TẬP ĐOÀN CÔNG NGHIỆP – VIỄN THÔNG QUÂN ĐỘI

BÁO CÁO MINI-PROJECT
Giải pháp nhúng iframe an toàn

NGUYỄN NAM HÁN
nnamhan157@gmail.com.

Chương trình Viettel Digital Talent 2023
Lĩnh vực: Software & Data Engineering

Mentor: Nguyễn Chí Đông

Đơn vị: Ban Công nghệ Thông tin

HÀ NỘI, 05/2023

Tóm tắt nội dung và đóng góp

Hiện nay với sự bùng nổ số lượng và tính năng của các website trên internet việc nhúng nội dung từ các nguồn bên ngoài thông qua iframe trở nên phổ biến và thuận tiện hơn. Với iframe, người phát triển có thể dễ dàng nhúng các ứng dụng và dịch vụ từ các nhà cung cấp khác vào trang web của mình. Điều này giúp tạo ra một trải nghiệm tốt hơn cho người dùng, đồng thời tăng tính linh hoạt và đa dạng hóa trong phát triển web. Bên cạnh đó cũng là rất nhiều lỗ hổng bảo mật của iframe để kẻ xấu lợi dụng thực thi mã độc, tấn công đánh cắp dữ liệu người dùng.

Đề tài đã đưa ra giải pháp ngăn chặn một phần mềm bị kẻ xấu lợi dụng khi bị nhúng thành iframe sử dụng x-frame-option header, content security policy và javascript defense. Tiếp đó, một mô hình microservice sử dụng API gateway và service discovery cũng được đề xuất giúp thêm xác thực để ngăn chặn truy cập trực triếp địa chỉ của iframe bởi src, tránh lộ lọt dữ liệu của trang web bị nhúng mà không có xác thực hay phân quyền. Cuối cùng đề tài triển khai thử nghiệm mô hình trên với các mã nguồn mở là zuul và eureka trên một máy chủ vps và rút ra một số nhận về kết quả.

Qua quá trình thực hiện đề tài, em đã học được cách để giải quyết một vấn đề từ xem xét thực trạng, tìm hiểu các nghiên cứu liên quan đã có, đề xuất giải pháp, triển khai thử nghiệm và đánh giá. Em rất mong nhận được góp ý của anh/chị để hoàn thiện hơn nữa giải pháp và đi tới triển khai để đem lại thêm nhiều giá trị thực tiễn.

MỤC LỤC

DANH MỤC HÌNH VẼ

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

1.1 Động lực tìm hiểu

Iframe, viết tắt của "Inline Frame", là một thành phần HTML mạnh mẽ cho phép nhúng nội dung từ một tài liệu hoặc trang web khác vào trong một trang web hiện tại. Điều này cho phép tích hợp và hiển thị nội dung từ các nguồn bên ngoài một cách dễ dàng và linh hoạt.

Trong quá trình phát triển web, iframe đã đóng một vai trò quan trọng. Trong giai đoạn Web 2.0, iframe được sử dụng rộng rãi để tích hợp và nhúng nội dung từ các nguồn bên ngoài, như video, bản đồ, biểu mẫu và ứng dụng. Điều này mang lại trải nghiệm tương tác và đa dạng cho người dùng là thứ mà Web 1.0 còn hạn chế. Đáp ứng yêu cầu của Responsive Web Design, iframe tiếp tục được sử dụng phổ biến để nhúng nội dung linh hoạt và thích ứng với kích thước màn hình khác nhau. Với vai trò đó, iframe đã đóng góp vào sự phát triển và linh hoạt của web, mở ra khả năng tích hợp và hiển thị nhiều nguồn thông tin trên trang web.

Một trong những ứng dụng phổ biến của iframe là nhúng các bản đồ từ các dịch vụ như Google Maps. Bằng cách nhúng một bản đồ trong frame, ứng dụng có thể hiển thị địa điểm cụ thể, chỉ đường và tạo ra trải nghiệm tương tác với bản đồ trên trang web của mình. Ngoài ra, việc nhúng video từ YouTube, biểu mẫu từ các dịch vụ quản lý email, hoặc widget xã hội từ các mạng xã hội cũng vô cùng thuận tiện khi sử dụng. Iframe cũng cho phép tích hợp các ứng dụng và dịch vụ từ các nhà cung cấp bên thứ ba như nhúng ứng dụng quản lý dự án, trình soạn thảo văn bản, trình xem file PDF, hay bất kỳ ứng dụng web phức tạp nào khác.

Tuy nhiên, việc sử dụng iframe cũng có một số hạn chế như có thể làm giảm tốc độ tải trang do phụ thuộc vào nguồn bên ngoài hay việc công cụ tìm kiếm không đánh giá nội dung trong frame như nội dung chính trên trang làm giảm khả năng xếp hạng và tìm thấy trang web trên công cụ tìm kiếm.

Nghiêm trọng hơn việc ảnh hưởng trải nghiệm người dùng hay SEO là các vấn đề về bảo mật như xác thực nguồn gốc, tấn công cross-site scripting, clickjacking, iframe injection,…

Đề tài **Giải pháp nhúng iframe an toàn** được xây dựng để giải quyết hai vấn đề chính:

- Ngăn chặn một phần mềm bị lợi dụng khi bị nhúng thành iframe để thực hiện các cuộc tấn công theo dõi và đánh cắp thông tin người dùng.
- Ngăn chặn rò rỉ thông tin khi bị truy cập trực tiếp vào địa chỉ iframe khi nhúng các dịch vụ không có xác thực người dùng thành thẻ iframe của ứng dụng.

1.2 Các nghiên cứu hiện nay

Có nhiều nghiên cứu đã được thực hiện để tìm hiểu về các vấn đề bảo mật liên quan đến Iframe. Dưới đây là một số nghiên cứu đáng chú ý:

1. “Beware of Finer-Grained Origins. In Web 2.0 Security and Privacy” (W2SP 2008)

Nghiên cứu đưa ra tổng quan các vấn đề an ninh của web 2.0 trong đó có việc tích hợp nội dung từ các nguồn khác nhau. Nghiên cứu này đề xuất các phương pháp và cách thức tấn công giúp hiểu rõ hơn về các vấn đề an ninh liên quan đến Iframe.

2. “Busting frame busting: a study of clickjacking vulnerabilities at popular sites”

IEEE Oakland Web 2.0 Security and Privacy (W2SP 2010)

Nghiên cứu này tập trung vào tấn công Clickjacking và những cách thức tấn công tiềm năng khác liên quan đến Iframe. Nghiên cứu này cung cấp những gợi ý và phương pháp để ngăn chặn và phát hiện các cuộc tấn công Clickjacking.

3. “Mitigating Cross-Site Scripting Attacks with a Content Security Policy” (2016)

Content Security Policy (CSP) có thể giúp các nhà phát triển ứng dụng Web và quản trị viên máy chủ kiểm soát tốt hơn nội dung trang web và tránh các lỗ hổng đối với tập lệnh chéo trang (XSS). Trong các thử nghiệm, việc triển khai CSP của tác giả đã giảm thiểu tất cả các loại tấn công XSS trong bốn trình duyệt phổ biến.

4. “On the Security of Parsing Security-Relevant HTTP Headers in Modern Browsers” (2022)

Mặc dù cộng đồng bảo mật nhận thức được tầm quan trọng của các HTTP Header liên quan đến bảo mật, nhưng các ứng dụng cũ và các yêu cầu riêng lẻ từ các bên khác nhau đã dẫn đến các cấu hình có thể không an toàn của các tiêu đề này. Ngay cả khi các tiêu đề bảo mật cụ thể được định cấu hình chính xác, các xung đột về chức năng của chúng có thể dẫn đến các hành vi và lỗ hổng không lường trước được của trình duyệt. Nghiên cứu đã phân tích các cấu hình tiêu đề cũng như các xung đột trong các tiêu đề sau đó thử nghiệm trên đầy đủ những trình duyệt phổ biến hiện nay.

Các nghiên cứu trên đề cập đến các vấn đề bảo mật phổ biến liên quan đến iframe và cung cấp cái nhìn sâu hơn về các mối đe dọa và biện pháp bảo mật. Việc theo dõi các nghiên cứu này có thể giúp các nhà phát triển web và chuyên gia bảo mật hiểu rõ hơn về các lỗ hổng bảo mật còn tồn tại và phát triển các biện pháp bảo vệ hiệu quả.

1.3 Đóng góp của đề tài

Đề tài có 3 đóng góp chính như sau:

1. Nghiên cứu, tổng hợp các hạn chế khi sử dụng iframe.
2. Đề xuất mô hình nhúng iframe an toàn khi sử dụng các mã nguồn mở.
3. Triển khai thử nghiệm mô hình thử nghiệm đảm bảo an toàn cho nhúng iframe.

1.4 Bố cục của đề tài

Phần còn lại của báo cáo đề tài được tổ chức như sau:

Chương 2 trình bày các cơ sở lý thuyết của các phương pháp mà tôi đề xuất trong đề tài bao gồm cách nhúng iframe một trang web, mô hình microservice và cách sử dụng api gateway và service discovery.

Chương 3 trình bày các phương pháp đề xuất để ngăn chặn phần mềm bị nhúng bởi các phần mềm khác, ngăn chặn truy cập trực tiếp tới iframe và mô hình microservice để bảo vệ.

Chương 4 trình bày thử nghiệm mô hình microservice với open source zuul và eureka từ đó đưa ra một số kết quả so sánh, nhận xét.

Chương 5 là chương tổng kết, tóm tắt lại những kết quả của đề tài cũng như đề xuất những hướng phát triển trong tương lai.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

2.1 Nhúng iframe cho trang web

Nhúng iframe là một công nghệ web cho phép nhúng một trang web hoặc một phần nội dung từ một trang web khác vào trang web hiện tại. Thông qua việc sử dụng thẻ <iframe> có thể hiển thị nội dung từ một trang web khác mà không cần chuyển hướng trực tiếp tới địa chỉ nguồn.

Cú pháp đơn giản để nhúng một iframe trong HTML như sau:

```
<iframe src="đường_dẫn_đến_tài_liệu_ngoài"></iframe>
```

Trong đó, “src” là thuộc tính quan trọng nhất của thẻ iframe và nó định rõ đường dẫn tới tài liệu hoặc trang web cần nhúng.

Một số thuộc tính quan trọng khác của ifame bao gồm:

- “name”: dùng để đặt tên cho frame.
- “width” và “height”: được sử dụng để xác định kích thước chiều rộng và chiều cao của iframe.
- “frameborder”: xác định xem khung viền của iframe có hiển thị hay không. Giá trị "0" sẽ tắt khung viền, và "1" sẽ bật khung viền.
- “allowfullscreen”: cho phép iframe mở rộng để toàn màn hình khi được kích hoạt.
- “sandbox”: thiết lập các ràng buộc bảo mật cho iframe, giới hạn quyền truy cập và khả năng tương tác với trang gốc.

Ví dụ sau đây minh họa cách nhúng một trang web Google vào một giao diện viết bởi HTML với địa chỉ Google là https://www.google.com kích thước 800x600px và không hiển thị khung viền :

```
<iframe src="https://www.google.com" width="800"
height="600" frameborder="0"></iframe>
```

Trong một số trường hợp địa chỉ trang web được nhúng yêu cầu xác thực thông qua header thì việc tạo thẻ iframe với thuộc tính “src” sẽ chưa thể thực hiện được trực tiếp. Thay vào đó, có thể tạo một thẻ iframe bằng JavaScript và sau đó tạo một yêu cầu HTTP sử dụng XMLHttpRequest hoặc fetch API để lấy nội dung từ URL cần nhúng. Trong yêu cầu HTTP này, thêm header xác thực bằng cách sử dụng phương thức setRequestHeader(). Khi nhận được phản hồi, JavaScript gắn nội dung vào thẻ IFrame đã tạo trước đó bằng cách sử dụng thuộc tính srcdoc hoặc innerHTML.

```
const iframe = document.getElementById('myIframe');
```

```
const url = 'https://example.com/your-html-file.html';
// Thay thế với URL của tệp HTML
const token = 'your_bearer_token';  // Bearer token để
xác thực

fetch(url, {
  headers: {
    Authorization: `Bearer ${token}`
  }
})
  .then(response => response.text())
  .then(html => {
    // Tạo một iframe tạm thời để truy cập tài liệu
    const                  tempIframe                  =
document.createElement('iframe');
    tempIframe.style.display = 'none';
    document.body.appendChild(tempIframe);

    // Thiết lập HTML vào iframe tạm thời
    tempIframe.contentDocument.open();
    tempIframe.contentDocument.write(html);
    tempIframe.contentDocument.close();

    // Tải CSS và áp dụng vào iframe chính
    const   cssUrl   =   'https://example.com/your-css-
file.css';  // Thay thế với URL của tệp CSS
    fetch(cssUrl)
      .then(response => response.text())
      .then(css => {
        const                  style                  =
tempIframe.contentDocument.createElement('style');
        style.textContent = css;

tempIframe.contentDocument.head.appendChild(style);

        // Thiết lập nội dung HTML và CSS vào iframe
chính
        iframe.srcdoc                                  =
tempIframe.contentDocument.documentElement.outerHTML;

        // Loại bỏ iframe tạm thời
        document.body.removeChild(tempIframe);
```

```
    })
    .catch(error => console.error(error));
})
.catch(error => console.error(error));
```

2.2 Các vấn đề an toàn khi nhúng iframe

2.2.1 Trang web bị nhúng bởi một trang web khác với mục đích xấu

Khi trang web không ngăn chặn việc bị nhúng trong trang web khác, kẻ tấn công có thể tạo ra một trang web hoặc một phần tử trang web giả mạo và nhúng trang web mục tiêu vào trong đó, che giấu nội dung gốc. Khi người dùng truy cập vào trang web hoặc tương tác với phần tử giả mạo, thực tế họ đang tương tác với nội dung trang web mục tiêu mà họ không biết hoặc khi người dùng đăng nhập vào trang web mục tiêu, kẻ tấn công có thể lấy được token đăng nhập và sử dụng nó.

Ví dụ kẻ tấn công có thể tạo ra một trang web giả mạo của một trang web có uy tín, như ngân hàng hoặc dịch vụ trực tuyến, và nhúng nội dung của trang web đó vào trong trang web giả mạo. Với việc cài các thủ thuật keylogger để theo dõi việc thao tác, người dùng có thể bị lừa để nhập thông tin đăng nhập hoặc thông tin cá nhân vào trang web giả mạo, và thông tin này sẽ được kẻ tấn công thu thập.

Đó là lý do các dịch vụ uy tín không cho phép tùy ý nhúng trực tiếp trang web của họ mà không có phải thông qua nhiều bước xác thực.

Ví dụ Google không cho nhúng trang login trực tiếp qua iframe sử dụng src:

Hình 1: Google từ chối việc nhúng trực tiếp trang login bởi một iframe

Cross-site scripting (XSS), clickjacking và iframe injection là những lỗ hổng bảo mật nổi tiếng về nhúng trang web có thể bị khai thác bởi các kẻ xấu để đe dọa bảo mật của một trang web hoặc sự riêng tư của người dùng.

1. Cross-site Scripting (XSS)

XSS xảy ra khi một kẻ tấn công chèn các đoạn mã độc hại vào một trang web được xem bởi người dùng khác. Điều này có thể xảy ra khi đầu vào của người dùng không được kiểm tra hoặc làm sạch đúng cách bởi trang web. Khi người dùng không ngờ đang xem trang bị nhiễm mã độc, các đoạn mã độc hại có thể thực thi trong trình duyệt của họ, cho phép kẻ tấn công đánh cắp thông tin nhạy cảm, thực hiện các hành động thay mặt người dùng hoặc thao túng nội dung của trang web. Để ngăn chặn XSS, trang web cần thực hiện các biện pháp bảo mật như kiểm tra và làm sạch đầu vào của người dùng, mã hóa dữ liệu trước khi hiển thị, và sử dụng HTTP Only và Secure cookies để cookie chỉ được thao tác bởi server mà không bị thao tác bởi các script phía người dùng.

2. Clickjacking

Clickjacking là một kỹ thuật mà kẻ tấn công lừa người dùng nhấp chuột vào một phần tử được tạo một cách độc hại trong khi làm cho họ tin rằng họ đang nhấp chuột vào một phần tử hợp pháp của một trang web khác. Điều này thường được thực hiện bằng cách chồng lên hoặc khung đè lên trang web mục tiêu với các phần tử trong suốt hoặc ẩn. Kết quả là, kẻ tấn công có thể thực hiện các hành động không đúng ý của người dùng mà họ không biết hoặc không đồng ý, như nhấp chuột vào các nút hoặc liên kết thực hiện các hoạt động độc hại. Để ngăn chặn Clickjacking, các trang web có thể sử dụng các phương pháp như HTTP Header X-Frame-Options hoặc Content Security Policy (CSP) để ngăn iframe hoặc frame trong trang web được nhúng vào trang khác.

3. Iframe injection

Iframe injection liên quan đến việc chèn một phần tử iframe vào trong trang web. Kẻ tấn công có thể chèn một iframe để hiển thị nội dung từ một trang web khác hoặc để thực hiện các hành động độc hại trong phạm vi trang web hiện tại. Việc chèn iframe một cách không an toàn có thể dẫn đến việc lợi dụng các lỗ hổng khác để tấn công hoặc đánh cắp thông tin của người dùng. Để ngăn chặn iframe injection, trang web cần thực hiện kiểm tra và xử lý đầu vào của người dùng một cách an toàn, kiểm tra và giới hạn nguồn gốc của các iframe được nhúng và đảm bảo rằng chỉ những trang web tin cậy mới được nhúng vào iframe.

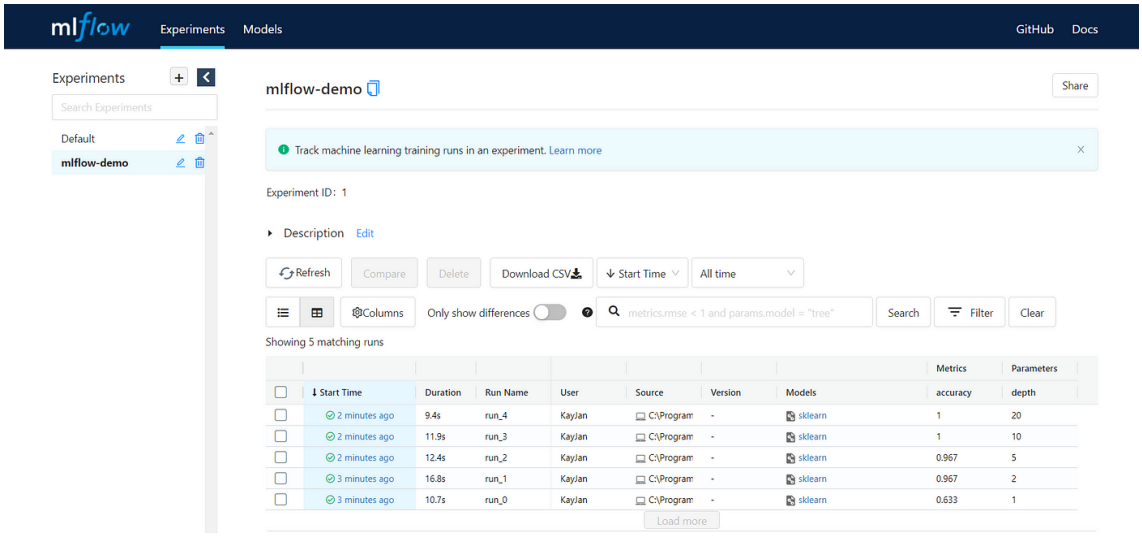
2.2.2 Thiếu an toàn khi trang web khác thành một iframe

Hiện nay để đáp ứng sự hoàn thiện cũng như đa dạng tính năng của phần mềm đồng thời cân đối với tiết kiệm thời gian và chi chí phát triển, các phần mềm có xu hướng sử dụng các trang web khác hay triển khai các mã nguồn mở như một dịch vụ bổ trợ tính năng của mình. Điều này dẫn đến thiếu một cơ chế thống nhất giữa các dịch vụ để xác thực, phân quyền đảm bảo an toàn dữ liệu.

Phổ biến là khi nhúng một nguồn tài nguyên open source không có cơ chế xác thực (authentication) vào trong một iframe, có thể xảy ra các vấn đề bảo mật như bypass

bảo mật. Nếu nguồn tài nguyên open source không yêu cầu xác thực, việc nhúng nó vào một iframe sẽ cho phép bất kỳ ai truy cập trang web của bạn xem nội dung của nguồn tài nguyên đó mà không cần xác thực. Điều này có thể là một vấn đề nếu nguồn tài nguyên chứa thông tin nhạy cảm hoặc yêu cầu quyền truy cập hạn chế.

Ví dụ MLFlow (open source hỗ trợ quản lý các mô hình học máy) thiếu sự phân quyền dù đã có xác thực, người dùng có thể truy cập trực tiếp địa chỉ url để lấy bất kỳ dữ liệu nào của người dùng khác. Do đó nếu sử dụng trực tiếp MLFlow như một dịch vụ hoặc nhúng thành iframe trên giao diện thì chưa an toàn.



Hình 2: Lấy tất cả dữ liệu MLFlow thông qua url

2.3 API gateway

API Gateway là một dịch vụ trung gian giữa người dùng và các dịch vụ trong một hệ thống mà ứng dụng được phân tách thành các thành phần nhỏ hơn gọi là microservices. Nó đóng vai trò là một cổng kết nối cho việc quản lý, bảo mật và kiểm soát lưu lượng truy cập vào các API của hệ thống.

Tác dụng chính của API Gateway bao gồm:

1. Quản lý lưu lượng: API Gateway giúp quản lý lưu lượng truy cập vào các API. Bằng cách áp dụng giới hạn tốc độ, cân bằng tải và quản lý các yêu cầu đồng thời, API Gateway đảm bảo rằng các dịch vụ backend không bị quá tải và hoạt động một cách ổn định.

2. Bảo mật và xác thực: API Gateway cung cấp các tính năng bảo mật để bảo vệ các API khỏi các cuộc tấn công. Nó cho phép xác thực người dùng, kiểm tra quyền truy cập và áp dụng các chính sách bảo mật để đảm bảo rằng chỉ những người dùng có quyền truy cập hợp lệ mới có thể sử dụng các API.

3. Định tuyến yêu cầu: API Gateway xử lý việc định tuyến yêu cầu từ khách hàng đến các dịch vụ backend tương ứng. Nó cho phép bạn định tuyến yêu cầu dựa trên các tiêu chí như đường dẫn, phương thức, tiêu đề hoặc thông tin khác để chuyển tiếp yêu cầu đến dịch vụ backend phù hợp.

4. Quản lý phiên và lưu trữ cache: API Gateway có khả năng quản lý phiên và lưu trữ cache để tối ưu hóa hiệu suất của hệ thống. Việc lưu trữ phiên giúp theo dõi trạng thái và dữ liệu của khách hàng trong quá trình giao tiếp với các API. Bộ nhớ cache giúp giảm tải cho các dịch vụ backend bằng cách lưu trữ kết quả của các yêu cầu trước đó và phục vụ lại chúng khi có yêu cầu tương tự.

5. Giám sát và phân tích: API Gateway cung cấp khả năng giám sát và phân tích lưu lượng truy cập vào các API. Bằng cách thu thập và phân tích dữ liệu, bạn có thể theo dõi hiệu suất, lỗi và các chỉ số quan trọng khác của

Dưới đây là một số API Gateway phổ biến được sử dụng trong các hệ thống:

- Zuul: Một API Gateway phát triển bởi Netflix, Zuul cung cấp các tính năng định tuyến, bảo mật, quản lý lưu lượng và kiểm soát truy cập. Zuul là một thành phần quan trọng trong kiến trúc dựa trên microservices và được sử dụng để xử lý yêu cầu từ client và chuyển tiếp chúng đến các dịch vụ tương ứng.
- Amazon API Gateway: Dịch vụ quản lý API Gateway từ Amazon Web Services (AWS), cung cấp các tính năng như định tuyến, bảo mật, quản lý lưu lượng, ghi nhật ký và phân tích.
- Kong: Một nền tảng mã nguồn mở cho API Gateway và quản lý API, hỗ trợ nhiều tính năng như định tuyến, ghi nhật ký, bảo mật, cân bằng tải và kiểm soát truy cập.
- Apigee: Một dịch vụ API Gateway và quản lý API của Google Cloud, cung cấp các tính năng như định tuyến, bảo mật, quản lý lưu lượng, phân tích và kiểm soát truy cập.
- Azure API Management: Dịch vụ quản lý API Gateway của Microsoft Azure, cho phép quản lý API, bảo mật, cân bằng tải, ghi nhật ký và quản lý lưu lượng.
- NGINX Plus: Bản mở rộng của máy chủ web NGINX, cung cấp các tính năng của một API Gateway như định tuyến, bảo mật, cân bằng tải và kiểm soát truy cập.
- Ambassador: Một API Gateway mã nguồn mở được xây dựng trên nền tảng Kubernetes, cung cấp các tính năng như định tuyến, bảo mật, quản lý lưu lượng và phân tích.

2.4 Service discovery

Service Discovery là một khái niệm trong kiến trúc microservice, đề cập đến quá trình tự động tìm kiếm và khám phá các dịch vụ có sẵn trong một mạng máy tính. Nó giúp các thành phần trong hệ thống phân tán tìm thấy và kết nối với nhau một cách tự động và linh hoạt.

Công dụng chính của Service Discovery bao gồm:

1. Độ mở rộng và linh hoạt: Khi một hệ thống phân tán có nhiều dịch vụ, việc quản lý và theo dõi các dịch vụ này trở nên khó khăn. Service Discovery giúp giảm bớt công việc quản lý bằng cách cung cấp cơ chế tự động tìm kiếm và cập nhật các

dịch vụ trong mạng máy tính. Điều này giúp hệ thống dễ dàng mở rộng và thay đổi mà không cần phải thay đổi cấu hình thủ công.

2. Định tuyến động: Service Discovery cung cấp khả năng định tuyến động cho các yêu cầu giữa các dịch vụ. Thay vì phải cấu hình địa chỉ IP và cổng của từng dịch vụ một cách thủ công, Service Discovery cho phép các thành phần trong hệ thống tìm thấy địa chỉ IP và cổng của dịch vụ cần truy cập một cách tự động. Điều này giúp tạo ra một hệ thống linh hoạt và có khả năng mở rộng.

3. Giảm thời gian chết (downtime): Khi một dịch vụ trong hệ thống phân tán bị lỗi hoặc không hoạt động, Service Discovery có thể phát hiện và loại bỏ dịch vụ đó khỏi danh sách các dịch vụ hoạt động. Điều này giúp giảm thời gian chết và đảm bảo rằng các yêu cầu sẽ không được chuyển tiếp đến dịch vụ không hoạt động.

4. Load balancing: Service Discovery cung cấp khả năng cân bằng tải tự động giữa các phiên bản hoặc phiên bản sao của cùng một dịch vụ. Nó giúp phân phối lưu lượng truy cập một cách công bằng và tăng khả năng chịu tải cho hệ thống.

5. Khả năng mở rộng: Khi một dịch vụ mới được triển khai, Service

Discovery tự động phát hiện và cập nhật danh sách các dịch vụ có sẵn. Điều này giúp hệ thống dễ dàng mở rộng và tích hợp các dịch vụ mới mà không cần can thiệp vào cấu hình hiện có. Một số công cụ phổ biến được sử dụng để triển khai Service Discovery là:

- Eureka: Eureka là một công cụ Service Discovery mã nguồn mở phát triển bởi Netflix. Nó cho phép các dịch vụ đăng ký và tìm kiếm thông qua tên dịch vụ. Eureka cung cấp các tính năng như tự động đăng ký, xóa bỏ và cập nhật dịch vụ, cùng với khả năng khám phá dịch vụ qua các mẫu định tuyến.
- Consul: Consul là một công cụ Service Discovery mã nguồn mở phát triển bởi HashiCorp. Nó cung cấp khả năng đăng ký, tìm kiếm và khám phá dịch vụ, cùng với tính năng cân bằng tải và phát hiện dịch vụ tự động. Consul hỗ trợ nhiều phương thức định tuyến và có tích hợp sẵn với các công cụ khác như DNS và Key-Value Store.
- ZooKeeper: ZooKeeper là một dự án mã nguồn mở của Apache, được sử dụng như một hệ thống phân tán để quản lý dịch vụ và khám phá dịch vụ. Nó cung cấp các tính năng như đăng ký, tìm kiếm, phát hiện và cân bằng tải dịch vụ. ZooKeeper được sử dụng rộng rãi trong các hệ thống phân tán để quản lý và theo dõi các thành phần.

CHƯƠNG 3. PHƯƠNG PHÁP ĐỀ XUẤT

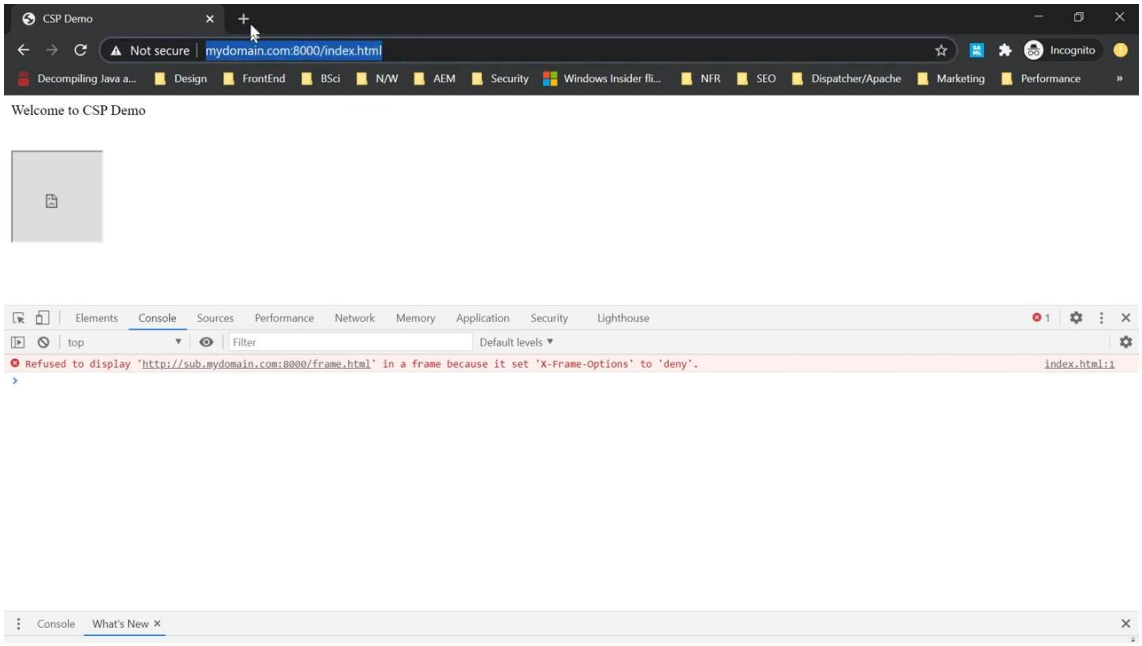
3.1 Ngăn chặn phần mềm bị nhúng bởi iframe của các phần mềm khác

Để ngăn chặn trang web bị nhúng vào bởi trang web khác, các phương pháp phổ biến là sử dụng X-Frame-Options Header, Content Security Policy và JavaScript Defense.

3.1.1 X-Frame-Options Header (thiếu hình ảnh minh họa, vd)

Header X-Frame-Options được gắn kết với phản hồi HTTP từ máy chủ và chỉ định các chính sách quy định cách trình duyệt hiển thị trang web trong một thẻ iframe hoặc frame. Có ba giá trị được sử dụng cho X-Frame-Options:

- DENY: Chặn mọi trình duyệt khỏi việc nhúng trang web trong một thẻ iframe hoặc frame.
- SAMEORIGIN: Cho phép trang web chỉ được nhúng trong một thẻ iframe hoặc frame nếu đang được truy cập từ cùng một nguồn gốc (origin). Nếu trang web được truy cập từ nguồn khác, nó sẽ bị chặn.
- ALLOW-FROM uri: Cho phép trang web chỉ được nhúng trong một thẻ iframe hoặc frame từ một nguồn cụ thể được chỉ định bởi URI. Ví dụ: ALLOW-FROM https://example.com.



Hình 3: Chặn nhúng iframe bởi X-Frame-Option Header

Ví dụ, để chặn trang web khỏi việc nhúng trong một thẻ iframe hoặc frame từ bất kỳ nguồn nào, X-Frame-Options sẽ được cấu hình là:

```
X-Frame-Options: DENY
```

Hoặc nếu chỉ muốn cho phép trang web được nhúng trong một thẻ iframe hoặc frame từ cùng một địa chỉ IP và giống PORT, có thể đặt header X-Frame-Options như sau:

```
X-Frame-Options: SAMEORIGIN
```

Cách đặt header X-Frame-Options phụ thuộc vào ngôn ngữ lập trình hoặc framework. Ví dụ trong Java với Spring Framework:

```
import
org.springframework.web.bind.annotation.GetMapping;
import
org.springframework.web.bind.annotation.RestController
;

import javax.servlet.http.HttpServletResponse;

@RestController
public class MyController {

    @GetMapping("/my-page")
    public String myPage(HttpServletResponse response)
{
        response.setHeader("X-Frame-Options", "DENY");
        // ...
    }
}
```

Đối với NGINX cần chèn mã sau vào tập tin /etc/nginx/nginx.conf:

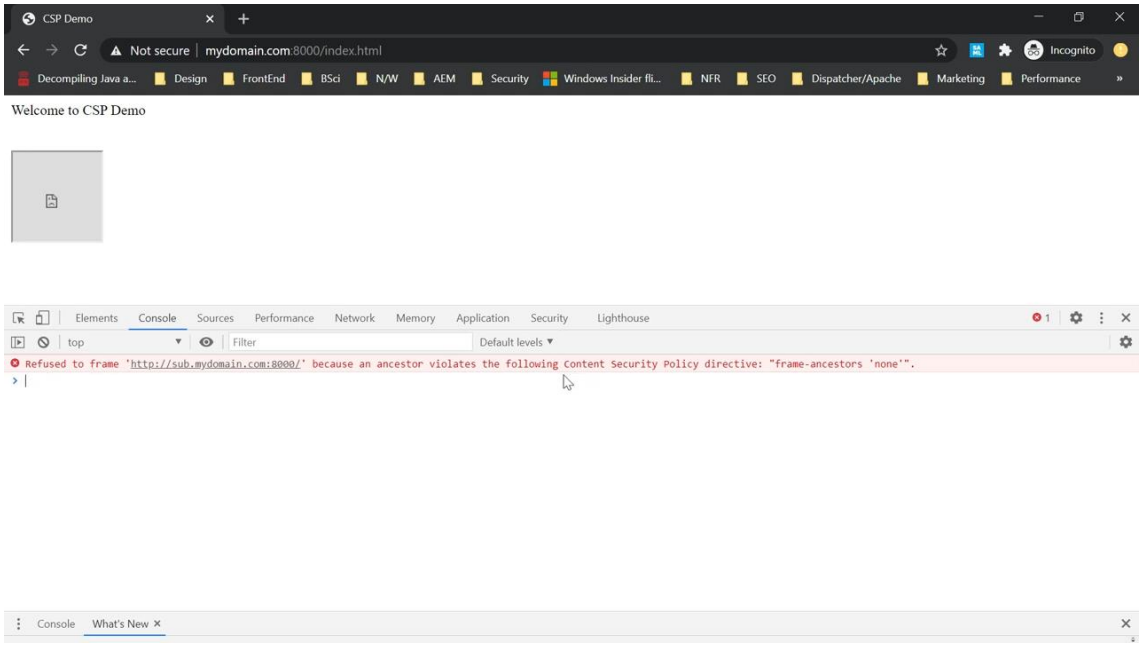
```
server {
    ...
    add_header X-Frame-Options DENY always;
    ...
}
```

3.1.2 Content Security Policy (CSP)

Sử dụng CSP để xác định chính sách an ninh cho trang web. CSP giúp giảm rủi ro các cuộc tấn công như XSS (Cross-Site Scripting) và clickjacking bằng cách kiểm soát và hạn chế việc thực thi mã JavaScript không an toàn từ các nguồn không tin cậy.

Directive "frame-ancestors" trong CSP được sử dụng để xác định các trang web được phép nhúng vào một thẻ iframe hoặc frame. Directive "frame-ancestors" có một số giá trị như:

- 'self': Cho phép nhúng trang web vào trang web cùng một nguồn gốc (origin).
- 'none': Không cho phép nhúng trang web vào từ bất kỳ nguồn gốc nào.
- 'unsafe-inline': Cho phép nhúng trang web vào từ bất kỳ nguồn gốc nào, bao gồm cả các nguồn gốc không an toàn.
- 'https://example.com': Cho phép nhúng trang web vào trang web có một nguồn gốc cụ thể.

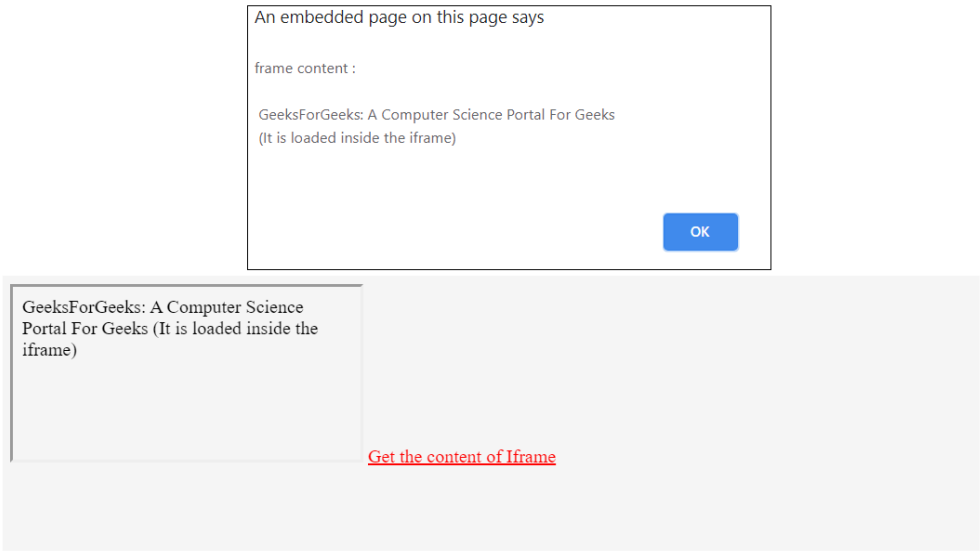


Hình 4: Chặn nhúng iframe bởi CSP

Ví dụ, dùng directive "frame-ancestors" để chỉ cho phép trang web được nhúng vào từ trang web có nguồn gốc là "http://example.com" và "https://example.com", tập tin /etc/nginx/nginx.conf của NGINX cần được cấu hình:

```
server {
    ...
    add_header Content-Security-Policy "frame-
ancestors http://example.com https://example.com;";
    ...
}
```

3.1.3 JavaScript Defense



Hình 5: Sử dụng JS để kiểm tra trang web bị nhúng bởi

Sử dụng JavaScript để kiểm tra nếu trang web có đang chạy trong một khung nhúng và tắt hoặc chuyển hướng nếu có. Ví dụ kiểm tra với đoạn mã JavaScript như sau:

```
function iniFrame() {
          if(window.self !== window.top) {
          document.write("The page is in an iFrame");
     }
     else {
          redirectOrigin(); //Chuyển hướng tới tới trực
tiếp địa chỉ trang web
     }
}
```

3.2 Mô hình microservice để nhúng iframe an toàn

Mô hình được thiết kế với API Gateway sẽ trở thành trung tâm phân phối các dịch vụ và chịu trách nhiệm đảm bảo xác thực và phân quyền tất cả các dịch vụ. Thay vì thực hiện việc xác thực và phân quyền trong từng dịch vụ riêng lẻ. Nhờ đó, có thể dễ dàng thêm các dịch vụ là các mã nguồn mở hay tập trung vào phát triển chức năng cốt lõi của ứng dụng mà không cần quan tâm đến việc xác thực của từng dịch vụ, trang web có thể nhúng vào bất kỳ trang web nào muốn mà không lo các vấn đề lộ đường dẫn không an toàn.

Một ví dụ của kết quả mô hình là việc triển khai thêm dịch vụ Mlflow để quản lý mô hình học máy, thêm Grafana để theo dõi log, nhúng các biểu đồ của Grafana lên giao diện hệ thống bằng các iframe với url đã được bảo vệ bởi API Gateway.

Chi tiết hơn, mô hình sử dụng API Gateway và Service Discovery còn giúp quản lý giao tiếp giữa các dịch vụ trong một hệ thống phân tán, cung cấp khả năng tự động phát hiện, đăng ký và điều phối yêu cầu đến các dịch vụ tương ứng.

1. Đăng ký dịch vụ:

- Khi một dịch vụ back-end được triển khai và sẵn sàng hoạt động, nó đăng ký thông tin của mình (ví dụ: địa chỉ IP, cổng, metadata) với Service Discovery.

- Thông tin đăng ký này cho phép các dịch vụ khác trong hệ thống phát hiện và liên lạc với dịch vụ này.

2. Phát hiện dịch vụ:

- Khi một dịch vụ cần giao tiếp với một dịch vụ khác, nó sử dụng Service Discovery để tìm kiếm địa chỉ và thông tin liên quan của dịch vụ mục tiêu.

- Service Discovery cung cấp cách để tìm kiếm dịch vụ dựa trên tên dịch vụ hoặc các tiêu chí khác như loại dịch vụ, thẻ (tags) hoặc metadata.

3. Giao tiếp thông qua API Gateway:

- Sau khi dịch vụ đã xác định được địa chỉ của dịch vụ mục tiêu từ Service Discovery, nó gửi yêu cầu đến API Gateway.

- API Gateway nhận yêu cầu từ client và điều phối nó đến dịch vụ mục tiêu dựa trên thông tin từ Service Discovery.

- API Gateway có thể sử dụng các chiến lược định tuyến và điều phối dựa trên trạng thái hoặc cân bằng tải để quyết định dịch vụ nào sẽ xử lý yêu cầu.

4. Định tuyến và điều phối:

- API Gateway có khả năng định tuyến yêu cầu từ client đến dịch vụ phù hợp dựa trên nhiều yếu tố, chẳng hạn như loại yêu cầu, URL, thẻ (tags) hoặc thông tin người dùng.

- Nó cũng có thể thực hiện cân bằng tải và quản lý phiên (session) để đảm bảo tải cân bằng và khả năng chịu lỗi trong hệ thống.

API Gateway cung cấp các tính năng xác thực và phân quyền để bảo vệ và kiểm soát quyền truy cập đến các API. Dưới đây là mô tả chi tiết về hai tính năng này:

5. Xác thực (Authentication):

- API Gateway hỗ trợ các phương pháp xác thực khác nhau như API Key, JWT (JSON Web Tokens), OAuth và hoặc sử dụng kết hợp nhiều phương thức.

- API Key: Đây là một chuỗi khóa được tạo ra bởi API Gateway và cung cấp cho người dùng hoặc ứng dụng. Người dùng hoặc ứng dụng sử dụng khóa này để xác thực với API Gateway trước khi truy cập vào API.

- JWT: API Gateway có thể kiểm tra và xác thực JWT để đảm bảo rằng yêu cầu API được gửi từ một nguồn tin cậy. JWT chứa các thông tin về người dùng hoặc ứng dụng và được ký số để đảm bảo tính toàn vẹn của nó.

- OAuth: API Gateway hỗ trợ xác thực OAuth để cho phép người dùng ủy quyền quyền truy cập vào API của họ cho các ứng dụng bên thứ ba. OAuth cho phép

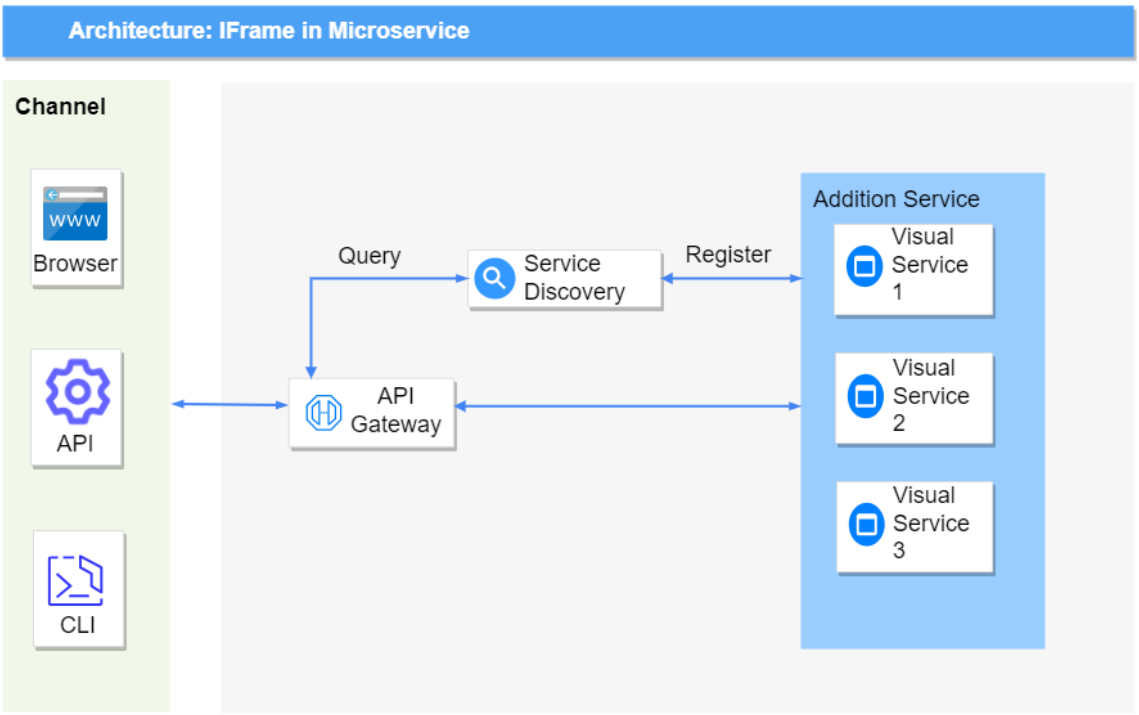
người dùng kiểm soát quyền truy cập và cung cấp một cách an toàn để chia sẻ thông tin xác thực.

6. Phân quyền (Authorization):

- API Gateway cho phép kiểm soát quyền truy cập vào các API bằng cách xác định các quyền và vai trò cho người dùng hoặc ứng dụng.

- Các phân quyền dựa trên vai trò (Role-Based Access Control – RBAC) xác định các vai trò khác nhau và gán các quyền tương ứng cho từng vai trò. Ví dụ, vai trò "quản trị viên" có quyền truy cập đầy đủ vào tất cả các API và vai trò "người dùng" chỉ có quyền truy cập vào một số API cụ thể.

- Các phân quyền tùy chỉnh (Custom Authorization): Ngoài RBAC, API Gateway cung cấp khả năng tạo ra các quy tắc phân quyền tùy chỉnh dựa trên các yêu cầu đặc biệt như kiểm soát quyền truy cập theo yếu tố nhóm người dùng, địa chỉ IP, thời gian và nhiều hơn nữa.



Hình 6: Mô hình nhúng iframe an toàn

CHƯƠNG 4. THỰC NGHIỆM

4.1 Cài đặt mô hình sử dụng open source

Mô hình sử dụng Zuul API Gateway và Eureka Service Discovery là một mô hình phổ biến trong kiến trúc dựa trên microservices. Dưới đây là mô tả chi tiết về mô hình này:

1. Zuul API Gateway:

- Zuul là một API Gateway phát triển bởi Netflix và được sử dụng rộng rãi trong kiến trúc dựa trên microservices.
- Zuul cung cấp các chức năng cơ bản của một API Gateway như điều phối yêu cầu, bảo mật, xử lý giao thức, kiểm soát và giám sát.
- Nó hoạt động như một điểm cuối (endpoint) cho các yêu cầu từ client và điều phối chúng đến các dịch vụ back-end tương ứng.
- Zuul có khả năng thực hiện các chức năng như bộ định tuyến (routing), lọc (filtering) và chuyển tiếp (forwarding) yêu cầu đến các dịch vụ back-end.

2. Eureka Service Discovery:

- Eureka là một dịch vụ phát hiện dịch vụ mã nguồn mở của Netflix, được sử dụng để quản lý và phát hiện các dịch vụ trong một môi trường phân tán.
- Eureka cho phép các dịch vụ đăng ký thông tin của mình và tra cứu thông tin về các dịch vụ khác trong cụm (cluster).
- Các dịch vụ sẽ đăng ký với Eureka server và cập nhật địa chỉ IP, cổng và các metadata liên quan.
- Các dịch vụ khác có thể sử dụng Eureka để tìm kiếm và truy cập các dịch vụ cần thiết mà không cần biết trước vị trí cụ thể của chúng.

Khi kết hợp Zuul API Gateway và Eureka Service Discovery, mô hình hoạt động như sau:

1. Đăng ký dịch vụ với Eureka:

- Khi một dịch vụ back-end được triển khai và sẵn sàng hoạt động, nó sẽ đăng ký thông tin của mình với Eureka server.
- Thông tin đăng ký này bao gồm tên dịch vụ, địa chỉ IP, cổng và các metadata liên quan.

2. Phát hiện dịch vụ thông qua Eureka:

- Khi một dịch vụ cần giao tiếp với một dịch vụ khác, nó sử dụng Eureka để tìm kiếm địa chỉ và thông tin liên quan của dịch vụ mục tiêu.
- Dịch vụ truy vấn Eureka server để lấy

danh sách các dịch vụ đã đăng ký và thông tin về chúng.

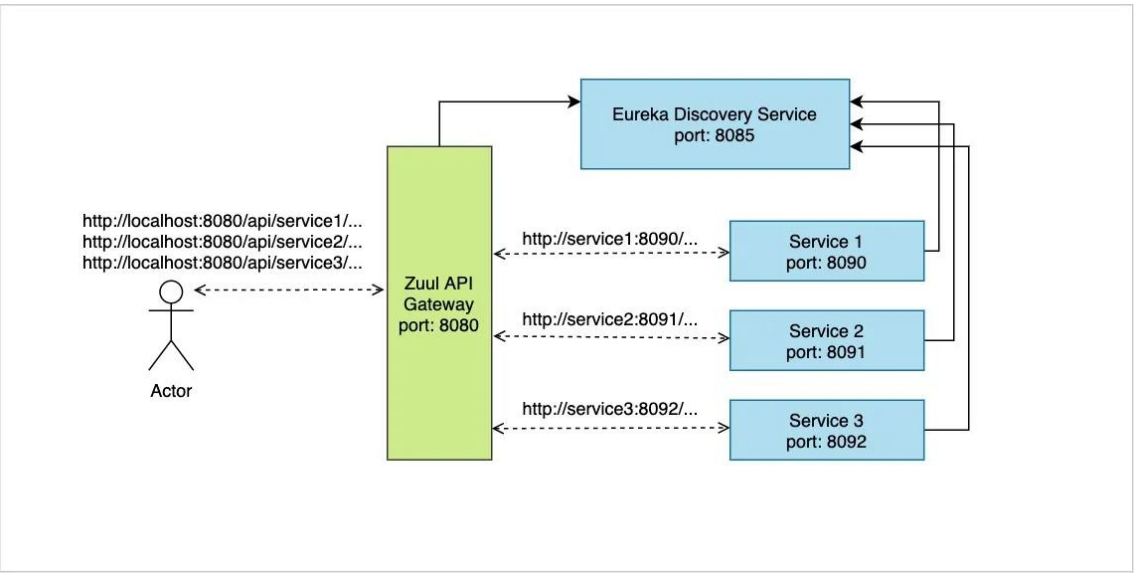
3. Định tuyến và điều phối thông qua Zuul:

- Khi một yêu cầu từ client được gửi đến, nó sẽ truy cập Zuul API Gateway.
- Zuul sử dụng Eureka để tìm kiếm và xác định dịch vụ mục tiêu dựa trên tên dịch vụ.

- Sau đó, Zuul điều phối yêu cầu đến dịch vụ mục tiêu và chuyển tiếp kết quả trả về từ dịch vụ đó đến client.

4. Các chức năng mở rộng:

- Mô hình này có thể được mở rộng và tùy chỉnh theo nhu cầu cụ thể.
- Có thể áp dụng các chức năng như xác thực, ủy quyền, kiểm soát truy cập, chuyển tiếp mạng (network forwarding) và quản lý lưu lượng trong Zuul.
- Có thể sử dụng các chức năng mở rộng khác như Ribbon để cân bằng tải giữa các phiên bản của cùng một dịch vụ.



Hình 7: Thử nghiệm với các mã nguồn mở

4.2 Môi trường triển khai

Toàn bộ thử nghiệm được cài đặt dạng container trên nền tảng máy chủ VPS với các thông số cấu hình như sau:

Tên	**Thông số**
IP	183.81.33.100
RAM	2GB
Memory	30GB
CPU	2 vCore E5-2696v4 CPU
OS	Ubuntu 20.04

Địa chỉ cần đảm bảo an toàn:

http://183.81.33.100:8080/ex1/tutorials?user=user2

Địa chỉ public có thể nhúng thành iframe:

http://183.81.33.100:8050/ex1/tutorials?user=user2&key=aaaaa

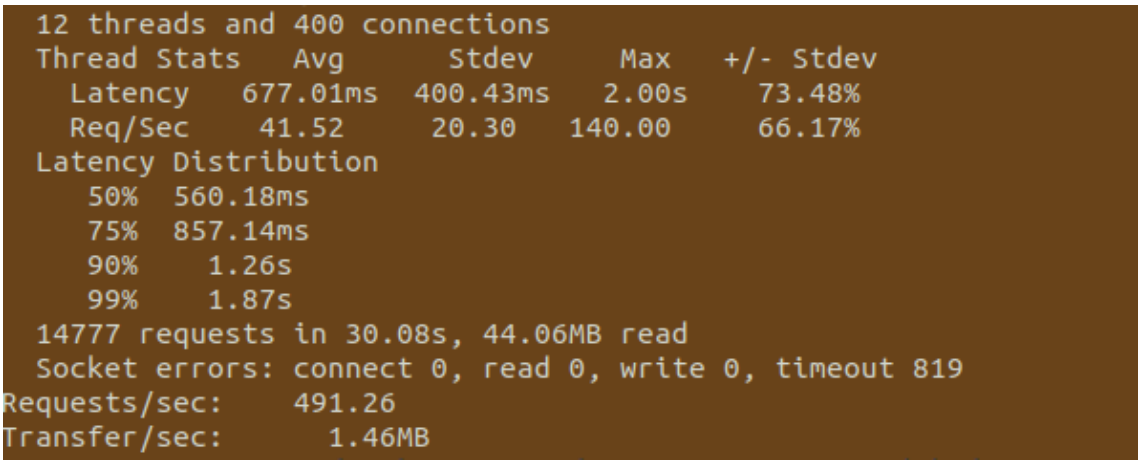
Server Zuul: http://183.81.33.100:8050

Server Eureka: http://183.81.33.100:8761

Dịch vụ xác thực: http://183.81.33.100:5555

4.3 Kết quả

Thử nghiệm với công cụ mã nguồn mở **wrk** đo tiêu chuẩn HTTP:



Hình 8: Thử nghiệm đánh giá hiệu năng của trang một web bởi wrk

So sánh hiệu năng của việc truy cập trực tiếp và truy cập thông qua API Gateway nhận được kết quả như sau:

Thông số đánh giá	**Địa chỉ trực tiếp:** http://183.81.33.100:8080/ex1	**Địa chỉ qua API Gateway:** http://183.81.33.100:8050/ex1
Time: 30s Thread: 12 HTTP_connection: 400	Latency: 1.26s Request/sec: 35.84 Transfer/sec: 143.55KB	Latency: 1.28s Request/sec: 34.00 Transfer/sec: 157.25KB
Time: 30s Thread: 12 HTTP_connection: 1000	Latency: 1.57s Request/sec: 47.30 Transfer/sec: 189.46KB	Latency: 1.57s Request/sec: 46.81 Transfer/sec: 182.30KB
Time: 30s Thread: 12 HTTP_connection: 2000	Latency: 1.62 Request/sec: 52.37 Transfer/sec: 209.79KB	Latency: 1.61 Request/sec: 51.66 Transfer/sec: 198.23KB

Kết quả thực nghiệm cho thấy việc xử lý request Gateway không gây ảnh hưởng xấu tới hiệu năng so với truy cập trực tiếp địa chỉ dịch vụ.

CHƯƠNG 5. KẾT LUẬN

5.1 Kết luận

Việc nghiên cứu về vấn đề an toàn cho iframe đã chỉ ra được các lỗ hổng bảo mật có thể mắc phải cùng với các cách ngăn ngừa chúng khi phát triển một phần mềm.

Mô hình đề xuất để nhúng iframe an toàn với thử nghiệm đã cho thấy sự hiệu quả của việc sử dụng Service Discovery kết hợp API Gateway làm nới điều phối trung tâm đảm bảo xác thực và phân quyền. Cùng với đó, các thông số đo đạc kết quả thử nghiệm cũng chỉ ra sự ổn định và hiệu năng tốt của mô hình.

Mặc dù vậy, các giải pháp vẫn cần được bổ sung và hoàn thiện bởi sự khác biệt của môi trường thử nghiệm so với môi trường triển khai thực tế.

5.2 Hướng phát triển của trong tương lai

Thời gian sắp tới, em sẽ hướng tới đưa ra thiết kế chi tiết cho việc nhúng an toàn thành iframe cho hai mã nguồn mở là MLFlow và Grafana như một dịch vụ quản lý, theo dõi, trực quan hóa cho phần mềm.

Ngoài ra, với xu hướng chuyển đổi các phần mềm sử dụng cloud, các thành phần của mô hình cũng có thể triển khai trực tiếp dạng cloud native giúp tăng tính linh hoạt, khả năng mở rộng và tốc độ phát triển của phần mềm.

- TÀI LIỆU THAM KHẢO

[1]	C. Jackson and A. Barth, "Beware of Finer-Grained Origins. In Web 2.0 Security and Privacy", W2SP 2008.
[2]	G. Rydstedt, E. Bursztein, D. Boneh, and C. Jackson, "Busting frame busting: a study of clickjacking vulnerabilities at popular sites", IEEE Oakland Web 2.0 Security and Privacy W2SP 2010.
[3]	Imran Yusof; Al-Sakib Khan Pathan, "Mitigating Cross-Site Scripting Attacks with a Content Security Policy ", 2017.
[4]	Hendrik Siewert, Martin Kretschmer, Marcus Niemietz, Juraj Somorovsky, "On the Security of Parsing Security-Relevant HTTP Headers in Modern Browsers ", 2022.
[5]	Some, D. F., Bielova, N., & Rezk, T., " Some, D. F., Bielova, N., & Rezk, “On the Content Security Policy Violations due to the Same-Origin Policy”, The 26th International Conference on World Wide Web," 2017.

- PHỤ LỤC

Mã nguồn và hướng dẫn cài đặt mô hình thử nghiệm:

https://github.com/mariorenger/iframe