



Bảo mật web và ứng dụng

Nội dung

- OWASP
- Các công cụ
- OWASP Cheat sheets
- OWASP Top 10

OWASP là gì?

- **O**pen **W**eb **A**pplication **S**ecurity **P**roject
- Một tổ chức phi lợi nhuận hướng đến việc cải thiện an toàn phần mềm
- Website: <https://owasp.org/>



Các công cụ OWASP

- Tài liệu
- Các công cụ OWASP
 - ZAP: Zed Attack Proxy¹
 - Web Testing Environment²
 - Juice Shop³
- Cheat sheets
- Công cụ khác
 - Burp Suite⁴



OWASP
Zed Attack Proxy



1. <https://owasp.org/www-project-zap/>
2. <https://owasp.org/www-project-web-testing-environment>
3. <https://owasp.org/www-project-juice-shop/>
4. <https://portswigger.net/burp>

OWASP Cheat sheets

- Cung cấp các thông tin hữu ích và có giá trị cao về các **chủ đề an toàn, bảo mật thông tin** cụ thể.
- Được tạo ra bởi các **chuyên gia** trong lĩnh vực.

<https://cheatsheetseries.owasp.org/>

OWASP Cheat sheets

SQL Injection Prevention Cheat Sheet

Introduction

This article is focused on providing clear, simple, actionable guidance for preventing SQL Injection flaws in your applications. [SQL Injection](#) attacks are unfortunately very common, and this is due to two factors:

1. the significant prevalence of SQL Injection vulnerabilities, and
2. the attractiveness of the target (i.e., the database typically contains all the interesting/critical data for your application).

It's somewhat shameful that there are so many successful SQL Injection attacks occurring, because it is EXTREMELY simple to avoid SQL Injection vulnerabilities in your code.

SQL Injection flaws are introduced when software developers create dynamic database queries that include user supplied input. To avoid SQL injection flaws is simple. Developers need to either: a) stop writing dynamic queries; and/or b) prevent user supplied input which contains malicious SQL from affecting the logic of the executed query.

Table of contents

Introduction

Primary Defenses

Defense Option 1: Prepared Statements (with Parameterized Queries)

Defense Option 2: Stored Procedures

Defense Option 3: Whitelist Input Validation

Defense Option 4: Escaping All User-Supplied Input

Database Specific Escaping Details

Oracle Escaping

Escaping Dynamic Queries

Turn off character replacement

Escaping Wildcard

OWASP Cheat sheets

Cross Site Scripting Prevention Cheat Sheet

Introduction

This article provides a simple positive model for preventing [XSS](#) using output encoding properly. While there are a huge number of XSS attack vectors, following a few simple rules can completely defend against this serious attack.

This article does not explore the technical or business impact of XSS. Suffice it to say that it can lead to an attacker gaining the ability to do anything a victim can do through their browser.

Both [reflected and stored XSS](#) can be addressed by performing the appropriate validation and encoding on the server-side. [DOM Based XSS](#) can be addressed with a special subset of rules described in the [DOM based XSS Prevention Cheat Sheet](#).

For a cheatsheet on the attack vectors related to XSS, please refer to the [XSS Filter Evasion Cheat Sheet](#). More background on browser security and the various browsers can be found in the [Browser Security Handbook](#).

Table of contents

Introduction

A Positive XSS Prevention Model

Why Can't I Just HTML Entity Encode Untrusted Data

You Need a Security Encoding Library

XSS Prevention Rules

RULE #0 - Never Insert Untrusted Data Except in Allowed Locations

RULE #1 - HTML Encode Before Inserting Untrusted Data into HTML Element Content

RULE #2 - Attribute Encode Before Inserting Untrusted Data into HTML Common Attributes

RULE #3 - JavaScript Encode

OWASP Juice Shop

- Một ứng dụng web không an toàn được xây dựng có mục đích
- Mô phỏng các lỗ hổng trong Top 10 OWAPS và một số lỗ hổng khác
- Có thể được cấu hình tùy chỉnh



Công cụ Burp Suite

- **Decoder**

- Giải mã (decode) và mã hóa (encode) các chuỗi theo nhiều dạng format khác nhau

- **Proxy Server**

- Bắt các yêu cầu và tùy ý sửa đổi chúng trước khi gửi lên máy chủ

- **Repeater**

- Sửa đổi và gửi lại những request nhanh chóng

- **Comparer**

- Phân biệt được sự khác nhau giữa các yêu cầu (requests) và phản hồi (reponses)

- **Web spider**

- Tự động dò quét cấu trúc của một trang web

OWASP Top 10 Web Application Security Risks

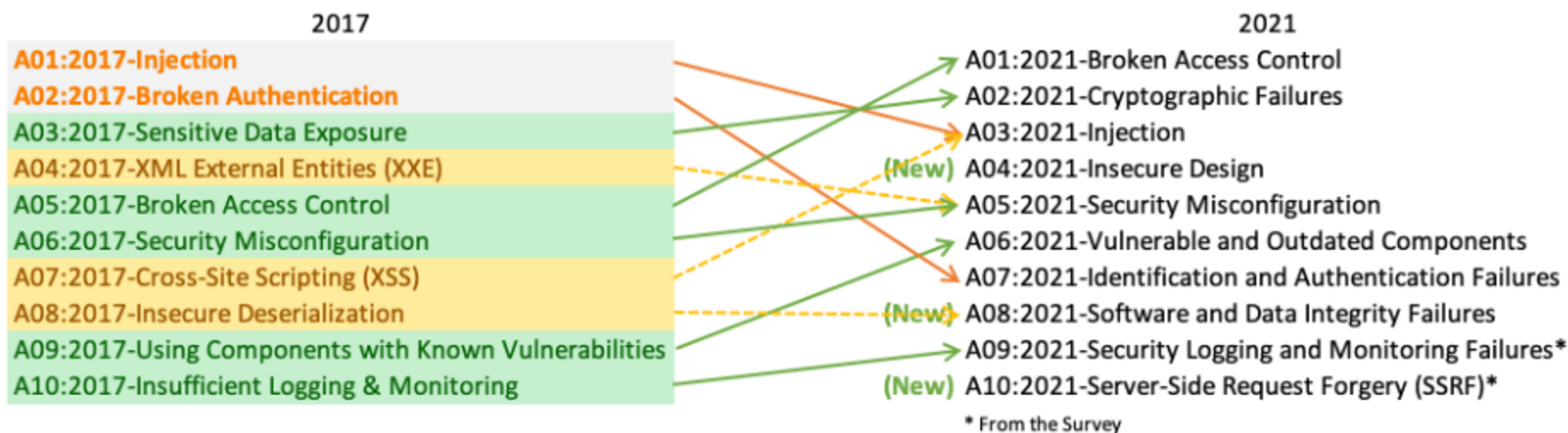
- Được lựa chọn từ các dữ liệu phổ biến và từ đề xuất của cộng đồng người dùng và các doanh nghiệp
- Phiên bản mới nhất: 2021

<https://owasp.org/Top10/>

OWASP Top 10 Web Application Security Risks

1. Broken Access Control
2. Cryptographic Failures
3. Injection
4. Insecure Design
5. Security Misconfiguration
6. Vulnerable and Outdated Components
7. Identification and Authentication Failures
8. Software and Data Integrity Failures
9. Security Logging and Monitoring Failures
10. Server-Side Request Forgery

So sánh với OWASP Top 10 2017



1. Broken Access Control

- Quyền truy cập luôn có sẵn cho bất kỳ ai mà không được cấp cho người dùng hay vai trò cụ thể
- Bypass việc kiểm tra truy cập bằng cách chỉnh sửa URL, trạng thái ứng dụng, mã nguồn HTML/HTTP request
- Cho phép xem hoặc chỉnh sửa tài khoản của người khác bằng cách sử dụng định danh duy nhất (vd: UUID)
- Truy cập API mà thiếu điều khiển truy cập cho các phương thức POST, PUT, DELETE
- Hoạt động với quyền của người dùng khi chưa đăng nhập hoặc đóng với trò là người quản trị khi đăng nhập với tài khoản người dùng thông thường
- Phát lại hoặc giả mạo JSON Web Token (JWT), cookie, token để nâng quyền hoặc lạm dụng JWT không hợp lệ
- Cấu hình sai CORS cho phép truy cập API từ nguồn trái phép hoặc không tin cậy

Biện pháp cho Broken Access Control

- Mặc định chặn truy cập, trừ những tài nguyên công cộng
- Triển khai các cơ chế kiểm soát truy cập và sử dụng trong toàn bộ ứng dụng
- Việc kiểm soát truy cập mô hình nên dựa trên quyền sử dụng bản ghi, hơn là việc người dùng có thể tạo/đọc/xóa/sửa bất kỳ bản ghi nào
- Tắt chức năng liệt kê thư mục trên web server và đảm bảo tập tin metadata (vd: git) và backup không có trong web root
- Ghi log các lỗi kiểm soát truy cập, cảnh báo đến người quản trị khi cần thiết
- Giới hạn tốc độ truy cập API và controller để giảm thiểu tác hại từ những công cụ tấn công tự động
- Vô hiệu hóa giá trị nhận dạng phiên khi đăng xuất, các mã JWT nên tồn tại trong thời gian ngắn để giảm khả năng bị tấn công

2. Cryptographic Failures

- Dữ liệu được truyền đi dưới dạng cleartext (HTTP, SMTP, FTP, Telnet)
- Dữ liệu được lưu trữ dưới dạng cleartext
- Sử dụng giao thực hoặc thuật toán mã hóa yếu hoặc lỗi thời
- Các khóa mặc định được sử dụng, tạo hoặc sử dụng lại các khóa yếu
- Mã hóa không được thực thi, ví dụ: tồn tại trang sử dụng HTTP
- Sử dụng hàm băm lỗi thời như MD5 hay SHA1

Biện pháp ngăn chặn Cryptographic Failures

- Phân loại dữ liệu được ứng dụng xử lý, lưu trữ hoặc truyền đi
- Xác định dữ liệu nào nhạy cảm theo luật/tiêu chuẩn bảo mật
- Không lưu dữ liệu nhạy cảm khi không cần thiết
- Đảm bảo mã hóa tất cả những dữ liệu nhạy cảm khi lưu trữ
- Mã hóa tất cả dữ liệu khi truyền đi bằng giao thức an toàn như TLS. Bắt buộc mã hóa bằng cách sử dụng chỉ thị như HSTS (HTTP Strict Transport Security)
- Tắt cache cho phản hồi chứa dữ liệu nhạy cảm
- Các khóa nên được tạo ngẫu nhiên và lưu trữ trong bộ nhớ
- Tránh sử dụng những hàm mật mã lỗi thời

3. Injection

- Ứng dụng thiếu cơ chế kiểm tra dữ liệu đầu vào, do đó có thể chấp nhận một số dữ liệu đầu vào chứa nội dung độc hại
- Có nhiều kiểu tấn công injection, bao gồm:
 - Code
 - Scripts
 - Commands được thực thi trên trình duyệt của nạn nhân
 - SQL/NoSQL
 - Các database commands có thể truy xuất hoặc thay đổi dữ liệu
 - OS commands
 - Chèn các command hệ thống để thực thi trên server của ứng dụng web

Ví dụ

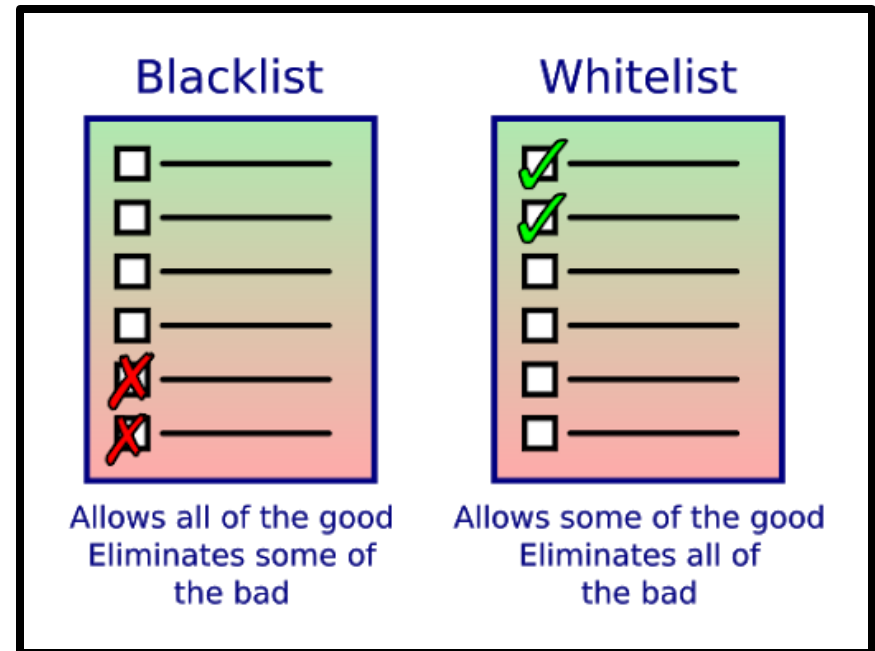
```
txtUserId = getRequestString("UserId");  
txtSQL = "SELECT * FROM Users WHERE UserId = " + txtUserId;
```

UserId:

```
SELECT * FROM Users WHERE UserId = 105 OR 1=1;
```

Biện pháp ngăn chặn Injection

- Kiểm tra dữ liệu nhập vào trên server, không nên phụ thuộc vào việc kiểm tra phía client
- Tạo danh sách các đầu vào hợp lệ (whitelist input)
- Sử dụng APIs phù hợp



4. Insecure Design

- Thiếu các yêu cầu liên quan đến bảo mật, tính toàn vẹn, tính sẵn sàng, tính xác thực của các tài sản dữ liệu
- Thiếu thông tin về logic kinh doanh dự kiến
- Thiếu phương pháp đánh giá liên tục các mối đe dọa và các tấn công đã biết
- Thiếu phân tích và thiếu giả định các luồng dự kiến và lỗi để đảm bảo ứng dụng hoạt động như mong muốn và chính xác

Biện pháp ngăn chặn Insecure Design

- Thiết lập và sử dụng những vòng đời phát triển an toàn (secure development lifecycle) giúp đánh giá và thiết kế các biện pháp liên quan đến bảo mật và quyền riêng tư
- Thiết lập và sử dụng những thư viện an toàn
- Sử dụng thread modeling cho việc xác thực, kiểm soát truy cập...
- Tách biệt các lớp (layer) trên hệ thống và mạng

5. Security Misconfiguration











- Các trang web, port, dịch vụ không được bảo vệ trước các truy cập trái phép
 - Ví dụ: ứng dụng cho phép liệt kê các thư mục, cho phép kẻ tấn công quét các file trên hệ thống
- Bật hoặc cài một số tính năng không cần thiết
- Các thông điệp báo lỗi (Error messages) cung cấp thông tin chi tiết về kiến trúc ứng dụng
 - Ví dụ: phiên bản các thư viện được sử dụng có thể được hiển thị trong error message, có thể giúp kẻ tấn công tìm ra các lỗ hổng hiện có của chúng
- Các tài khoản và mật khẩu mặc định vẫn được kích hoạt hoặc không thay đổi
- Nâng cấp hệ thống nhưng không bật những tính năng bảo mật mới nhất hoặc không cấu hình đúng đúng cách
- Phần mềm không được cập nhật hoặc có lỗ hổng

Biện pháp cho Security Misconfiguration

- Loại bỏ các tính năng, thành phần, tài liệu không cần thiết. Không cài đặt hoặc gỡ bỏ những tính năng và framework không sử dụng
- Xem xét các cập nhật phần mềm, bản vá cho phù hợp và nhanh chóng



Index of /wp-includes

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 Parent Directory		-	
 ID3/	2014-11-07 08:24	-	
 SimplePie/	2014-11-07 08:24	-	
 Text/	2014-11-07 08:24	-	
 admin-bar.php	2015-09-24 22:03	24K	
 atomlib.php	2015-09-24 22:03	11K	
 author-template.php	2015-09-24 22:03	14K	
 bookmark-template.php	2015-09-24 22:03	11K	
 bookmark.php	2015-09-24 22:03	13K	
 cache.php	2015-09-24 22:03	19K	

X

6. Vulnerable and Outdated Components

- Không rõ phiên bản các thành phần đang sử dụng, bao gồm các thành phần phụ thuộc
- Phần mềm có lỗ hổng, không còn được hỗ trợ, hoặc lỗi thời. Bao gồm: hệ điều hành, web server, hệ quản trị CSDL, ứng dụng, API, thư viện, ...
- Không thường xuyên dò quét các lỗ hổng, các tin tức bảo mật liên quan đến các thành phần đang sử dụng
- Cấu hình không an toàn cho các thành phần (Xem 5. Security Misconfiguration)



Biện pháp cho Vulnerable and Outdated Components

- Tất cả các thành phần, phần mềm của ứng dụng và hệ điều hành đều phải cập nhật phiên bản mới nhất
- Cần phải xem xét các báo cáo về lỗ hổng bảo mật cho các phần mềm đã cài đặt
- Cần phải thường xuyên tải các bản vá và nâng cấp phần mềm
- Chỉ nên nâng cấp phần mềm từ các nguồn chính thức

7. Identification and Authentication Failures

- Cho phép các mật khẩu mặc định, yếu, hay được sử dụng
- Sử dụng mật khẩu dưới dạng cleartext hoặc mã hóa bằng hàm băm yếu
- Không có cơ chế chống lại các tấn công tự động dò mật khẩu
- Lộ mã định danh session trên URL
- Các phương pháp khôi phục password không an toàn
- Sử dụng lại được những định danh session sau khi đăng nhập thành công

Top 10 Worst Passwords - Historic Analysis

	2021	2015	2010	2005	2000
#1	123456	123456	123456	password	password
#2	123456789	password	password	123456	123456
#3	qwerty	12345	12345678	12345678	12345678
#4	password	12345678	qwerty	abc123	qwerty
#5	1234567	qwerty	abc123	qwerty	abc123
#6	12345678	1234567890	123456789	monkey	monkey
#7	12345	1234	111111	letmein	1234567
#8	iloveyou	baseball	1234567	dragon	letmein
#9	111111	dragon	iloveyou	111111	trustno1
#10	123123	football	adobe123	baseball	dragon

Biện pháp cho Identification and Authentication Failures

- Sử dụng xác thực nhiều yếu tố (Multi-factor authentication) nếu có thể
- Không triển khai hoặc sử dụng bất kỳ thông tin xác thực mặc định nào, đặc biệt là người quản trị
- Thực hiện kiểm tra mật khẩu yếu
- Điều chỉnh độ dài, độ phức tạp và chính sách xoay vòng của mật khẩu
- Giới hạn số lần đăng nhập lỗi
- Sử dụng session ID an toàn

8. Software and Data Integrity Failures

- Liên quan đến code và cơ sở hạ tầng không được bảo vệ khỏi các vi phạm tính toàn vẹn
- Sử dụng những plugin, thư viện, mô-đun từ nguồn không đáng tin cậy
- Các ứng dụng có chức năng tự động tải xuống bản cập nhật mà không xác minh tính toàn vẹn và áp dụng cho ứng dụng tin cậy đã được cài đặt trước đó
- Dữ liệu hoặc các đối tượng (object) được encode và tuần tự hóa (serialize) thành một cấu trúc mà kẻ tấn công có thể xem và chỉnh sửa

9. Security Logging and Monitoring Failures

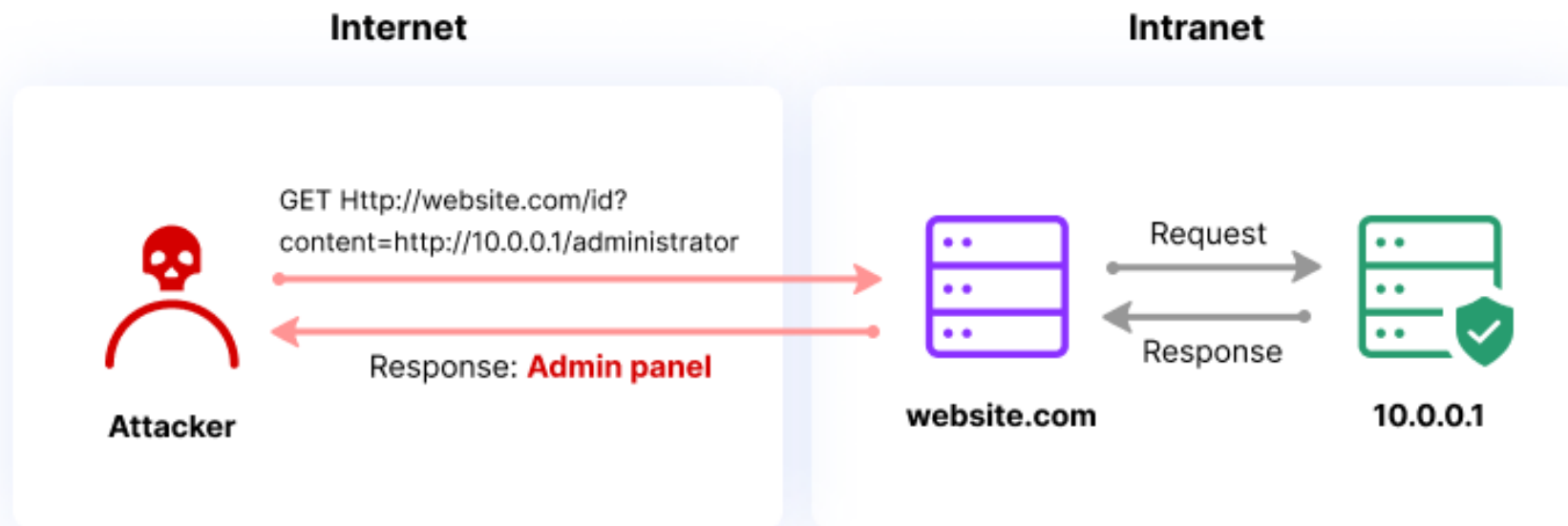
- Không ghi lại những sự kiện như login, login sai, giao dịch có giá trị cao
- Các cảnh báo hoặc lỗi tạo nhật ký không đầy đủ, không rõ ràng
- Log ứng dụng và API không ghi lại những hoạt động đáng ngờ
- Chỉ lưu trữ log cục bộ
- Không xem xét lại các sự kiện đã ghi log
- Ứng dụng không thể phát hiện báo cáo hoặc cảnh báo các cuộc tấn công đang active theo thời gian thực hoặc gần như vậy

Biện pháp cho Security Logging and Monitoring Failures

- Ghi log các sự kiện
 - Đăng nhập thành công/lỗi, điều khiển truy cập, các lỗi xác thực dữ liệu đầu vào phía server
- Logs phải đầy đủ chi tiết, định dạng phù hợp để các giải pháp quản lý log có thể sử dụng
- Log cần được mã hóa chính xác để ngăn chặn việc thay đổi hoặc tấn công
- Lưu trữ log ở các vị trí an toàn ngoài web server
- Logs phải được đối chiếu và xem xét để phát hiện kịp thời các hành vi bất thường
- Các giao dịch có giá trị cao phải được kiểm tra với các biện pháp kiểm soát tính toàn vẹn để ngăn chặn việc giả mạo hoặc xóa

10. Server-Side Request Forgery (SSRF)

- Xảy ra với bất kỳ ứng dụng web nào đang tải một tài nguyên từ xa và không xác thực URL của người dùng
- Nó cho phép kẻ tấn công gửi một yêu cầu được chỉnh sửa thủ công đến một đích (destination) không mong muốn, ngay cả khi được bảo vệ bởi tường lửa, VPN, ACL



Biện pháp cho SSRF

- Ở mức Network
 - Phân chia các tài nguyên ở những lớp mạng riêng biệt để giảm tác động của SSRF
 - Thực hiện chính sách **deny by default** trên tường lửa hoặc ACL để ngăn chặn những traffic không cần thiết
- Ở mức ứng dụng
 - Làm sạch (sanitize) và xác minh (validate) tất cả dữ liệu đầu vào mà người dùng cung cấp
 - Thực thi lược đồ URL, port, và đích (destination) với danh sách được cho phép
 - Không gửi dữ liệu chưa xử lý (raw data) đến người dùng
 - Tắt chuyển hướng HTTP

Các vấn đề TOP 10 trước đây

- Cross-Site Request Forgery (CSRF)
- Unvalidated Redirects and Forwards

Next steps

- Đọc tài liệu liên quan đến Top 10
- Xem các cheat sheets liên quan đến việc phát triển ứng dụng
- Các nội dung sẽ được đi sâu:
 - XSS (Cross-site scripting)
 - SQL Injection
 - CSRF
 - LFI & RFI

Tài liệu

- The OWASP Foundation
 - <https://www.owasp.org>
 - <https://owasp.org/www-project-top-ten/>

Bảo mật web và ứng dụng



Trường ĐH CNTT TP. HCM