*, International Journal of Computer Sciences and Engineering, vol. 4, iss. 4, pp. 213-216, 2016.

[4] Ioannis K. M., Imtiaz H., Claudia A., Fred H., Yann A., Luc D., Xian Z., Christopher J. W., Jeremy L. J., Nicholas H., John T., and Christian N. P., *"Jenkins-CI, an Open-Source Continuous Integration System, as a Scientific Data and Image-Processing Platform",* Society for Laboratory Automation and Screening, SLAS Discovery, vol. 12, pp. 1-12, 2016, DOI: 10.1177/1087057116679993.

[5] Pranoday P. D.,*"CI/CD Pipeline Using Jenkins Unleashed - Solutions While Setting Up CI/CD Processes" (*Book), Apress Berkeley, CA, Edition number 1, 2022, DOI: 10.1007/978-1-4842-7508-5.

[6] Sriniketan M., Vaibhav B.*, "Continuous Integration and Continuous Deployment Pipeline Automation Using Jenkins Ansible"*, IEEE Explore, International Conference on Emerging Trends in Information Technology and Engineering, 2020, DOI: 10.1109/ic-ETITE47903.2020.239.

[7] Rayanagoudar S.F., Hampannavar P. S., Pujari J.D., Parvati V. K.,*"Enhancement of CICD Pipelines with Jenkins BlueOcean",* International Journal Of Computer Sciences And Engineering, vol. 6, iss. 6, pp.1048-1052, 2018, DOI: 10.26438/ijcse/v6i6.10481053.

[8] Behind Java, *"Log4J Vulnerability (Log4Shell) Explained - for Java developers*", Available at: https://www.incibe-cert.es/en/blog/log4shell-analysis-vulnerabilities-log4j, 2021 [Accessed 27 Feb 2023].

[9] Mohammad I., Abdulrahman A. A., Bilal A., *"Role of firewall Technology in Network Security",* International Journal of Innovations & Advancement in Computer Science, vol. 4, iss. 12, pp. 3-6, 2015.

[10] Sahithi D., Tarik E., *"Firewalls Implementation in Computer Networks and Their Role in Network Security"*, Journal of Multidisciplinary Engineering Science and Technology, vol. 2, iss. 3, pp. 408-411, 2015.

[11] Alex Tatistcheff, *"Protecting against Log4j with Secure Firewall & Secure IPS",* Available at: https://blogs.cisco.com/security/protecting-against-log4j-with-secure-firewall-secure-ips?scid=6WCg7b7JiR3&id=1RqLEkeW8BL, 2021 [Accessed 27 Feb 2023].