# I. Injection

## 1. Khái niệm chung

* **Injection** là một lỗ hổng trong ứng dụng web, cho phép tin tặc thực hiện các câu lệnh hay truy vấn động bằng cách chèn các đoạn mã độc vào các trường dữ liệu trong ứng dụng web.
* Các đoạn mã độc thường được sử dụng thể thực hiện các hành động trái phép như truy cập vào các tài nguyên, thay đổi dữ liệu hoặc thực hiện các hoạt động độc hại khác.
* Có nhiều dạng chèn mã phổ biến như Buffer overflow, SQL Injection, Shell Command Injection, XPath/XQuery Injection, …

## 2. Khái niệm riêng

* **SQL Injection** là dạng chèn mã phổ biến, tin tặc sẽ thực hiện chèn các câu truy vấn SQL vào các trường dữ liệu của ứng dụng web.
* Lỗ hổng này thường xảy ra ở các ứng dụng web sử dụng câu lệnh SQL động (mã SQL gốc được kết hợp với dữ liệu đầu vào để tạo thành câu truy vấn) và không thực hiện kiểm tra đầu vào từ phía người dùng.
* Khi các câu lệnh này được thực hiện sẽ giúp tin tặc có thể vượt qua khâu xác thực, truy vấn các dữ liệu nhạy cảm, thêm, sửa hoặc xóa dữ liệu,..
* Nguy hiểm hơn là hacker có thể chiếm quyền điều khiển thông qua việc thực hiện các lời gọi hệ thống.

## 3. **Các dạng SQL Injection**

Diagram

Description automatically generated

### 3.1. In-band SQLi

* Là dạng tấn công mà hacker có thể tấn công và thu thập kết quả trực tiếp trên cùng một kênh liên lạc.
* Có 2 dạng tấn công chính
  + Error-based SQLi
  + Union-based SQLi

#### **Error-based SQLi**

* + Là dạng tấn công thực hiện chèn các giá trị đầu vào không hợp lệ, làm cho máy chủ CSDL trả về các thông báo lỗi.
  + Dựa vào những thông báo lỗi này có thể thu thập được các thông thông tin về CSDL như phiên bản CSDL, tên CSDL, tên bảng, cấu trúc của CSDL,….

#### **Union-based SQLi**

* + Là kỹ thuật sử dụng các toán tử UNION để kết hợp nhiều câu truy vấn và nhận được HTTP Response chứa các dữ liệu được trả về từ CSDL. Lỗi này được sử dụng để truy vấn dữ liệu từ nhiều bảng khác nhau trong CSDL
  + Câu truy vấn sử dụng UNION SELECTcần 2 điều kiện:
    - Số cột trong câu truy vấn bên vế trái của UNION SELECT phải bằng số cột của câu truy vấn bên vế phải.
    - Các cột ở 2 vế phải có kiểu dữ liệu tương ứng.

Dựa vào điều kiện này có thể thực hiện để đoán số cột có trong bảng.

### 3.2. Blind SQLi

* Là kỹ thuật mà hacker gửi các payload đến máy chủ và quan sát các phản hồi, hành vi của máy chủ để tìm hiểu, đoán về cấu trúc của nó.
* Loại chèn mã này không có dữ liệu truy vấn nào được trả về thông qua ứng dụng web và hacker không thể nhìn thấy được kết quả như tấn công In-band, chỉ nhìn được các thông báo chung chung.
* Có 2 dạng tấn công chính:
  + Boolean-based SQLi
  + Time-based SQLi

#### **Boolean-based SQLi**

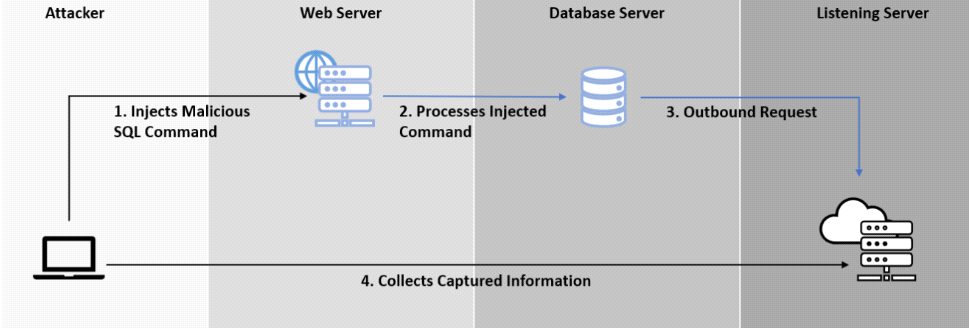
* + Là kỹ thuật tấn công chèn các payload làm ứng dụng trả về các kết quả khác nhau, phụ thuộc vào câu truy vấn là True hay False.
  + Tùy thuộc vào kết quả trả về của câu truy vấn mà HTTP Response có thể thay đổi hoặc giữ nguyên.
  + VD như khi tìm kiếm phim, ứng dụng chỉ trả về HTTP response là bộ phim có tồn tại hay không. Ứng dụng sẽ dựa vào số hàng trong kết quả của truy vấn, nếu bằng 0 thì trả về phim không tồn tại và ngược lại.
  + Kiểu tấn công này thường chậm và tốn nhiều thời gian do người tấn công cần phải liệt kê từng dữ liệu, thử từng trường hợp hoặc mò từng ký tự.

#### **Time-based SQLi**

* + Là kỹ thuật gửi các câu truy vấn tới CSDL, buộc CSDL phải chờ một khoảng thời gian (thường tính bằng giây) trước khi phản hồi về cho ứng dụng web.
  + Dựa vào thời gian CSDL phản hồi (ngay lập tức hoặc trễ theo khoảng thời gian được set), hacker có thể suy đoán được câu truy vấn đã thực hiện là đúng hay sai mà không cần dựa vào dữ liệu được trả về từ CSDL.
  + Kiểu tấn công này cũng tốn thời gian để thử từng trường hợp giống như Boolean-based SQLi.
  + Thường sử dụng mệnh đề điều kiện IF để chèn mã. Nếu mệnh đề IF được thỏa mãn thì sẽ thực hiện lệnh SLEEP.
  + Nếu truy vấn thực sự đúng thì CSDL sẽ thực hiện câu truy vấn và sleep trong thời gian được set. Và phía trình duyệt sẽ ở trạng thái loading trong thời gian đang sleep đó.
  + Nếu truy vấn sai thì sẽ không sleep theo thời gian. Và phía trình duyệt sẽ load rất nhanh.

### 3.3. Out-of-band SQLi

* Là dạng tấn công truyền vào payload khiến cho databaseserver xử lý, tạo request ngoài băng tần tới 1 máy mà hacker kiểm soát. Hacker không nhận được phản hồi từ ứng dụng bị tấn công trên cùng một kênh liên lạc mà thay vào đó có thể khiến ứng dụng gửi dữ liệu đến một máy tính từ xa mà chúng kiểm soát.
* Kiểu tấn công này phụ thuộc vào khả năng máy chủ thực hiện các request DNS hoặc HTTP để chuyển dữ liệu cho hacker.
* Kịch bản:



* B1: Kẻ tấn công gửi sql độc hại đến web server
* B2: Web server truy vấn tới DB server
* B3: DB server gửi DNS outbound request đến server mà kẻ tấn công kiểm soát
* B4: Kẻ tấn công lấy dữ liệu qua DNS

### 3.4. Second-order Injection (Stored SQLi)

* Còn được gọi là Stored SQL Injection là phương pháp chèn vào cơ sở dữ liệu một đoạn mã SQL mà chưa được thực thi ngay nên chưa gây nguy hiểm cho hệ thống
* Khi ứng dụng xử lý một HTTP request khác (một giao diện thứ 2) mà cần sử dụng đến dữ liệu này, nó sẽ truy vấn đến dữ liệu đó (đoạn mã đã được lưu trong CSDL) và đoạn mã đó sẽ được thực thi
* VD:

Diagram

Description automatically generated

## 4. Các biện pháp phòng tránh

* Cần thực hiện, sử dụng các phương pháp lọc, kiểm tra dữ liệu được nhập vào từ phía người dùng, hạn chế nhập các ký tự đặc biệt (như \*, ‘, =, --) và các từ khóa của ngôn ngữ SQL (SELECT, INSERT, UPDATE, DELETE, DROP,....)
* Sử dụng các thủ tục lưu trữ trong CSDL và cơ chế tham số hóa dữ liệu. Dữ liệu truyền vào thủ tục thông qua các tham số, giúp tách dữ liệu khỏi mã lệnh SQL
* Cấm hoặc vô hiệu hóa việc thực hiện các thủ tục hệ thống (các thủ tục cơ sở dữ liệu có sẵn) cho phép can thiệp vào hệ quản trị cơ sở dữ liệu và hệ điều hành nền.
* Xử lý ngoại lệ, các lỗi truy vấn trả về các thông báo không chi tiết để hacker không thu thập được thông tin qua thông báo lỗi.
* Sử dụng công cụ rà quét như SQLMap, hoặc Acunetix Vulnerability Scanner để chủ động rà quét, tìm các lỗ hổng chèn mã SQL và khắc phục kịp thời.

# II. Broken Authentication

## 1. Khái niệm

- Là lỗ hổng bảo mật xảy ra khi các thủ tục xác thực và quản lý phiên không an toàn dẫn đến việc hacker lấy được quyền truy cập vào các tài khoản người dùng hoặc thậm chí lấy được quyền quản trị hệ thống.

- Hậu quả:

* Hacker có thể truy cập vào 1 tài khoản của người dùng hoặc admin của hệ thống
* Hoặc có thể thực hiện thao tác khác tùy theo đặc quyền của tài khoản mà có thể gây nguy hiểm cho hệ thống.

## 2. Các nguyên nhân phổ biến

* Sử dụng mật khẩu yếu, dễ đoán
* Mật khẩu không được mã hóa hoặc dễ bị giải mã trong khi lưu trữ
* Thông báo lỗi xác thực quá chi tiết (VD báo lỗi không đúng mật khẩu nhưng tên người dùng thì đã đúng)
* Không sử dụng SSL để mã hóa trên đường truyền
* Sử dụng các session ID dễ đoán, có thời gian sống lâu hoặc xác thực phiên yếu
* Không kết thúc phiên đăng nhập đầy đủ khi người dùng đăng xuất.

## 3. Các dạng tấn công

* **Password-based attack:** tấn công bằng cách dự đoán mật khẩu hoặc sử dụng các công cụ như metasploit, hydra, burp intruder,… để vét cạn, thử tất cả các tài khoản và mật khẩu có thể.

VD: Các trang web không cấu hình ràng buộc chính sách mật khẩu mạnh, cho phép người dùng sử dụng mật khẩu yếu, không giới hạn số lần đăng nhập sai, hacker có thể tấn công vét cạn mật khẩu.

-> hacker thử vét vật khẩu dựa trên wordlist.

* **Session hijacking:** sử dụng các kỹ thuật trung gian như tấn công XSS để lấy được session ID chiếm phiên đăng nhập của người dùng và sử dụng nó để truy cập vào các tài khoản hoặc thực hiện các hoạt động không mong muốn.
* **Man in the Middle:** nghe trộm, chặn bắt các tham số xác thực trên đường truyền không được mã hóa.

## 4. Các biện pháp phòng tránh

* Sử dụng các ràng buộc về mật khẩu mạnh, khó đoán và yêu cầu người dùng thay đổi mật khẩu định kỳ.
* Sử dụng Session ID mạnh, ngẫu nhiên và có thời gian sống ngắn.
* Sử dụng xác thực đa yếu tố.
* Chấm dứt phiên đăng nhập đầy đủ khi người dùng đăng xuất.

# III. Sensitive Data Exposure

## 1. Khái niệm

- Là loại lỗ hổng mà một trang web vô tình tiết lộ, làm rò rỉ các thông tin quan trọng, nhạy cảm như mật khẩu, thông tin thẻ tín dụng, thông tin cá nhân của người dùng, … được lưu trữ hoặc truyền không an toàn trên đường truyền

- Đối với cá nhân với cá nhân, các thông tin nhạy cảm gồm có số điện thoại, email, CCCD, thẻ ngân hàng, địa chỉ nhà riêng, thông tin gia đình,…

- Đối với tổ chức, các thông tin nhạy cảm gồm thông cá nhân của các nhân viên, thông tin hợp đồng của công ty, IP của hệ thống nội bộ, thông tin khách hàng, sơ đồ mạng trong tổ chức,…

- Với 1 website thì những thông tin nhạy cảm là phiên bản hệ thống của trang web đang chạy, công nghệ web sử dụng, source code, thông tin đăng nhập quản trị, CSDL,...

- Tin tặc có thể đánh cắp và sử dụng thông tin, dữ liệu này để phục vụ cho việc nghiên cứu và tấn công vào các lỗ hổng khác.

## 2. Các dạng, hình thức rò rỉ

* Rò rỉ tên của các thư mục ẩn, cấu trúc và nội dung của các thư mục thông qua tệp robots.txt
* Lưu trữ thông tin nhạy cảm như mật khẩu dưới dạng thô, không được mã hóa
* Sử dụng các thuật toán mã hóa yếu, tin tặc có thể dễ dàng giải mã và truy cập các thông tin này như mã hóa base64.
* Dữ liệu, các thông tin nhạy cảm được truyền tải qua các kênh không an toàn, không được mã hóa như sử dụng giao thức HTTP thay vì HTTPS.
* Lưu trữ thông tin nhạy cảm lâu dài trong hệ thống.
* Thông tin đăng nhập lưu trong mã html để test nhưng chưa xóa đi.
* Không bảo vệ các bộ nhớ đệm (cache) đúng cách khiến các thông tin nhạy cảm được lưu trữ trong bộ nhớ đệm bị đánh cắp.

## 3. Các biện pháp phòng tránh

* Sử dụng thuật toán mã hóa mạnh, lưu trữ các thông tin nhạy cảm dưới dạng mã hóa
* Sử dụng giao thức HTTPS để truyền tải dữ liệu giữa máy khách và máy chủ.
* Bảo vệ bộ nhớ cache, không cho phép lưu trữ dữ liệu nhạy cảm trong cache.

# IV. XML External Entities (XXE)

## 1. Khái niệm

- **XXE** hay tấn công thực thể bên ngoài XML là là lỗ hổng lợi dụng tính năng phân tích cú pháp của XML dùng để phân tích cú pháp đầu vào XML từ người dùng, cho phép kẻ tấn công xem các tệp trên hệ thống tệp của máy chủ ứng dụng hoặc tương tác với hệ thống back-end hoặc hoặc hệ thống bên ngoài nào mà ứng dụng có thể truy cập.

- Ngoài ra, kẻ tấn công có thể thực hiện các cuộc tấn công giả mạo yêu cầu phía máy chủ (SSRF).

## 2. Nguyên nhân

* Sử dụng các thư viện XML không an toàn hoặc không được cập nhật
* Không kiểm tra đầu vào XML đúng cách, cho phép tin tặc chèn các thực thể bên ngoài
* Không sử dụng các cơ chế bảo vệ như sandbox để giới hạn quyền truy cập của ứng dụng web đến các tài nguyên trên hệ thống.

## 3. Các loại tấn công XXE

* Khai thác XXE để truy xuất tệp
  + Một thực thể bên ngoài được định nghĩa có chứa nội dung của tệp và được trả về trong phản hồi của ứng dụng.
  + Hacker có thể sửa đổi XML theo 2 cách:
    - Sửa phần tử DOCTYPE, khai báo một thực thể bên ngoài (external entity) chứa đường dẫn đến tệp
    - Chỉnh sửa giá trị của dữ liệu trong XML được trả về trong phản hồi của ứng dụng, để sử dụng thực thể bên ngoài đã định nghĩa
* VD: định nghĩa, khai báo một thực thể bên ngoài &xxe có giá trị là nội dung của file /etc/passwd. Sử dụng thực thể bên trong thẻ <productId>

Graphical user interface, text, application, email

Description automatically generated

ứng dụng sẽ trả về phản hồi chứa nội dung của file.

Text

Description automatically generated

* Phân tích cú pháp XXE:

Text

Description automatically generated

* DOCTYPE được sử dụng để định nghĩa kiểu tài liệu của tệp XML (ở đây được định nghĩa kiểu foo).
* foo là tên kiểu tài liệu được định nghĩa và là tên của phần tử gốc (thẻ <foo> </foo>). Trong phần tử gốc foo sẽ bao gồm các phần tử hoặc thuộc tính khác.
* Cú pháp để định nghĩa 1 thực thể bên ngoài là

Graphical user interface, text, application

Description automatically generated

+ ENTITY được sử dụng để khai báo một thực thể sẽ được tái sử dụng trong tài liệu.

+ name là tên của thực thể

+ SYSTEM để chỉ định rằng nội dung của thực thể được lấy từ một hệ thống bên ngoài. Không thể bỏ hoặc thay thế từ khóa SYSTEM bằng từ khác vì đây là tiêu chuẩn để khai báo thực thể bên ngoài XML. Nếu bỏ SYSTEM đi thì XML parser sẽ hiểu đây là thực thể bên trong Internal Entity.

+ Từ khóa "PUBLIC": khi cùng một thực thể được sử dụng trong nhiều tài liệu hoặc khi thực thể được chia sẻ bởi nhiều ứng dụng.

Dưới đây là một ví dụ về khai báo DTD ENTITY với mã định danh PUBLIC:

**<!ENTITY entityName PUBLIC "publicIdentifier" "systemIdentifier">**

"**entityName**" là tên của thực thể được xác định, "**publicIdentifier**" là mã định danh công khai cho thực thể "**systemIdentifier**" là mã định danh hệ thống chỉ định vị trí định nghĩa của thực thể.

Ví dụ:

**<!ENTITY logo PUBLIC "-//Example//DTD Example1.0//EN" "http://www.example.com/logo.gif">**

Điều này sẽ cho phép thực thể "logo" được tham chiếu theo tên trong bất kỳ tài liệu XML nào bao gồm DTD này và mãđịnh danh công khai sẽ cho phép thực thể được xác định và truy cập duy nhất từ ​​bên ngoài tài liệu.

Nội dung của logo sẽ được lấy từ một tài liệu công khai xác định là **"-//Example//DTD Example 1.0//EN"** nếu XML parser không có sẵn bản sao của tài liệu này thì sẽ sử dụng **"http://www.example.com/logo.gif"**để tìm kiếm.

+ URL là đường dẫn tới nơi chứa nội dung của thực thể (tới file “/etc/paswd” trên máy chủ web)

* Thực thể có tên xxe được gọi ở trong thẻ productId và hiển thị nội dung của thực thể
* Khai thác XXE để thực hiện tấn công SSRF
  + Là loại tấn công mà ứng dụng phía máy chủ có thể thực hiện các HTTP request tới bất kỳ URL nào mà máy chủ có thể truy cập.
  + Hacker định nghĩa một thực thể XML bên ngoài với URL mà hacker muốn khai thác tấn công và sử dụng thực thể đã định nghĩa trong giá trị của dữ liệu.
  + Nếu có thể sử dụng thực thể đã định nghĩa trong giá trị của dữ liệu được trả về trong response của ứng dụng thì hacker sẽ có thể truy cập đến các tài nguyên trên máy chủ đó, tương tác hai chiều với hệ thống back-end.
  + VD: Định nghĩa một xxe có nội dung nằm trong URL đến máy chủ web, có thể truy cập lần lượt từng thư mục trên máy chủ web đó cho đến khi đọc được nội dung cùa file admin

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

* Khai thác Blind XXE lọc dữ liệu out-of-band
  + Dữ liệu nhạy cảm được truyền từ máy chủ ứng dụng đến hệ thống mà kẻ tấn công kiểm soát.
  + Kiểu tấn công này khiến máy chủ thực hiện request HTTP back-end tới URL đã chỉ định và kẻ tấn công có thể theo dõi kết quả DNS lookup và request HTTP qua hệ thống mà chúng kiểm soát.
* Khai thác Blind XXE để truy xuất dữ liệu qua thông báo lỗi
  + Hacker có thể chèn các thực thể XML bên ngoài mà gây ra các lỗi phân tích cú pháp tại trình phân tích XML và ứng dụng sẽ trả về các thông báo lỗi có chứa dữ liệu nhạy cảm.

## 4. Các biện pháp phòng tránh

* Sử dụng các thư viện XML an toàn và được cập nhật thường xuyên
* Kiểm tra, lọc đầu vào XML để chặn các ký tự đặc biệt hoặc các thực thể bên ngoài có thể được chèn vào tài liệu XML.
* Vô hiệu hóa các tính năng của thư viện phân tích XML không cần thiết.
* Sử dụng các định dạng dữ liệu ít phức tạp hơn như JSON và tránh tuần tự hóa dữ liệu nhạy cảm.

# V. Broken Access Control

## 1. Khái niệm

- Là lỗ hổng cho phép tin tặc có thể truy cập và thực hiện các hoạt động, thao tác bên trong hệ thống mà không được phép (vượt quá quyền được cấp).

- Kiểm soát truy cập được chia thành 3 loại:

* ***Kiểm soát truy cập theo chiều dọc (Vertical access controls)***
  + Là cơ chế hạn chế quyền truy cập của người dùng có quyền thấp vào các chức năng nhạy cảm của người dùng có quyền cao hơn.
  + VD: admin có thể sửa đổi hoặc xóa tài khoản của bất kỳ người dùng nào, trong khi user bình thường không có quyền thực hiện.
* ***Kiểm soát truy cập theo chiều ngang (Horizontal access control)***
  + Là cơ chế hạn chế quyền truy cập của người dùng vào tài nguyên, chức năng của những người dùng khác có cùng role, quyền.
  + VD: ứng dụng ngân hàng chỉ cho phép người dùng xem các giao dịch và thực hiện thanh toán từ tài khoản của chính họ chứ không phải tài khoản của bất kỳ người dùng nào khác.
* ***Kiểm soát truy cập phụ thuộc vào ngữ cảnh***
  + Kiểm soát truy cập phụ thuộc vào ngữ cảnh hạn chế quyền truy cập vào chức năng và tài nguyên dựa trên trạng thái của ứng dụng hoặc tương tác của người dùng với nó.
  + Kiểm soát truy cập phụ thuộc vào ngữ cảnh giúp ngăn người dùng thực hiện các hành động sai thứ tự.
  + VD: một trang web bán lẻ không cho phép người dùng sửa đổi số lượng sản phẩm trong giỏ hàng của họ sau khi họ đã thanh toán.

## 2. Các dạng lỗ hổng kiểm soát truy cập

- Leo thang đặc quyền theo chiều dọc

* Là việc người dùng có đặc quyền (role) thấp nhưng có thể truy cập vào chức năng của người dùng có quyền cao hơn.
* Ví dụ: một người dùng bình thường có thể truy cập vào trang quản trị của admin và thực hiện xóa tài khoản người dùng.
* Nguyên nhân
* Chức năng không được bảo vệ
* Phương pháp kiểm soát truy cập dựa trên tham số
* Kiểm soát truy cập bị hỏng do cấu hình sai nền tảng
* Kiểm soát truy cập bị hỏng do sự khác biệt về kết hợp URL

- Leo thang đặc quyền theo chiều ngang

* Phát sinh khi người dùng có thể có quyền truy cập vào tài nguyên thuộc về người dùng khác, thay vì tài nguyên thuộc loại đó của chính họ.
* Ví dụ: một nhân viên có thể truy cập hồ sơ, xem bảng lương của các nhân viên khác thay vì chỉ xem của chính họ.

- Leo thang đặc quyền theo chiều ngang sang chiều dọc

* Một cuộc tấn công leo thang đặc quyền theo chiều ngang có thể biến thành một cuộc tấn công leo thang đặc quyền theo chiều dọc bằng cách xâm phạm một người dùng có nhiều đặc quyền hơn.
* Ví dụ: leo thang theo chiều ngang có thể cho phép kẻ tấn công đặt lại hoặc chiếm được mật khẩu của người dùng khác. Nếu kẻ tấn công nhắm mục tiêu một người dùng quản trị và xâm phạm tài khoản của họ, thì họ có thể có quyền truy cập quản trị và do đó thực hiện leo thang đặc quyền theo chiều dọc.
* Nguyên nhân
  + Tham chiếu đối tượng trực tiếp không an toàn (Insecure DOR)
  + Lỗ hổng kiểm soát truy cập trong quy trình nhiều bước
  + Kiểm soát truy cập dựa trên người giới thiệu
  + Kiểm soát truy cập dựa trên vị trí
* Tham chiếu đối tượng trực tiếp không an toàn (Insecure Direct Object Reference)
  + Lỗ hổng này xảy ra khi chương trình cho phép người dùng truy cập tài nguyên (dữ liệu, file, thư mục, database..) một cách trực tiếp thông qua dữ liệu do người dùng cung cấp nhưng kém an toàn.
  + Cho phép attacker có thể bypass việc phân quyền và truy cập trực tiếp tới các tài nguyên không thuộc thẩm quyền bằng cách sửa đổi tham số mà được dùng để tham chiếu đến đối tượng.
  + Attacker có thể xem được các thông tin nhạy cảm của các người dùng khác, thực thi các chức năng mà bình thường không được phép, thậm chí có thể chiếm đoạt tài khoản khác.
  + VD:
    - Lỗ hổng IDOR với tham chiếu trực tiếp đến các đối tượng cơ sở dữ liệu

+ Ví dụ truy cập tài khoản khách hàng thông qua tham số ID thì hacker có thể sửa ID để vào tài khoản của khách hàng khác => IDOR leo thang đặc quyền theo chiều ngang.

* + - Lỗ hổng IDOR với tham chiếu trực tiếp đến các tệp tĩnh

+ Phát sinh khi tài nguyên nhạy cảm được đặt trong các tệp tĩnh.

+ Ví dụ: một trang web có thể lưu bản chép lại tin nhắn trò chuyện vào đĩa bằng cách sử dụng tên tệp tăng dần và cho phép người dùng truy xuất những bản ghi này bằng cách truy cập một URL. Kẻ tấn công có thể chỉ cần sửa đổi tên tệp để truy xuất bản ghi do người dùng khác tạo và có khả năng lấy thông tin đăng nhập của người dùng cũng như dữ liệu nhạy cảm khác.

## 3. Ví dụ

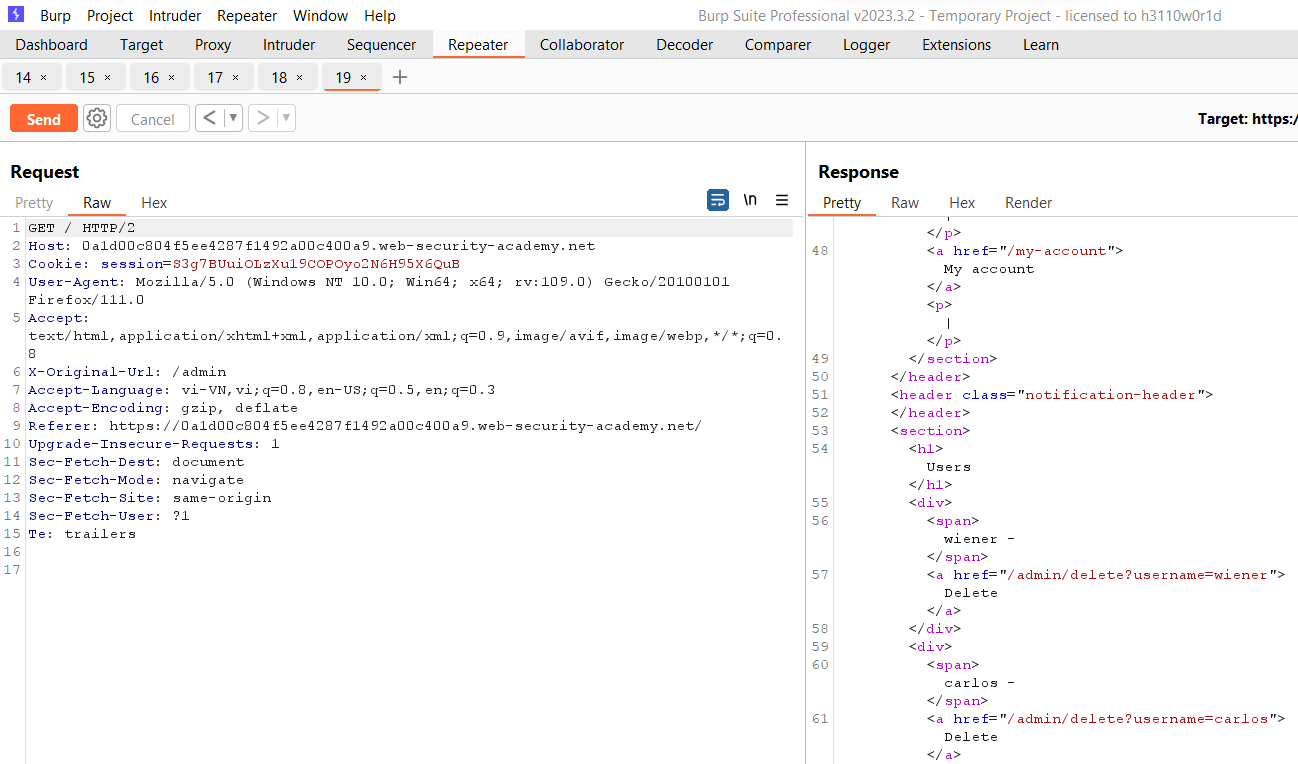
- Ứng dụng sử dụng javascript để kiểm tra role admin trực tiếp trên trang web (không tách riêng ra file script khác) -> để lộ ra đường dẫn đến trang quản trị của admin qua mã nguồn.

Graphical user interface, text, application

Description automatically generated

- Một số ứng dụng hỗ trợ các header không chuẩn như X-Original-URL hoặc X-Rewrite-URL để cho phép ghi đè URL mục tiêu trong các yêu cầu với URL được chỉ định trong giá trị header.

- Đường dẫn đến trang /admin không được truy cập từ URL được nhập trên Request Line nhưng khi truyền /admin qua header X-Original-URL thì phía backend vẫn xử lý, ghi đè lên URL trên Request Line và được phép truy cập.



## 4. Các biện pháp phòng tránh

* Ngoại trừ các tài nguyên được công khai thì đều từ chối truy cập với các tài nguyên còn lại.
* Giảm thiểu việc sử dụng chia sẻ tài nguyên nguồn gốc chéo (CORS).
* Vô hiệu hóa danh sách thư mục của máy chủ web đảm bảo các tệp siêu dữ liệu (.git) và tệp sao lưu không có trong thư mục gốc của web.
* Ghi nhật ký (log) các lỗi kiểm soát truy cập và cảnh báo cho quản trị viên
* Các Session ID có trạng thái nên bị vô hiệu hóa trên máy chủ sau khi đăng xuất.
* Không thực hiện kiểm soát truy cập thông qua các tham số được truyền vào từ người dùng .

# VI. Security Misconfigurations

## 1. Khái niệm

- Là lỗ hổng do các thiết lập bảo mật trong ứng dụng web không được thiết lập hoặc cấu hình đúng cách hoặc bị thiết sót dẫn đến các lỗ hổng bảo mật có thể bị khai thác bởi kẻ tấn công

- VD như:

* HTTP header cấu hình sai, máy chủ không gửi các header bảo mật hoặc không được set giá trị
* Thông báo lỗi hiển thị có quá nhiều thông tin nhạy cảm như mô tả quá chi tiết về lỗi
* Không cập nhật bản cập nhật, bản vá từ framework, các thư viện, phần mềm
* Giữ nguyên các cấu hình mặc định ban đầu của framework, thư viện
* Sử dụng các tài khoản, mật khẩu mặc định

## 2. Các nguyên nhân

* Thiếu kiểm tra cấu hình.
* Thiếu quản lý cập nhật để đảm bảo rằng các cài đặt bảo mật được cập nhật đầy đủ và kịp thời.
* Các nhà phát triển không có đủ hiểu biết và kiến thức về bảo mật để cấu hình và quản lý bảo mật.

## 3. Các biện pháp phòng tránh

* Thường xuyên cập nhật bản vá lỗi từ framework, thư viện, phần mềm
* Vô hiệu hóa các chức năng không thực sự cần thiết.
* Thực hiện kiểm thử, đánh giá bảo mật trong suốt quá trình phát triển ứng dụng
* Tăng cường hiểu biết về bảo mật của các nhà phát triển phần mềm

# VII. Cross Site Scripting

## 1. Khái niệm

- Là lỗ hổng cho phép hacker thực thi mã Javascript độc hại phía trình duyệt của người dùng thông qua việc chèn mã độc vào các trang web hoặc các trường input trên trang web để có thể phá vỡ chính sách same origin (chính sách cùng nguồn gốc tách biệt các trang web với nhau) và có thể truy cập vào dữ liệu của người dùng.

Diagram

Description automatically generated

## 2. Các loại tấn công XSS

- Có 3 loại tấn công XSS chính:

* Stored XSS
* Reflected XSS
* DOM-based XSS

### 2.1. Reflected XSS

* + Là dạng tấn công chèn script độc hại vào HTTP request (thông qua URL hoặc form trên trang web) và script độc hại sẽ được phản hồi trực tiếp trong phản hồi từ máy chủ.
  + Khi người dùng truy cập vào liên kết hoặc gửi form, mã độc sẽ được thực thi trong trình duyệt của họ.
  + Thường xuất hiện khi dữ liệu từ người dùng cung cấp (qua URL hoặc form) được bên phía mảy chủ sử dụng để tạo ra response và gửi lại kết quả cho người dùng (lấy dữ liệu từ URL để gắn vào trả về 1 trang HTML)

Diagram

Description automatically generated

* + Mô tả các bước:
    - B1: Người dùng đăng nhập vào hệ thống và được cấp Session ID được lưu trong Cookie.
    - B2: Kẻ tấn công bằng cái nào đấy gửi URL khai thác tới cho người dùng. Trong URL có chứa 1 parameter có giá trị là 1 đoạn JavaScript để thực hiện khai thác.
    - B3: Người dùng truy cập vào URL khai thác được gửi từ attacker.
    - B4: Phía server phản hồi lại cho người dùng response có chứa đoạn script (đoạn script được lấy từ parameter là đính kèm vào response body).
    - B5: Trình duyệt nhận được response và thực thi đoạn script để hiển thị cho người dùng.
    - B6: Đoạn script lấy được cookie của người dùng và gửi về cho attacker thông qua request tới tên miền mà hacker sở hữu.
    - B7: Kẻ tấn công theo dõi các request tới tên miền và nhận được request từ người dùng có chứa cookie, chúng sử dụng cookie vừa có được để chiếm phiên của người dùng

### 2.2. Stored XSS

* + Phát sinh khi một ứng dụng nhận dữ liệu từ một nguồn không đáng tin cậy và trả về dữ liệu (script độc hại) đó trong các HTTP response sau này theo một cách không an toàn.
  + Mã script thường được chèn vào các trường nhập liệu như bình luận, như bình luận, thư viện hình ảnh, hoặc hộp thoại.
  + Các đoạn mã script này được lưu trữ vĩnh viễn trong CSDL của website.
  + Lỗ hổng này chỉ ảnh hưởng tới người dùng đã login
  + Khi người dùng truy cập vào trang web bị nhiễm, trang web sẽ thực hiện truy vấn từ CSDL và mã độc sẽ được thực thi trên trình duyệt của người dùng.

Diagram

Description automatically generated

* + Mô tả các bước:
    - B1: Hacker gửi nộp dung có chứa đoạn Javascript với ứng dụng web (thông qua tính năng comment) và phía máy chủ web sẽ lưu mã này vào CSDL
    - B2: Người dùng đăng nhập vào ứng dụng (nếu đã đăng nhập từ trước thì vẫn có sẵn session id được lưu trong cookie)
    - B3: Người dùng truy cập vào đường dẫn mà đối tượng được trả về có chứa mã JS (VD một bài báo đã được comment)
    - B4: Máy chủ web sẽ trả về phản hồi có kèm theo mã JS.
    - B5: Trình duyệt hiển thị response cho người dùng đồng thời thực thi đoạn mã độc, đánh cắp được cookie của người dùng
    - B6: Session được gửi về cho hacker thông qua request tới tên miền hacker sở hữu
    - B7: Hacker sử dụng cookie để chiếm phiên của người dùng.

-> Nhận xét: Stored XSS có mức độ nguy hiểm cao hơn Reflected XSS vì Stored XSS có thể khai thác nhiều người dùng cùng lúc, chỉ cần người dùng đó truy cập vào đối tượng có chứa mã JS đã được lưu trữ. Còn Reflected XSS thì cần phải lừa người dùng truy cập vào URL độc hại nên khó khăn hơn và mức độ nguy hiểm không cao bằng.

### 2.3. DOM-based/local XSS

* + DOM là biểu diễn phân cấp của trình duyệt web về các phần tử trên trang. Các trang web có thể sử dụng JavaScript để thao tác các nút và đối tượng của DOM, cũng như các thuộc tính của chúng.
  + Các lỗ hổng dựa trên DOM phát sinh khi một trang web chứa JavaScript lấy một giá trị mà kẻ tấn công có thể kiểm soát, được gọi là nguồn (source) và chuyển nó vào một hàm nguy hiểm, được gọi là phần chìm (sink). Lỗ hổng bảo mật này tồn tại phía client.
  + Taint flow (luồng dữ liệu):
    - Sources (nguồn) là thuộc tính JavaScript mà chấp nhận dữ liệu có khả năng bị kẻ tấn công kiểm soát. VD thuộc tính location.search đọc đầu vào từ chuỗi truy vấn. Bất kỳ thuộc tính nào có thể được kiểm soát bởi kẻ tấn công đều là một nguồn tiềm năng. Điều này bao gồm referring URL (document.referrer), cookie của người dùng (document.cookie) và web messages.
    - Sink là một hàm JavaScript hoặc đối tượng DOM nguy hiểm tiềm ẩn có thể gây ra các tác động không mong muốn nếu dữ liệu do kẻ tấn công kiểm soát được chuyển đến nó. Ví dụ: hàm eval() là hàm chìm vì nó xử lý đối số được truyền cho nó dưới dạng JavaScript. Ví dụ : HTML sink (hàm chìm HTML) là document.body.innerHTML vì nó có khả năng cho phép kẻ tấn công đưa HTML độc hại và thực thi JavaScript tùy ý.
  + Lỗ hổng DOM-based phát sinh khi một trang web chuyển dữ liệu từ sources sang sink, sau đó sink xử lý dữ liệu theo cách không an toàn trong phiên của máy khách.
  + Nguồn phổ biến nhất là URL, thường được truy cập bằng đối tượng window.location. Kẻ tấn công có thể tạo một liên kết để đưa nạn nhân đến một trang dễ bị tấn công bằng một payload trong query tring hoặc fragment trên URL
  + VD:

Ứng dụng sử dụng script để lấy giá trị của tham số search

Graphical user interface, text, application, email

Description automatically generated

Nhập “> <script>alert(1)</script> trong đó “> để đóng thẻ img và chèn thêm thẻ script để thực thi.

Graphical user interface, text, application

Description automatically generated

* + DOM based khác với Reflected là phía máy chủ chỉ gửi lại HTML tĩnh, không lấy các tham số trên URL để gắn vào HTML. Xử lý sẽ do script phía máy khách lấy các tham số từ source (URL) và chuyển đến sink để xử lý, thực thi.

## 3. Cách tìm và test XSS

- Test với Reflected và Stored XSS:

* Nhập đầu vào đơn giản vào mọi trường nhập liệu trong ứng dụng, xác định mọi vị trí mà đầu vào được trả về trong HTTP response và kiểm tra từng vị trí xem có chỗ nào phù hợp để thực thi Javascript hay không.

- Test với DOM-based XSS:

* Phát sinh từ URL parameter: Nhập vào các tham số đơn giản qua URL, tìm kiếm đầu vào trong DOM và kiểm tra từng vị trí xem có khai thác được không.

## 4. Các biện pháp phòng tránh

Diagram

Description automatically generated

* Sử dụng các bộ lọc XSS để lọc bỏ các thẻ HTML, Script khỏi dữ liệu nhập từ người dùng.
* Sử dụng các biểu thức chính quy (Regular Expressions) để tăng hiệu quả của bộ lọc
* XSS Escape: thay thế các ký tự riêng của HTML/scripts để chuyển các đoạn mã có thể thực hiện thành dữ liệu thông thường. VD: ký tự mở thẻ HTML < được chuyển thành &#60.
* HTML ENTITY
  + Các trường input được nhận vào từ phía người dùng nếu có chứa các thực thể HTML thì sẽ được HTML encode
  + VD như dấu ‘<’ được mã hóa thành **&lt;** hoặc **&#60;**
* Sử dụng các response header thích hợp: để ngăn XSS trong các phản hồi HTTP không có ý định chứa bất kỳ HTML hoặc JavaScript nào, có thể sử dụng các header Content-Type và X-Content-Type-Options để đảm bảo rằng các trình duyệt sẽ thông dịch các response theo cách mình muốn.
* Cách hiệu quả nhất để tránh các lỗ hổng dựa trên DOM là tránh cho phép dữ liệu từ bất kỳ nguồn không đáng tin cậy nào tự động thay đổi giá trị được truyền tới bất kỳ phần chìm nào.
* Nếu muốn sử dụng dữ liệu từ các nguồn thì cần triển phải biện pháp bảo vệ như xác thực các dữ liệu đó dựa trên whitelist chỉ cho phép các dữ liệu được xác định là an toàn.

# VIII. Insecure Deserialization

## 1. Khái niệm

- Tuần tự hóa là quá trình chuyển cấu trúc dữ liệu phúc tạp như đối tượng, các trường của chúng thành chuỗi byte tuần tự.

- Giải tuần tự hóa là quá trình chuyển đổi chuỗi byte thành đối tượng ban đầu.

Diagram

Description automatically generated

- Insecure deserialization: Cho phép kẻ tấn công lợi dụng các đối tượng được tuần tự hóa để chuyển dữ liệu có hại vào mã ứng dụng. Thậm chí có thể thay thế một đối tượng được tuần tự hóa bằng một đối tượng của một lớp hoàn toàn khác. Một đối tượng được tuần tự hóa bằng một đối tượng của một lớp khác mà các đối tượng thuộc bất kỳ lớp nào có sẵn trong trang web đều sẽ được giải tuần tự hóa và khởi tạo.

- Giải tuần tự hóa không an toàn có thể dẫn đến khả năng thực hiện mã từ xa hoặc tấn công phát lại, chèn mã, hoặc leo thang đặc quyền.

## 2. Các loại tấn công

- Có 2 loại tấn công chính:

* Tấn công thực thi mã từ xa: kẻ tấn công chèn mã độc vào chuỗi hoặc byte để sửa đổi logic của ứng dụng hoặc thực thi mã từ xa tùy ý nếu ứng dụng có sẵn các lớp có thể thay đổi hành vi trong hoặc sau quá trình giải tuần tự hóa.
* Tấn công giả mạo dữ liệu, thay đổi dữ liệu có trong chuỗi hoặc byte gây ảnh hưởng đến hoạt động của hệ thống.
  + VD: sử dụng tuần tự hóa đối tượng để chuyển thông tin về phiên của người dùng (gồm ID, tên, mật khẩu dạng băm,…) thành 1 chuỗi

Graphical user interface, text

Description automatically generated

Hacker sẽ thực hiện sửa đổi tên người dùng, quyền của người dùng ở chuỗi trên thành Alice và admin.

Graphical user interface, text

Description automatically generated

Khi ứng dụng web giải tuần tự hóa chuyển chuỗi thành đối tượng phiên thì hacker có thể chiếm quyền người quản trị.

* + VD: Ứng dụng sử dụng đối tượng User được tuần tự hóa để lưu trữ dữ liệu về phiên của người dùng trong cookie. Đối tượng sau khi tuần tự hóa được mã hóa base 64 và URL-> decode URL và base 64, thấy giá trị của tham số admin:b:0

Graphical user interface, text, application, email

Description automatically generated

Sửa thành admin:b:1 mà thực hiện mã hóa lại bằng base 64 và URL

Graphical user interface, text, application, email

Description automatically generated

Đã có thể truy cập được vào đường dẫn của admin và thực hiện chức năng xóa người dùng (leo thang đặc quyền theo chiều dọc).

Graphical user interface, text, application, email

Description automatically generated

## 3. Cách phát hiện lỗ hổng

- Có thể xác định lỗ hổng giải tuần tự hóa không an toàn bằng cách xem xét tất cả các dữ liệu được truyền vào trang web. Với mỗi ngôn ngữ khác nhau sẽ có định dạng tuần tự hóa đối tượng đặc trưng của ngôn ngữ đó nên dễ nhận biết. Cách phát hiện chủ yếu là dựa trên việc kiểm tra các header, tham số của request có định dạng tiêu biểu là định dạng tuần tự hóa PHP và định dạng tuần tự hóa Java.

### 3.1. Định dạng tuần tự hóa PHP

- PHP sử dụng định dạng chuỗi hầu như con người có thể đọc được, với các chữ cái biểu thị kiểu dữ liệu và các số biểu thị độ dài của mỗi mục nhập.

VD:

Graphical user interface, text

Description automatically generated with medium confidence

Khi được tuần tự hóa thì đối tượng này có dạng:



* o:4:”User” - một object mà tên lớp là **User**, có 4 ký tự.
* 2 - đối tượng User có 2 thuộc tính.
* s:4:”name” - tên thuộc tính thứ nhất là **name**, có **4** ký tự, kiểu dữ liệu **string**.
* s:6:”carlos” - giá trị của thuộc tính thứ nhất là **carlos**, có **6** ký tự, kiểu dữ liệu **string**.
* s:10:”isLoggedIn” – tên thuộc tính thứ hai là **isLoggedIn**, có **10** ký tự, kiểu **string.**
* **+** b:1 – giá trị của thuộc tính thứ hai là **1** tương ứng giá trị True kiểu **boolean.**

### 3.2. Định dạng tuần tự hóa Java

- Java sử dụng định dạng tuần tự hóa nhị phân, khiến cho khó đọc nhưng vẫn có một số dấu hiệu nhận biết.

- Ví dụ: các đối tượng Java được tuần tự hóa luôn bắt đầu bằng cùng một byte, được mã hóa thành ac ed ở dạng thập lục phân và rO0 ở dạng Base64.

Graphical user interface, text, application

Description automatically generated

Table

Description automatically generated

## 4. Thao tác với đối tượng tuần tự hóa

- Có hai cách tiếp cận thực hiện thao tác với các đối tượng được tuần tự hóa:

* Sửa đổi thuộc tính đối tượng
  + Khi giả mạo dữ liệu, miễn là kẻ tấn công truyền một đối tượng tuần tự hóa hợp lệ, quá trình giải tuần tự hóa sẽ tạo một đối tượng phía máy chủ với các giá trị thuộc tính đã sửa đổi.
  + Ví dụ: sửa đổi tham số isAdmin:b:0 thành isAdmin:b:1
* Sửa đổi kiểu dữ liệu của thuộc tính
  + Logic dựa trên PHP thường bị khai thác theo loại sửa đổi này. Do toán tử so sánh = = khi so sánh các loại dữ liệu khác nhau ví dụ như số nguyên và chuỗi thì PHP sẽ cố chuyển đổi chuỗi thành 1 số nguyên: 5 = = “5”
  + Với chuỗi bất kỳ được bắt đầu bằng 1 số, PHP sẽ chuyển toàn bộ chuỗi thành giá trị số nguyên dựa trên số ban đầu. VD 5 = = “5 abc” do được coi là 5 = = 5
  + Còn khi so sánh số 0 với chuỗi, nếu chuỗi đó không bắt đầu bằng 1 số thì PHP coi toàn bộ chuỗi đó là số 0

-> điều này sẽ dẫn đến các lỗi logic nghiêm trọng.

VD: sửa thuộc tính password chứa giá trị integer 0 thì khi ứng dụng so sánh với chuỗi password có sẵn sẽ luôn trả về true và có thể vượt qua cơ chế xác thực. Điều này chỉ khả thi khi giải tuần tự hóa bảo toàn kiểu dữ liệu.

Graphical user interface, text, application

Description automatically generated

Sửa đổi tên người dùng thành administrator, access\_token thành kiểu integer và có giá trị bằng 0

Graphical user interface

Description automatically generated with medium confidence

- không cần biết access\_token chính xác của admin là gì, ứng dụng giải tuần tự hóa, so sánh và vẫn trả về true nên có thể dùng chuỗi này để truy cập đường dẫn admin.

## 5. Các biện pháp phòng tránh

* Nên tránh giải tuần tự hóa đầu vào của người dùng trừ khi thực sự cần thiết.
* Nếu cần giải tuần tự hóa dữ liệu từ các nguồn không tin cậy, cần thực hiện kiểm tra tính toàn vẹn của đối tượng như sử dụng chữ ký số để xác thực đối tượng được tuần tự hóa, ngăn chặn việc sửa đổi chuỗi byte, tạo đối tượng giả mạo.

# IX. Using Components with known vulnerabilities

## 1. Khái niệm

- Using Components with known vulnerabilities là sử dụng các thành phần (components) trong phát triển phần mềm mà đã được biết đến có các lỗ hổng bảo mật, sự yếu kém trong mã hoặc có các vấn đề khác liên quan đến bảo mật.

- Thông thường, các thành phần này được lấy từ các thư viện, framework hoặc các codebase khác được phát triển bởi các bên thứ ba. Tuy nhiên, các thành phần này có thể chứa các lỗ hổng bảo mật đã biết đến nhưng không được cập nhật hoặc sửa chữa.

- Các hacker có thể quét ứng dụng web để có được các thông tin về phần mềm, phiên bản, thư viện,…mà ứng dụng đang sử dụng. Sau đó tìm kiếm về các lỗ hổng CVE đã được phát hiện và công bố của những thành phần này. Nếu thành phần, thư viện tồn tại lỗ hổng thì hacker thực hiện khai thác dựa trên PoC của lỗ hổng đó.

- VD: ứng dụng web sử dụng thư viện log4j (phiên bản <= 2.14.1) để ghi log, có tồn tại lỗ hổng Log4Shell.

(Log4Shell là một lỗ hổng bảo mật mạng quan trọng trên thư viện Log4j, ảnh hưởng đến hoạt động cốt lõi của thư viện. Nó cho phép kẻ tấn công có thể kiểm soát một thiết bị hoặc ứng dụng được kết nối mạng internet bằng cách thực hiện thực thi mã từ xa. Khi thực thi thành công kẻ tấn công có thể :

* Chạy bất kỳ mã nào trên thiết bị hoặc hệ thống
* Truy cập tất cả mạng và dữ liệu
* Sửa đổi hoặc mã hóa bất kỳ tệp nào trên ứng dụng hoặc thiết bị bị ảnh hưởng.)

## 2. Nguyên nhân

- Nguyên nhân của việc sử dụng các thành phần có lỗ hổng bảo mật có thể là do thiếu ý thức, áp lực thời gian phát triển sản phẩm, mã nguồn cũ, không cập nhật thường xuyên các thành phần hoặc nhà phát triển thiếu kinh nghiệm, kiến thức về bảo mật.

## 3. Hậu quả

- Các hậu quả của lỗ hổng:

* Tấn công được thực hiện dễ dàng: Nếu ứng dụng web sử dụng các thành phần đã biết có lỗ hổng bảo mật, kẻ tấn công có thể dễ dàng tìm thấy và khai thác các lỗ hổng này để thực hiện các cuộc tấn công.
* Nếu kẻ tấn công khai thác thành công các lỗ hổng bảo mật trong các thành phần được sử dụng, họ có thể gây ra mất dữ liệu hoặc chiếm quyền kiểm soát hệ thống.

## 4. Các biện pháp phòng tránh

* Các ứng dụng web cần có quản lý cập nhật để đảm bảo rằng các thành phần được sử dụng đều được cập nhật đầy đủ và kịp thời các bản vá lỗi
* Cần thường xuyên kiểm tra bảo mật các phiên bản của các thành phần được sử dụng, đảm bảo rằng chúng không có lỗ hổng bảo mật.
* Các nhà phát triển cần được đào tạo để có đủ hiểu biết và kiến thức về bảo mật để đánh giá và chọn lựa các thành phần an toàn để sử dụng trong ứng dụng web.
* Chỉ sử dụng các thành phần, thư viện do bên thứ 3 uy tín cung cấp.

# X. Insufficient logging and monitoring

## 1. Khái niệm

- Là một lỗi phổ biến trong ứng dụng web, đặc biệt là khi các ứng dụng được triển khai trên môi trường product, xảy ra khi hệ thống không đủ các cơ chế để ghi lại các hoạt động bất thường trong ứng dụng hoặc không theo dõi các hoạt động đó để phát hiện và xử lý kịp thời.

- Khi hệ thống không ghi lại đầy đủ các hoạt động và sự kiện trong ứng dụng, thì khi xảy ra sự cố hoặc tấn công, việc đánh giá và xác định nguyên nhân sẽ trở nên khó khăn hơn. Nếu không có các cơ chế giám sát và báo động thích hợp, các hoạt động độc hại có thể được thực hiện trong ứng dụng mà không bị phát hiện.

- Việc giám sát và ghi log cảnh báo sẽ góp phần giúp hệ thống phản ứng, ứng phó kịp thời với các loại khai thác tấn công đã biết và giúp ghi lại nhật ký để phân tích sau này đổi với những loại tấn công, lỗ hổng zeroday.

## 2. Ví dụ

- Khi ứng dụng bị tấn công DoS, do thiếu hệ thống giám sát hoặc phần log không ghi lại địa chỉ IP gửi request khiến cho hệ thống giám sát không có dữ liệu để thực hiện lock IP.

- Hoặc đối với người dùng từ bên trong mạng nội bộ, do thiếu cơ chế giám sát ghi log từ bên trong mà người dùng từ bên trong có thể truy cập, sửa, xóa các dữ liệu nhạy cảm mà không bị ghi lại nhật ký các hành động đã thực hiện trong mạng nội bộ.

## 3. Nguyên nhân

- Các nguyên nhân gây ra lỗ hổng Insufficient logging and monitoring bao gồm:

* Không cấu hình sử dụng thư viện ghi nhật ký, hoặc không cung cấp đầy đủ thông tin về các sự kiện và hoạt động xảy ra trên hệ thống.
* Không cung cấp chức năng giám sát để theo dõi các hoạt động của người dùng và hệ thống, từ đó không thể phát hiện các sự cố bảo mật kịp thời.
* Không tuân thủ các chuẩn bảo mật để cung cấp chức năng ghi nhật ký và giám sát hiệu quả, ghi nhật ký quá chung chung dẫn đến việc bỏ qua các hoạt động bất thường trên hệ thống.
* Thiếu cơ chế sao lưu dự phòng đối với log, khi xảy ra sự cố khiến mất hết các log đã ghi.

## 4. Các biện pháp phòng tránh

* Ghi lại chi tiết tất cả những failure trong hệ thống, đặc biệt là các failure về xác thực và trao quyền, các kiểm tra an toàn như chính sách CORS, …
* Log được cung cấp theo đúng định dạng mà các tool giám sát có thể tự động xử lý, phải đúng và đầy đủ dữ liệu cần thiết để dễ dàng phân tích khi cần.
* Phân loại log theo từng level để dễ xử lý như WARNING, ERROR, DEBUG, INFO
* Không được để thông tin nhạy cảm vào trong log. Chẳng hạn như nhập sai user password mà log chi tiết password not match 'xxxxx'.
* Cần bảo vệ các file log không bị truy cập trái phép, sao lưu dự phòng để tránh sự cố xảy ra.
* Triển khai các hệ thống giám sát truy cập để ứng phó kịp thời với bất thường xảy ra.