LAB 1 - Tổng quan các lỗ hổng bảo mật web thường gặp

Bảo mật web và ứng dụng– NT213.P11.ANTN

Ngày báo cáo: 07/10/2024

GVHD: Ngô Khánh Khoa

Tên nhóm: 2

STT	Họ và Tên	MSSV
1	Nguyễn Đức Luân	22520825
2	Đào Hoàng Phúc	22521110
3	Vũ Ngọc Quốc Khánh	22520661

Security misconfiguration

Link video: Youtube

Loại lỗ hổng: Security misconfiguration

Mô tả lỗ hổng:

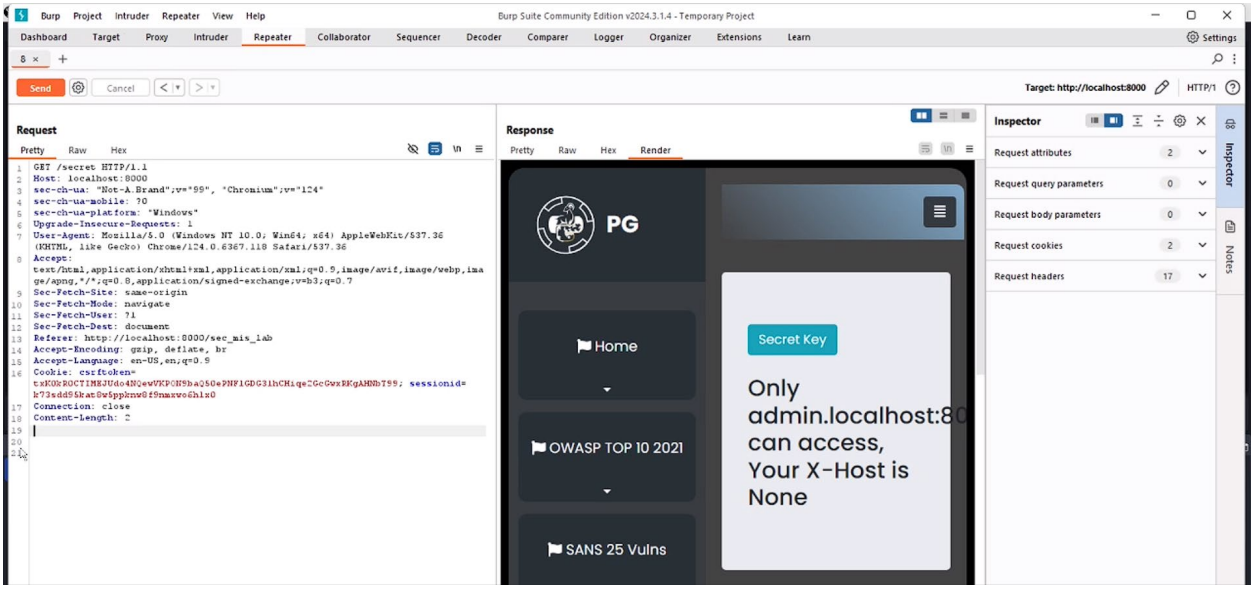
Lỗ hổng cấu hình này xảy ra do trang quản trị yêu cầu header HTTP "X-Host" có giá trị cụ thể là "admin.localhost:8080" để cấp quyền truy cập. Tuy nhiên, không có cơ chế bảo mật để kiểm tra và giới hạn người dùng có thể thay đổi header này, dẫn đến việc bất kỳ ai cũng có thể điều chỉnh header và truy cập vào trang quản trị trái phép.

Mức độ ảnh hưởng:

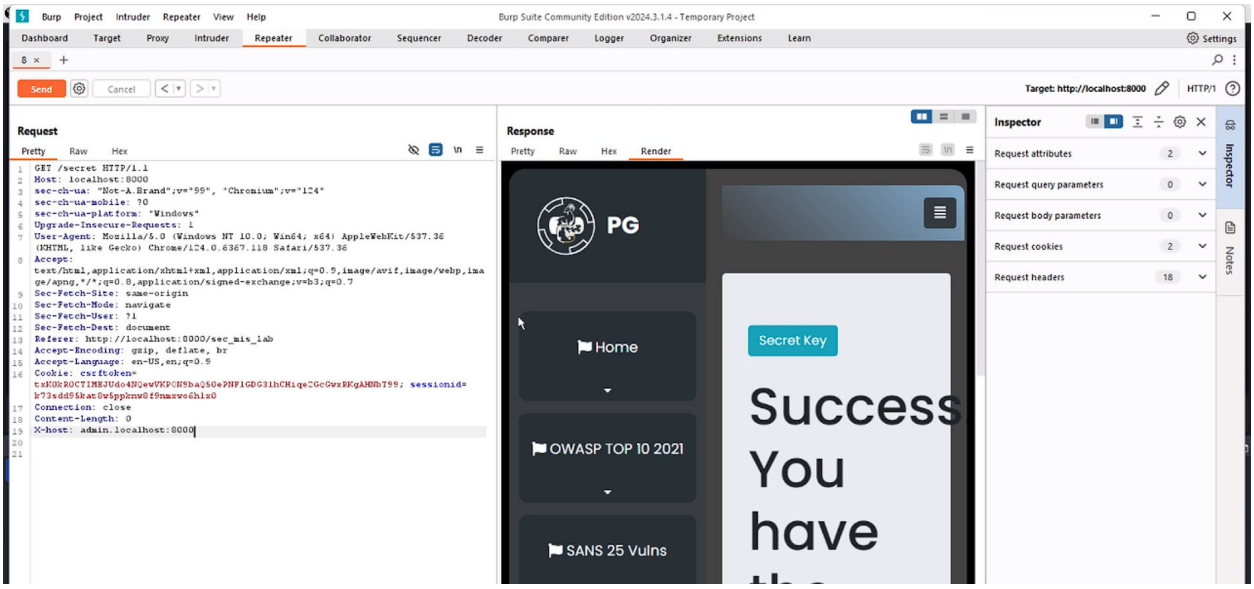
Lỗ hổng này cho phép kẻ tấn công có thể truy cập và chiếm quyền điều khiển trang quản trị mà không cần thông tin đăng nhập hợp lệ, từ đó có thể chỉnh sửa cấu hình hệ thống, xem hoặc thay đổi dữ liệu nhạy cảm, và thực hiện các thao tác quản trị nguy hiểm khác. Mức độ ảnh hưởng là rất nghiêm trọng nếu trang quản trị không có các biện pháp bảo mật khác như xác thực người dùng hoặc giới hạn truy cập IP.

Trình bày các bước:

Ở đây để vào được trang admin thì gói tin cần có X-Host là admin.localhost:8080



Thêm giá trị trên vào trường X-host là ta pass được lab này



Đề xuất giải pháp:

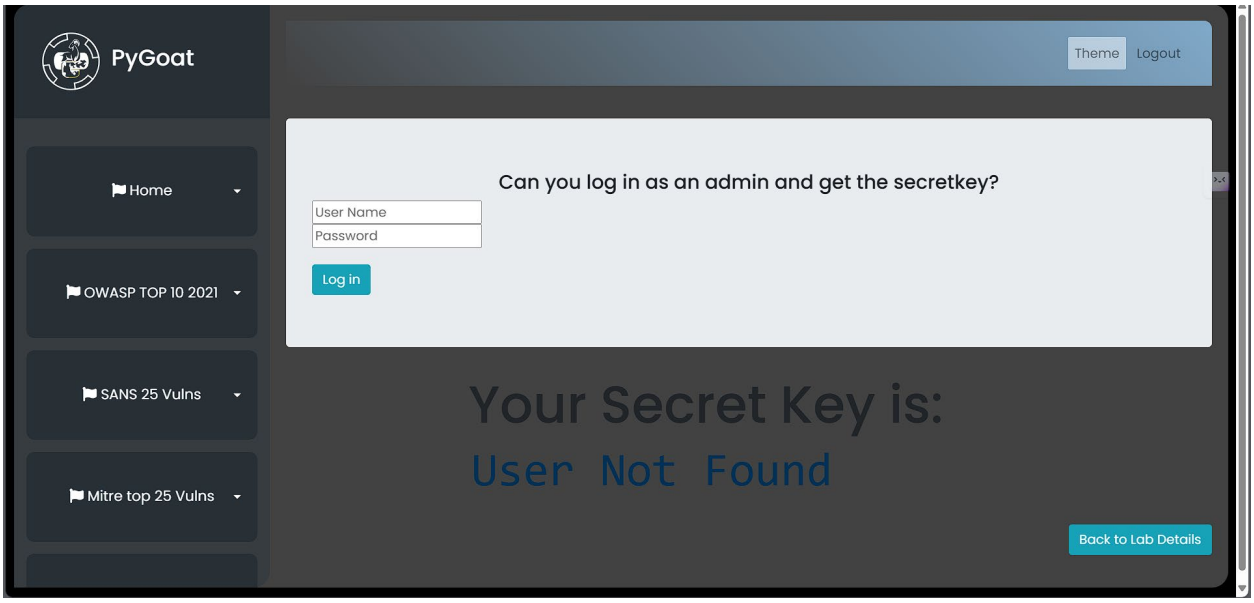
- Bảo mật header X-Host: Kiểm tra chặt chẽ giá trị X-Host hoặc loại bỏ nếu không cần thiết.
- Thực hiện xác thực đa yếu tố (MFA): Yêu cầu người dùng thực hiện xác thực qua nhiều bước như SMS, email hoặc ứng dụng xác thực.

- Giới hạn truy cập theo địa chỉ IP: Chỉ cho phép một dải IP tin cậy (ví dụ: nội bộ hoặc VPN) truy cập trang quản trị.
- Sử dụng HTTPS và kiểm tra header: Yêu cầu mọi kết nối tới trang quản trị phải qua HTTPS để bảo mật các thông tin truyền tải.
- Xác thực người dùng trước khi vào trang quản trị: Bắt buộc người dùng phải đăng nhập với thông tin hợp lệ trước khi truy cập trang quản trị
- Cấu hình kiểm soát truy cập (RBAC): Áp dụng kiểm soát truy cập dựa trên vai trò, chỉ cấp quyền quản trị cho người dùng được phép.

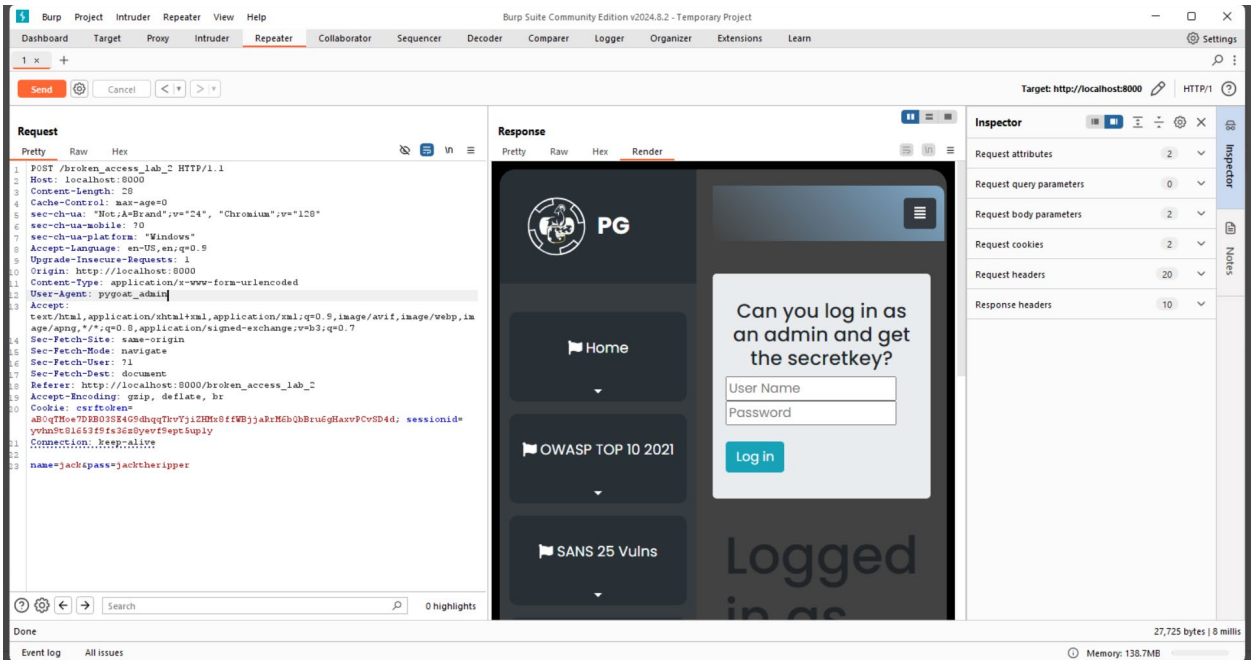
Broken-Access control 2:

Link video: Youtube

Không vào được admin, nhưng đề bài có gợi ý rằng hãy thay user agent thành pygoat_admin



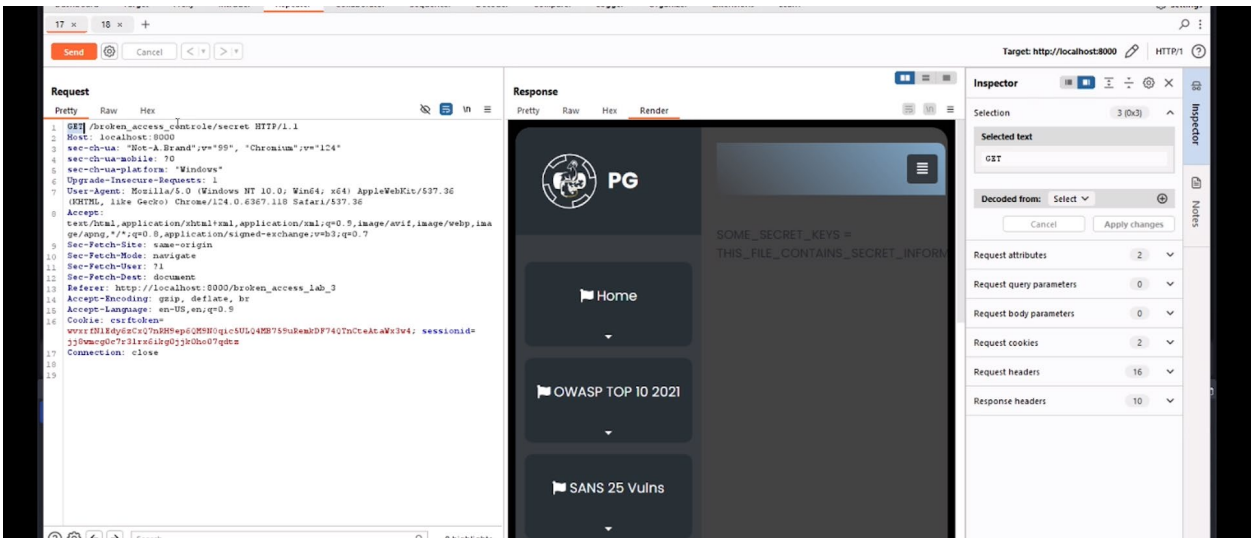
Thay user-agent ta pass được lab



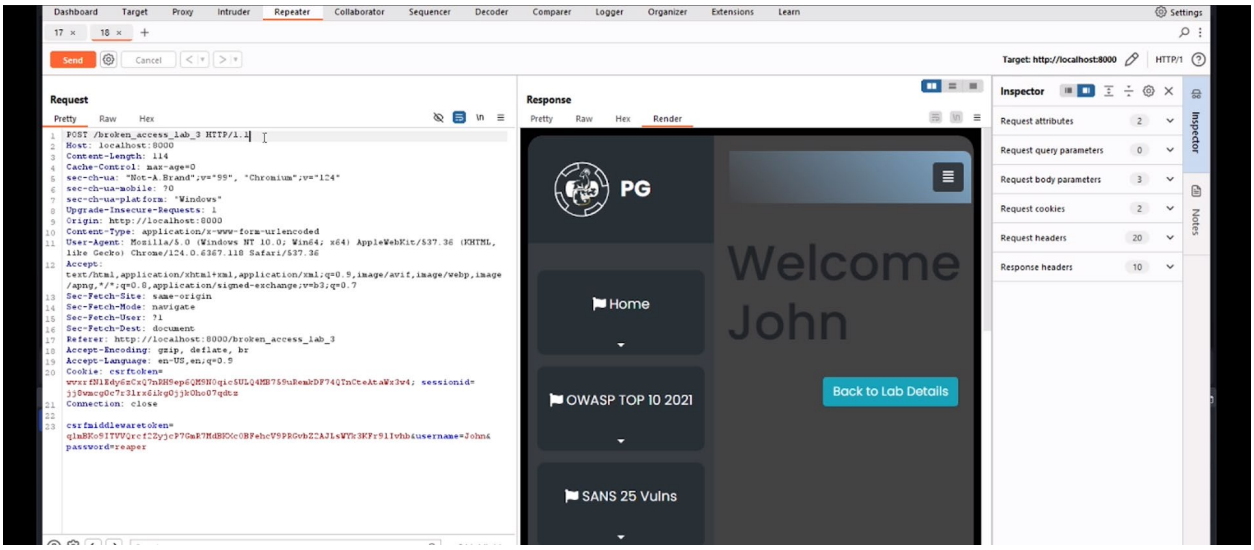
Broken Acess Control 3

Link video: Phần 1 và Phần 2

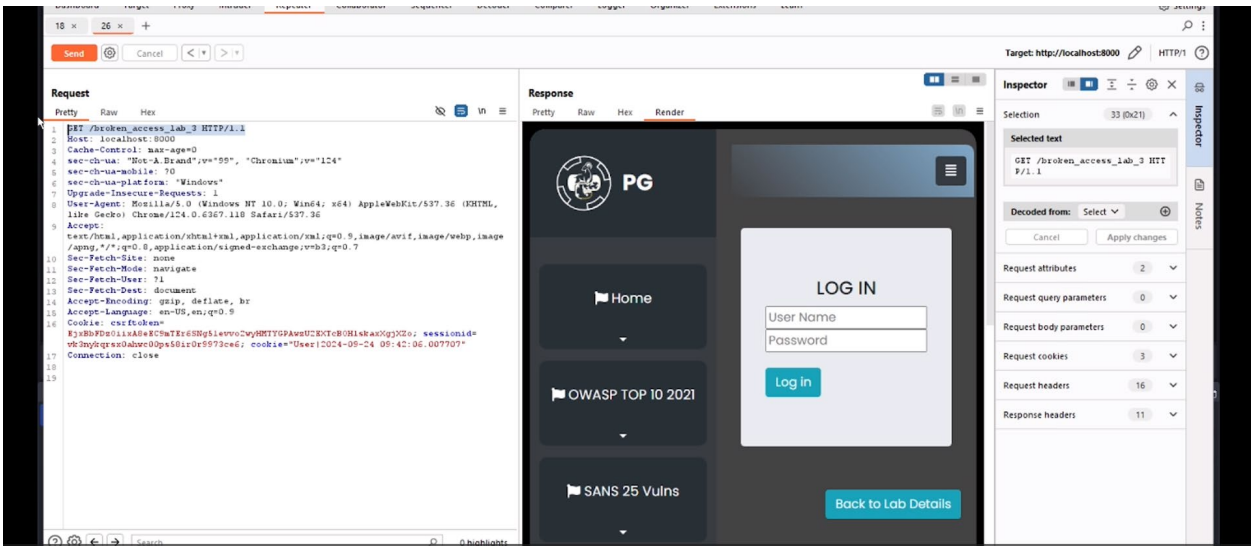
Ta vào được trang chứa secret với quyền admin



Lấy đường dẫn tới trang đó ta vào được trang secret với account của John



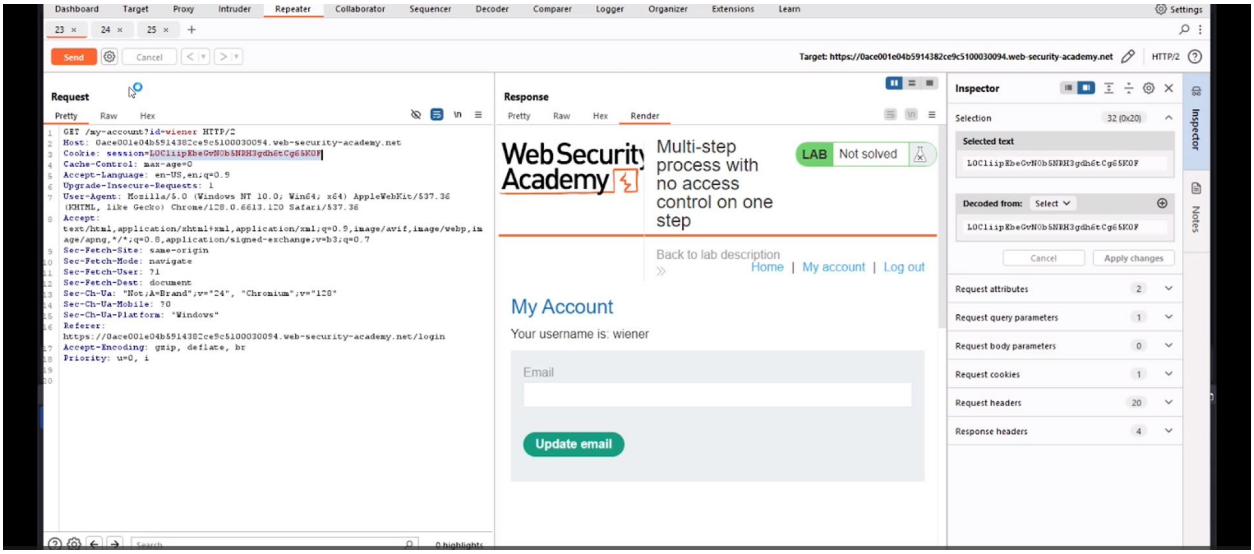
Ở đây cũng chỉ cần đưa đường dẫn vào là ta có thể vào thẳng trang secret dù không cần đăng nhập



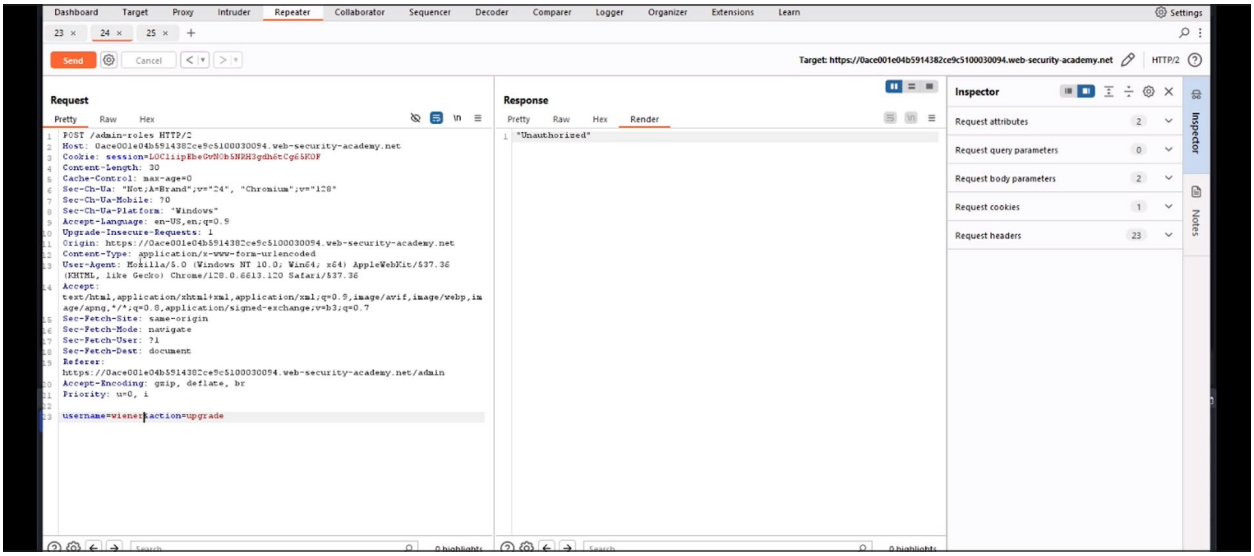
https://portswigger.net/web-security/access-control/lab-multi-step-process-with-no-access-control-on-one-step

Link video: YOUTUBE

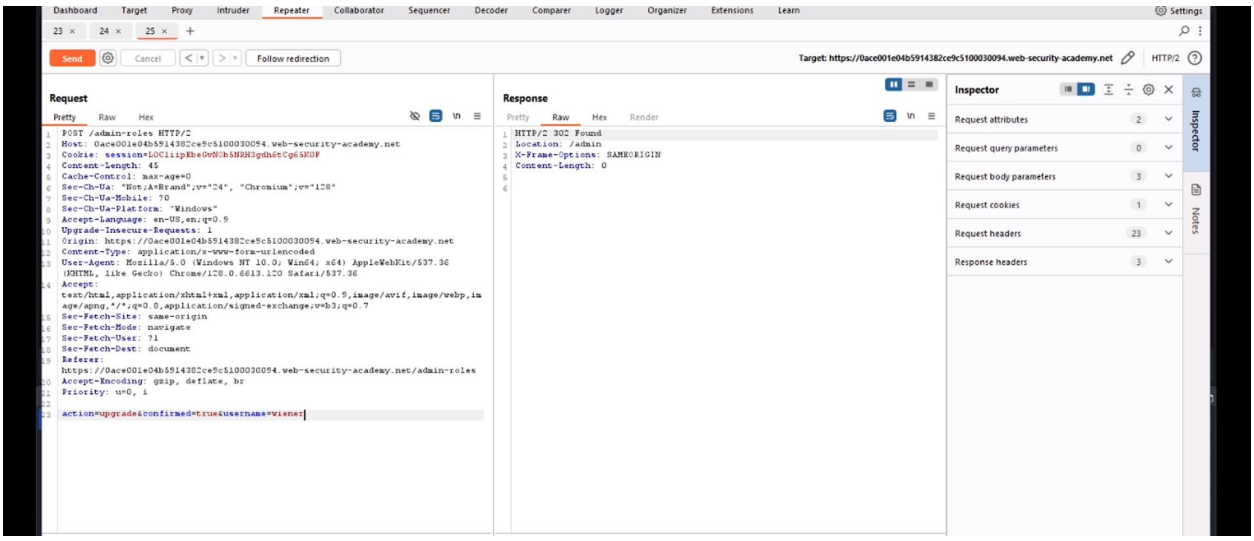
Ở đây user không có quyền chuyển đổi thành admin, tuy nhiên ta có thể thay đổi được trường session thành giá trị session của admin



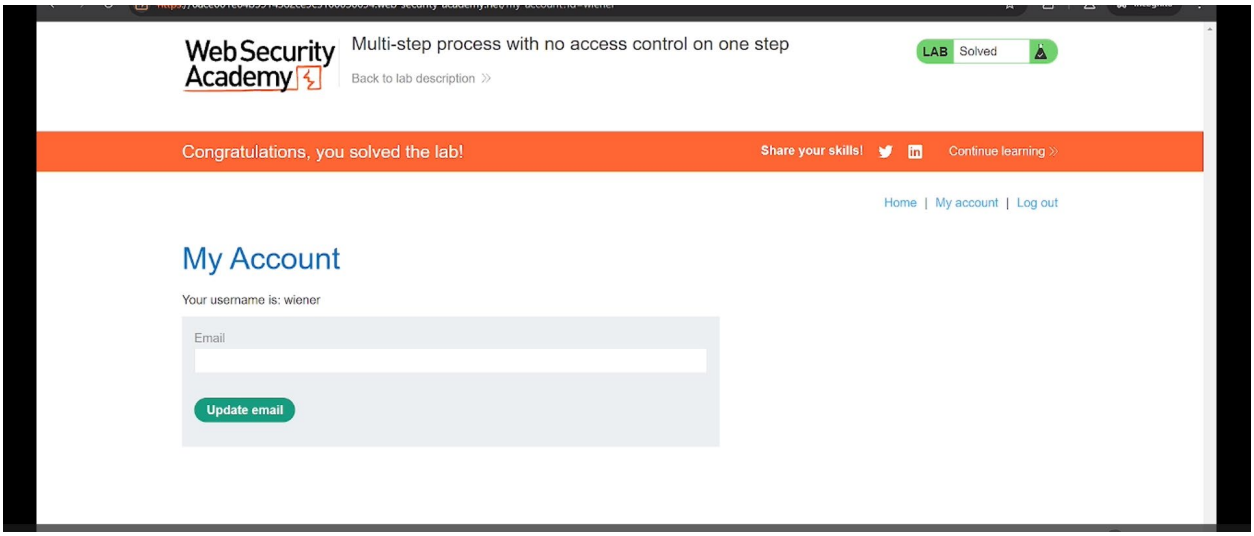
Thay đổi cookie session và tên của user Carlos ở request nâng quyền cho Carlos sang user Wiener, sau đó send đi



Thực hiện request để nâng quyền cho user Wiener với Cookie và Username đã thay đổi



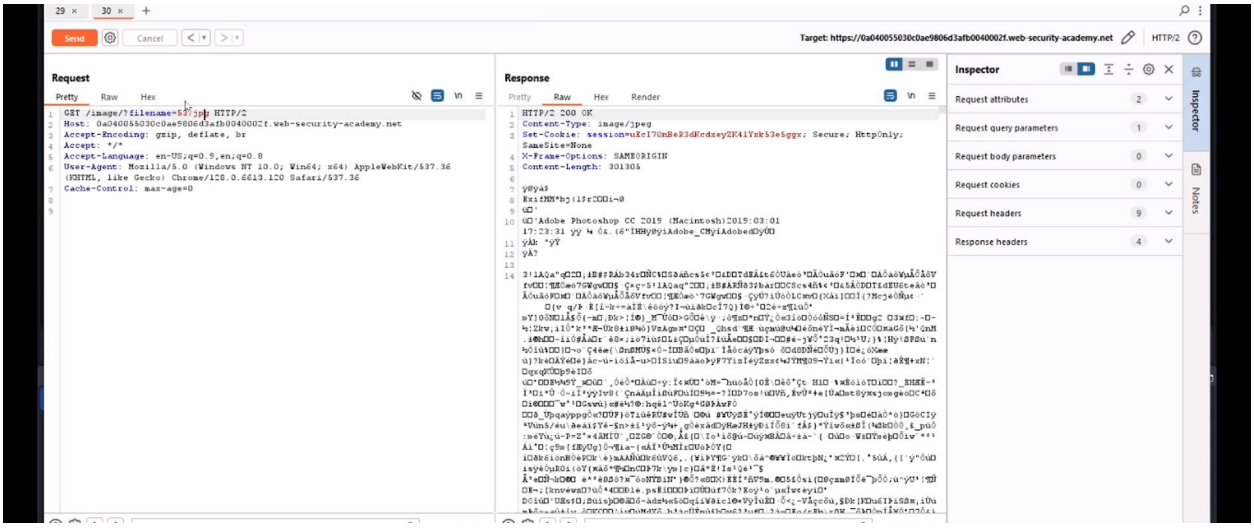
Bài lab hoàn thành



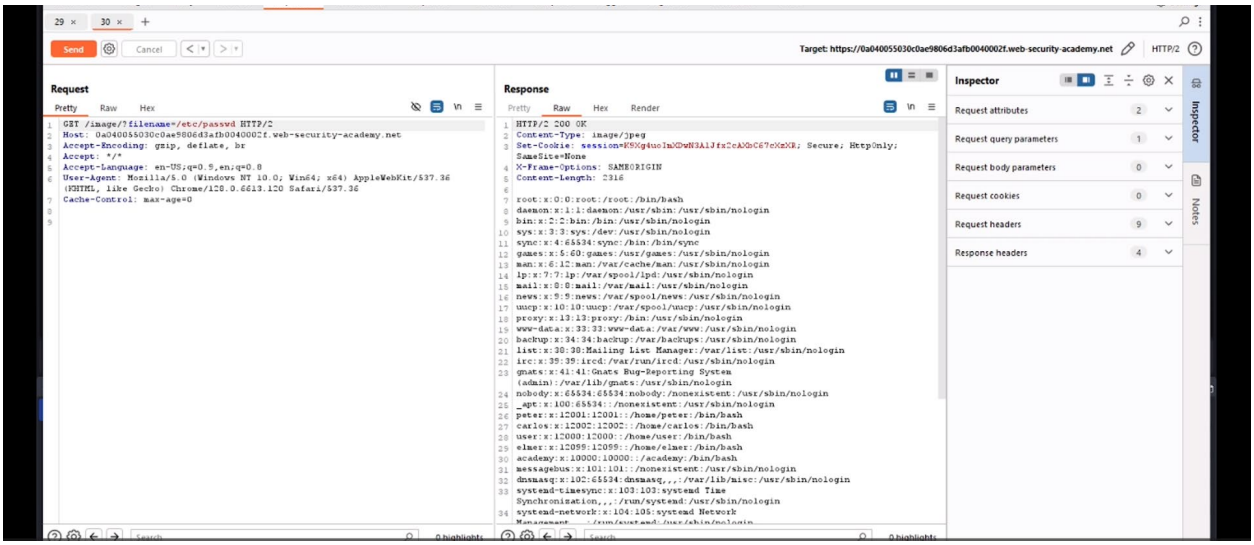
https://portswigger.net/web-security/file-path-traversal/lab-absolute-path-bypass

Link video: Youtube

Khi bấm vào 1 post bất kỳ trong trang, chúng ta có ảnh được load lên bằng cách lấy file ở trường filename, và ta có thể sử dụng trường này để file path traversal



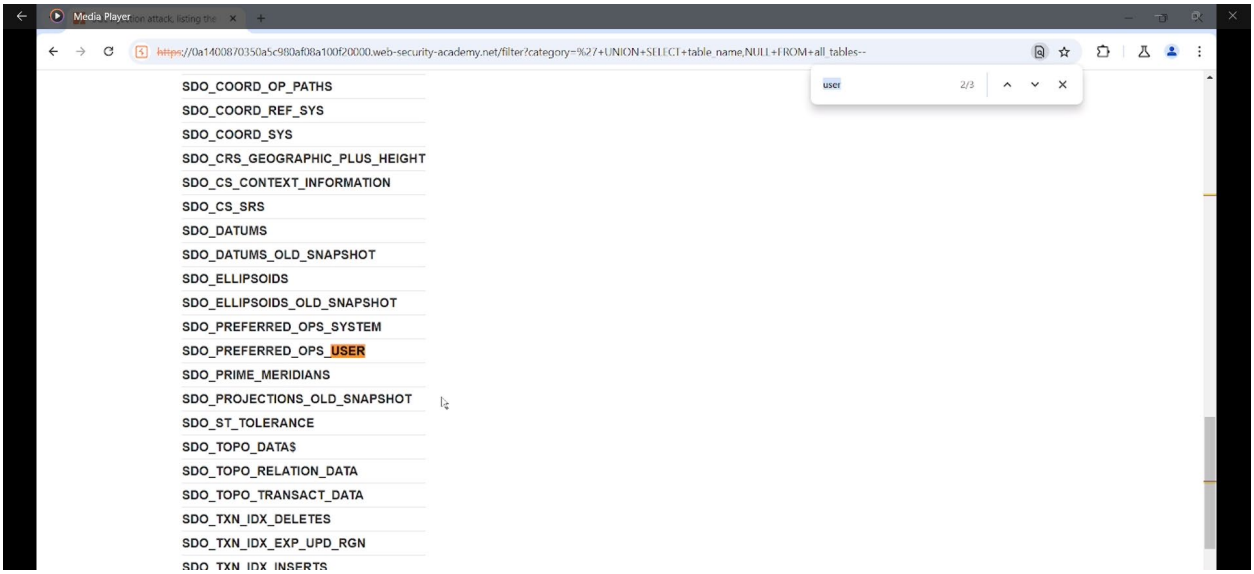
Thay giá trị của trường filename trên thành /etc/passwd, ta có giá trị ẩn cần tìm



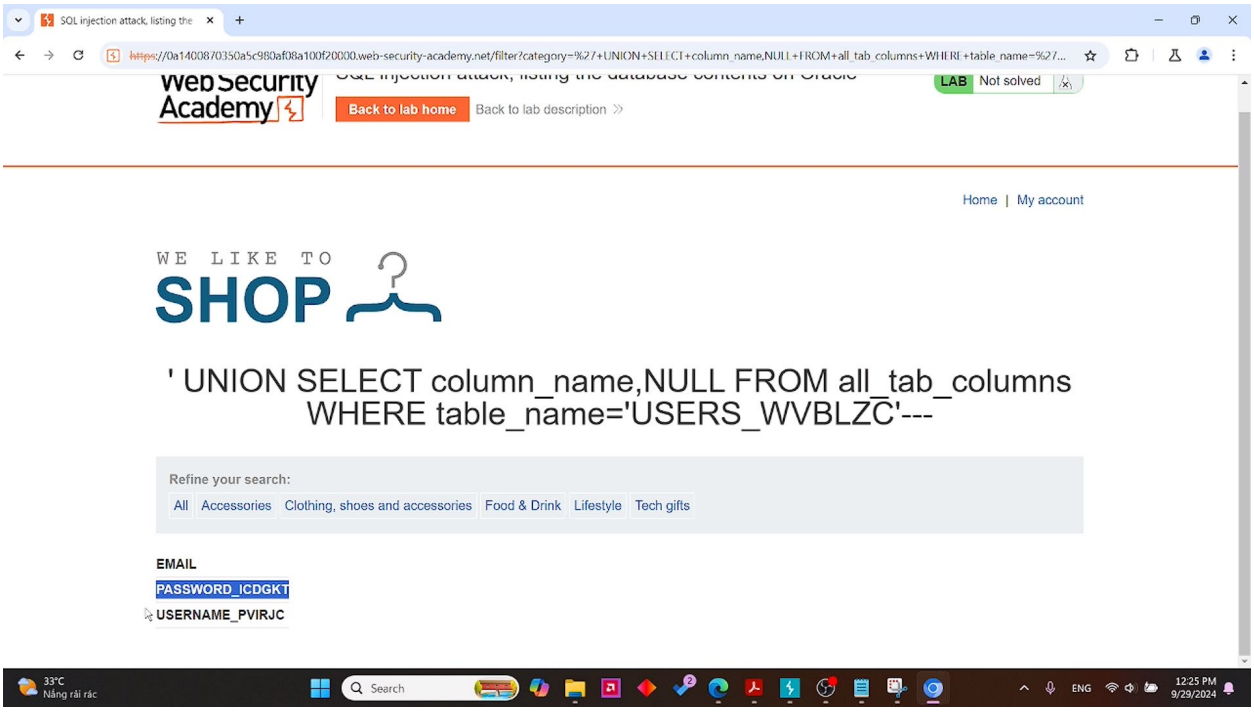
https://portswigger.net/web-security/sql-injection/examining-the-database/lab-listingdatabase-contents-oracle

Link video: Youtube

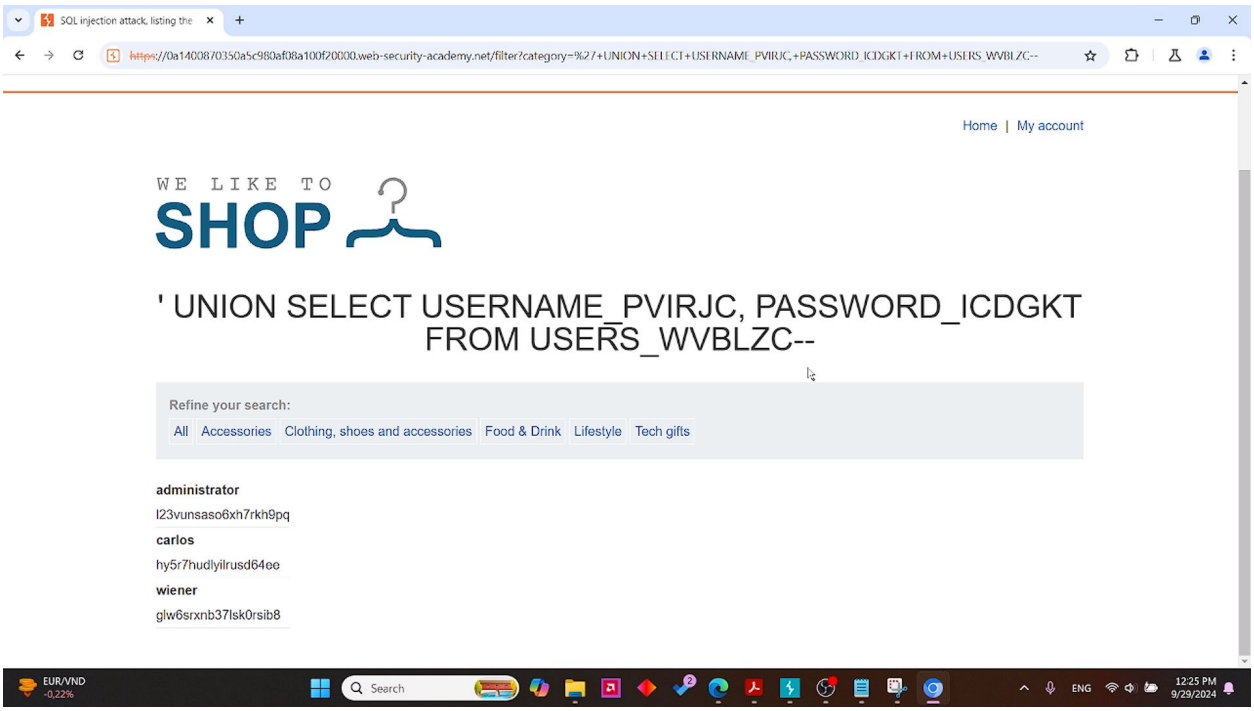
Thực hiện bỏ payload vào query để tìm ra tên table chứa user



Truy vấn từ table trên để tìm ra tên các cột có trong table



Thực hiện truy vấn để tìm ra tất cả các user, và ta có user admin



Cryptographic Failures 2

Link video: Youtube

Bước 1: Lấy hash password của admin và đảo ngược thứ tự

Bước 2: Tìm kiếm crack hash vừa đảo ngược và tìm ra kết quả: password777

Bước 3: Đăng nhập bằng admin:password777

Cryptographic Failures 3

Link video: Youtube

Bước 1: Đăng nhập bằng tài khoản user

Bước 2: Sửa cookie thành admin

Sensitive Data Exposure

Link video: Youtube

Bước 1: Truy cập vào 1 trang không tồn tại để kích hoạt lỗi 404

Bước 2: Tìm đường link dẫn đến 500error

Bước 3: Tìm với từ khóa SENSITIVE

Security Misconfiguration 3

Link video: Youtube

Bước 1: Lấy cookie sau khi đăng nhập

Bước 2: Sử dụng trang jwt.io để chỉnh sửa jwt

Bước 3: Tìm SECRET_COOKIE_KEY trong mã nguồn của pygoat

Bước 4: Thay role thành admin và secret key là PYGOAT

Bước 5: Sửa cookie cũ thành cookie mới

Logic flaws infinite money (Insecure Design)

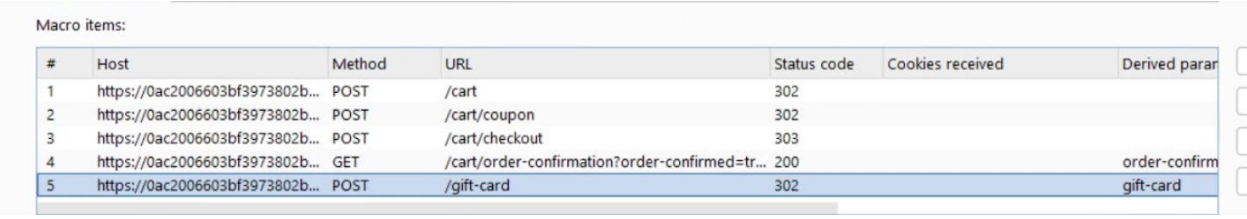
Bước 1: Đăng nhập bằng tài khoản winer:peter

Bước 2: Sử dụng email được cung cấp để đăng kí vào newsletter để nhận voucher SIGNUP30

Bước 3: Mua gift card giá $10, sử dụng voucher SIGNUP30 để giảm 30% còn $7.

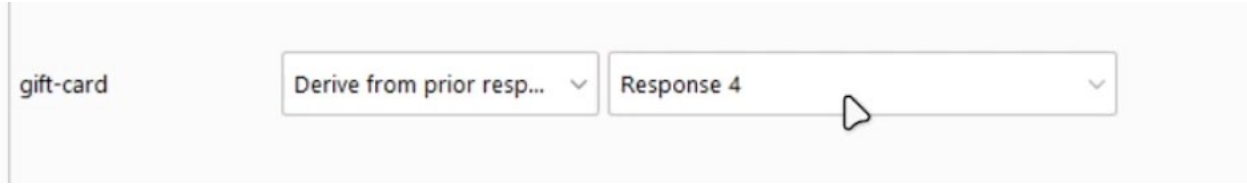
Bước 4: Sử dụng mã gift card để nhận lại $10, từ đây ta biết cách khai thác

Bước 5: Sử dụng chức năng macro trong Burp suite để tấn công, lựa chọn 5 requests liên quan đến việc chọn gift card vào giỏ hàng, sử dụng voucher, thanh toán, sử dụng mã gift card.


Macro items:

#	Host	Method	URL	Status code	Cookies received	Derived parar
1	https://0ac2006603bf3973802b...	POST	/cart	302		
2	https://0ac2006603bf3973802b...	POST	/cart/coupon	302		
3	https://0ac2006603bf3973802b...	POST	/cart/checkout	303		
4	https://0ac2006603bf3973802b...	GET	/cart/order-confirmation?order-confirmed=tr...	200		order-confirm
5	https://0ac2006603bf3973802b...	POST	/gift-card	302		gift-card

Bước 6: Ở request GET /cart/order-confirmation, ta tùy chọn để có thể thay đổi được mã gift card mỗi lần và truyền thành data input cho POST /gift-card bên dưới



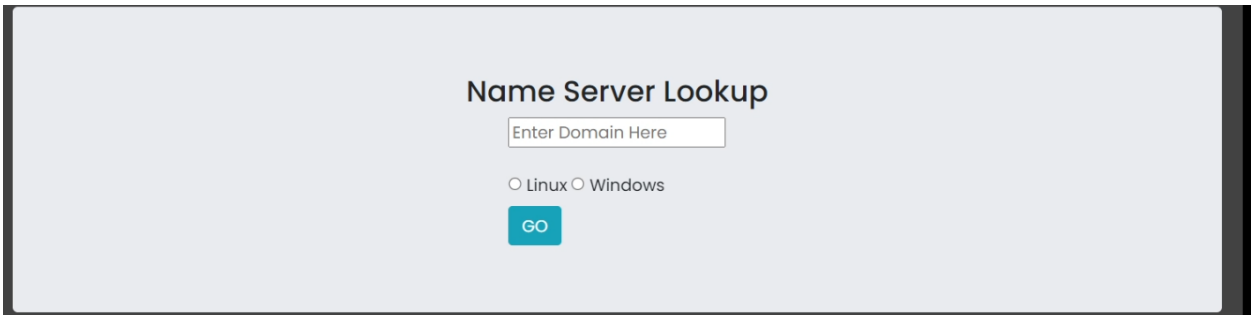
Bước 7: Ở request GET /my-account, ta chuyển request sang intruder, gửi payload null và lặp lại 412 lần để đạt được $1337

Bước 8: Mua món đồ để kết thúc

Command injection:

Link video: Youtub

Định nghĩa : là một lỗ hổng bảo mật xảy ra khi kẻ tấn công có thể thực thi các lệnh tùy ý trên hệ thống thông qua một ứng dụng dễ bị tấn công. Lỗ hổng này thường xuất hiện khi dữ liệu do người dùng cung cấp không được kiểm tra kỹ càng và được đưa trực tiếp vào một lệnh hệ thống, tạo điều kiện cho kẻ tấn công gửi các lệnh độc hại.

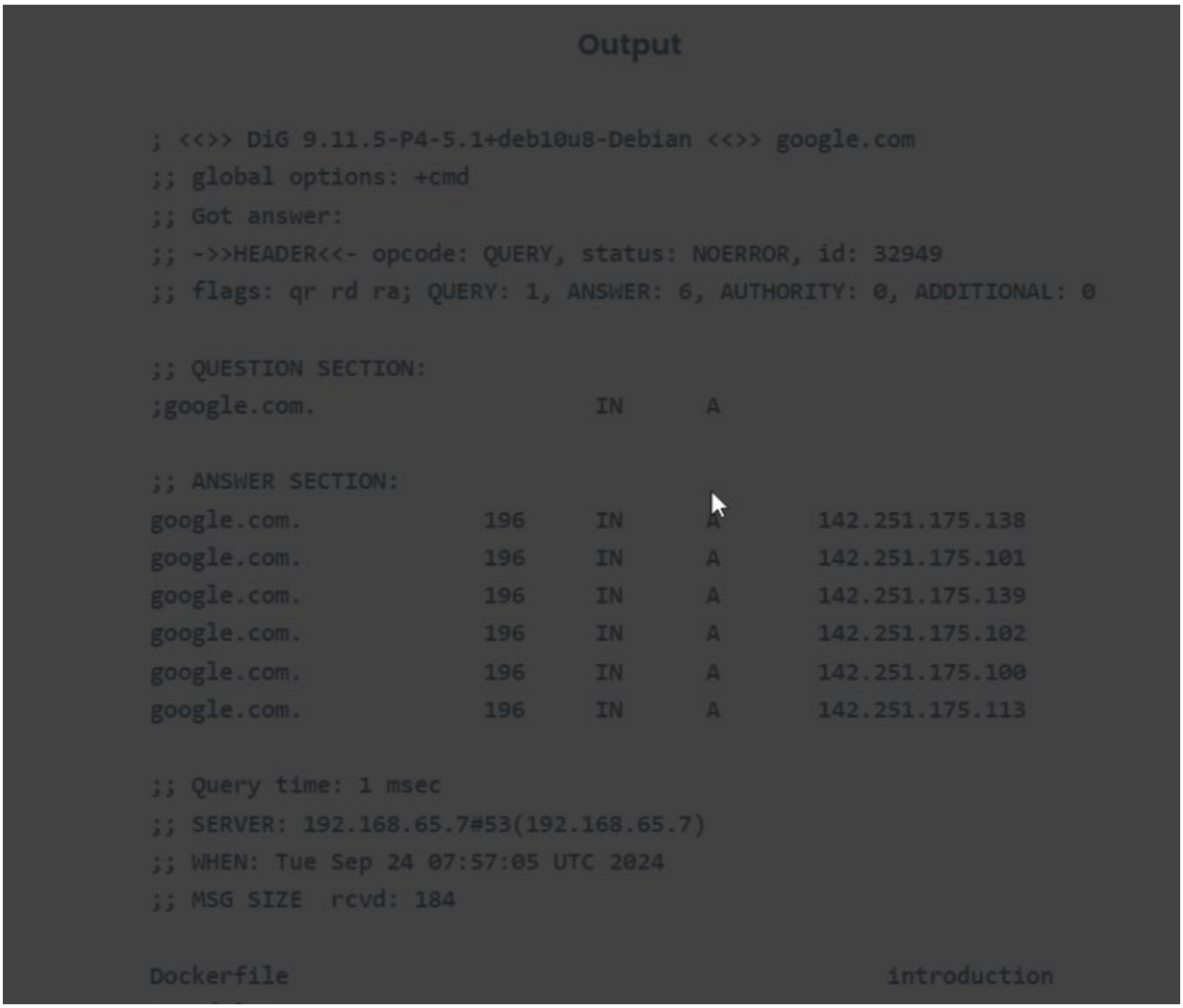


Mục tiêu: thực hiện chèn câu lệnh thực thi vào chuỗi truy vấn

Như ta đã biết thì name sever lookup hoạt động như sau:

```
command="nslookup {}".format(domain)
```

Vì thế nếu ta nhập chuỗi truy vấn như : google.com && dir thì name sever lookup sẽ truy vấn google.com sau đó thực hiện lệnh dir



Mức độ ảnh hưởng:

Thực thi mã từ xa: Kẻ tấn công có thể thực thi mã trên máy chủ, từ đó chiếm quyền kiểm soát hệ thống, đánh cắp dữ liệu hoặc làm gián đoạn dịch vụ.

Rò rỉ dữ liệu: Thông tin nhạy cảm trên máy chủ có thể bị truy xuất bởi kẻ tấn công.

Tấn công leo thang đặc quyền: Nếu lệnh được thực thi với quyền cao (như quyền root), kẻ tấn công có thể chiếm quyền kiểm soát toàn bộ hệ thống.

Phá hủy dữ liệu hoặc làm gián đoạn hệ thống: Các lệnh độc hại có thể xóa dữ liệu hoặc làm cho hệ thống không hoạt động.

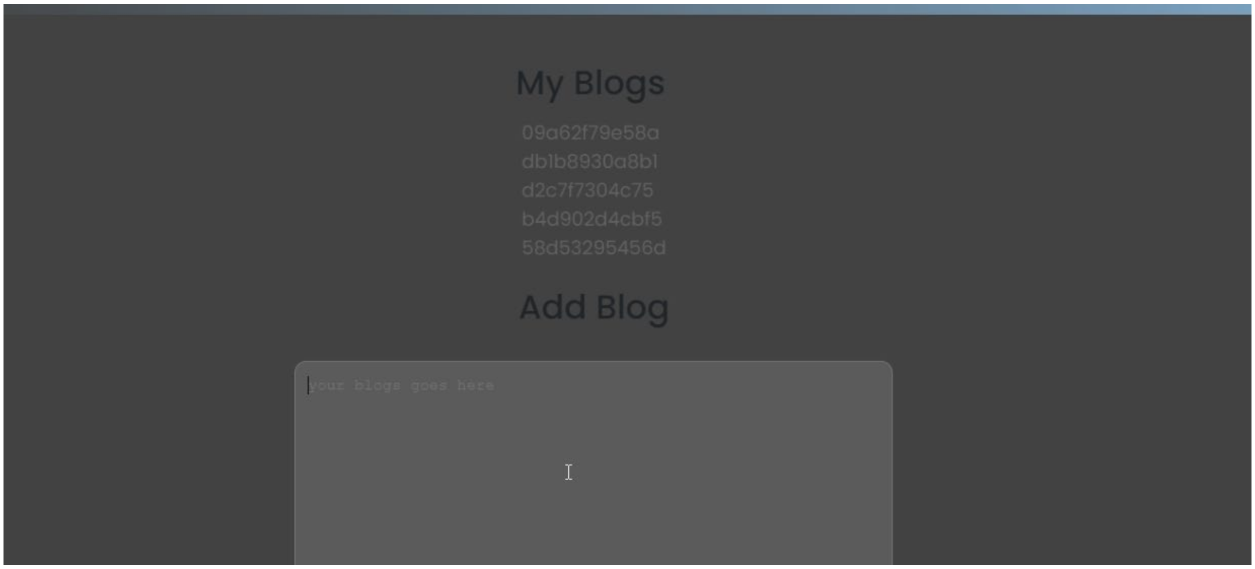
Cách ngăn ngừa Command Injection

1. **Kiểm tra và xác thực đầu vào của người dùng**: Không bao giờ tin tưởng dữ liệu đầu vào từ người dùng. Hãy sử dụng các phương pháp kiểm tra chặt chẽ hoặc giới hạn các kiểu đầu vào được chấp nhận.
2. **Sử dụng các API an toàn**: Thay vì trực tiếp thực thi các lệnh hệ thống từ chuỗi lệnh người dùng cung cấp, hãy sử dụng các API có cơ chế tự động thoát (escaping) và xử lý tham số an toàn.
3. **Sử dụng cơ chế escaping**: Khi phải sử dụng shell commands, hãy đảm bảo thoát các ký tự đặc biệt một cách an toàn (escaping) để ngăn chặn các chuỗi độc hại.
4. **Hạn chế quyền hệ thống**: Chạy ứng dụng với quyền hạn thấp nhất có thể để giảm thiểu thiệt hại trong trường hợp bị tấn công.
5. **Tách biệt môi trường**: Sử dụng sandboxing hoặc container hóa để cô lập các dịch vụ, giúp ngăn ngừa lan rộng của một cuộc tấn công nếu bị xâm phạm.

Template injection

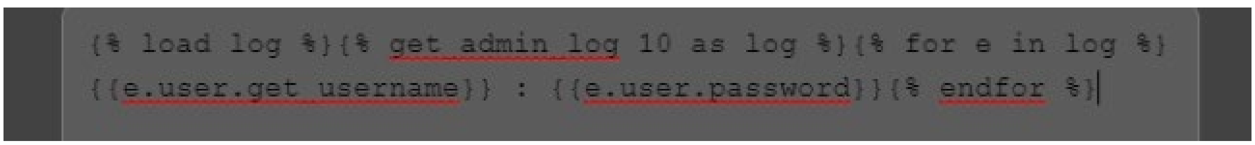
Link video: Youtube

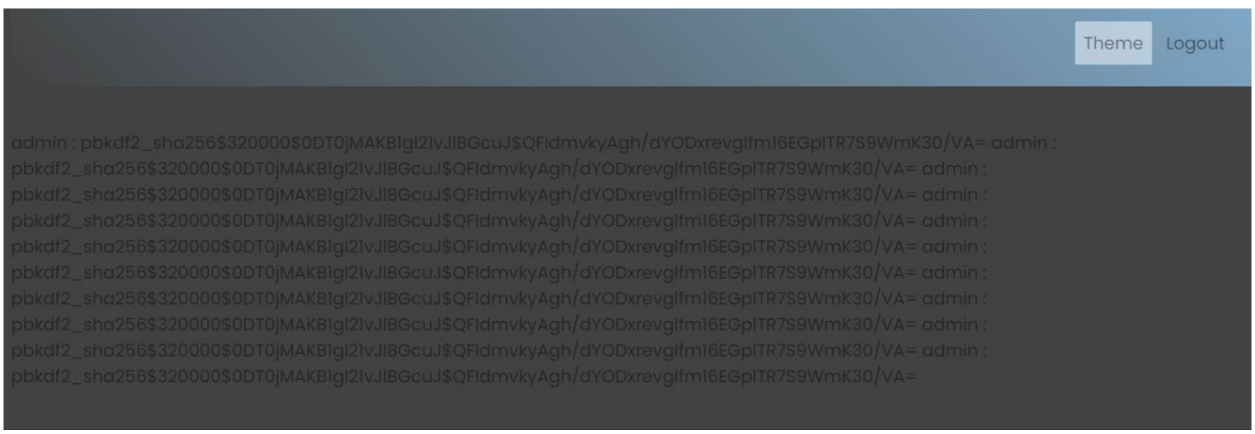
Định nghĩa: **Template Injection** là một lỗ hổng bảo mật xảy ra khi kẻ tấn công có thể tiêm mã vào các mẫu (template) động của ứng dụng web. Template Injection thường xuất hiện trong các ứng dụng sử dụng hệ thống template để hiển thị nội dung động mà không xử lý hoặc lọc kỹ lưỡng dữ liệu đầu vào từ người dùng.



Mục tiêu: dùng django default template để lấy được hash password của admin

Cách làm:





Mức độ ảnh hưởng :

Thực thi mã từ xa (RCE): Trong trường hợp SSTI, kẻ tấn công có thể thực thi mã trên máy chủ, dẫn đến khả năng chiếm quyền kiểm soát hệ thống, đánh cắp dữ liệu, hoặc gây ra các cuộc tấn công leo thang đặc quyền.

Rò rỉ dữ liệu: Template Injection có thể bị lợi dụng để truy xuất các dữ liệu nhạy cảm như biến môi trường, thông tin cấu hình của hệ thống, hoặc thông tin cá nhân của người dùng.

Tấn công Cross-Site Scripting (XSS): Trong CSTI, kẻ tấn công có thể tiêm mã JavaScript vào trình duyệt của người dùng, gây ra các cuộc tấn công XSS, đánh cắp cookie hoặc thông tin nhạy cảm.

Tấn công từ chối dịch vụ (DoS): Kẻ tấn công có thể sử dụng injection để tạo ra các vòng lặp vô hạn hoặc làm cho máy chủ quá tải, gây gián đoạn dịch vụ.

Cách ngăn ngừa Template Injection

Kiểm tra và lọc dữ liệu đầu vào: Hạn chế tối đa việc cho phép người dùng cung cấp các đoạn mã hay biến có thể ảnh hưởng đến hệ thống template.

Không render trực tiếp dữ liệu không đáng tin: Không bao giờ render trực tiếp dữ liệu người dùng trên template mà không qua kiểm tra kỹ lưỡng.

Sử dụng các hệ thống template an toàn: Một số template engine hiện nay đã có các cơ chế an toàn như escaping tự động hoặc sandboxing, giúp hạn chế khả năng thực thi mã độc hại.

Escaping nội dung đặc biệt: Thực hiện escaping các ký tự đặc biệt như {, }, $ khi render dữ liệu đầu vào của người dùng vào template.

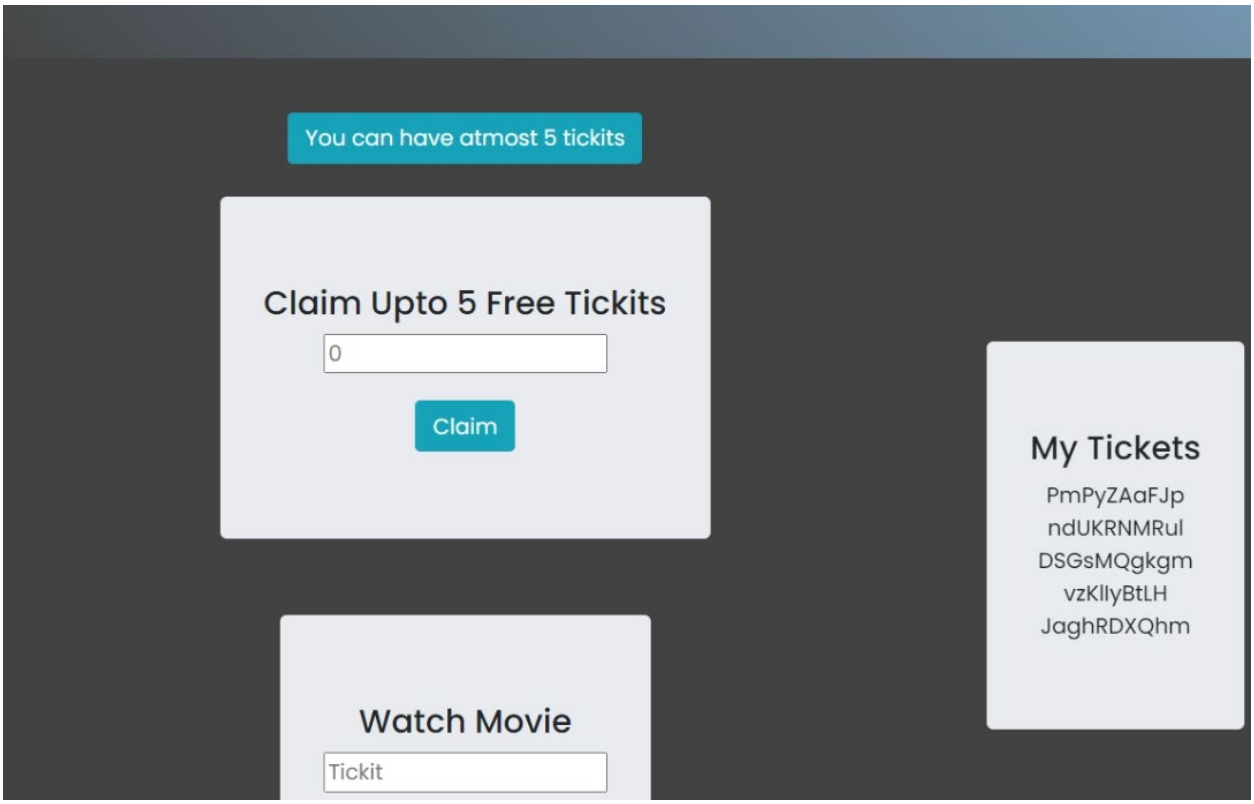
Cấu hình an toàn cho template engine: Nhiều hệ thống template cho phép thiết lập quyền hạn để giới hạn những gì có thể thực thi. Ví dụ, có thể cấm việc gọi các hàm nguy hiểm hoặc giới hạn tài nguyên cho phép truy cập từ template.

Tách biệt các vùng mã và dữ liệu: Giữ mã template và dữ liệu của người dùng tách biệt rõ ràng, giúp giảm thiểu khả năng tấn công qua injection.

Insecure design

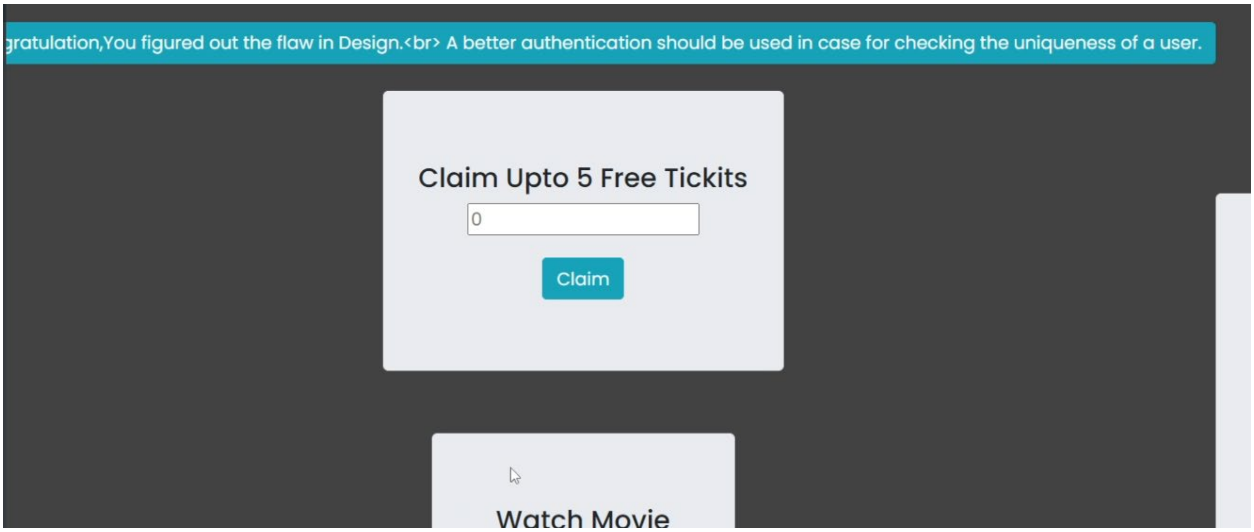
Link video: Youtube

Định nghĩa: **Insecure Design** là một lỗ hổng bảo mật phát sinh do những quyết định thiết kế sai lầm hoặc thiếu sót trong quá trình phát triển một hệ thống hoặc ứng dụng. Đây không phải là một lỗ hổng kỹ thuật cụ thể mà là một vấn đề liên quan đến cách hệ thống được thiết kế, dẫn đến khả năng bị khai thác bởi các lỗ hổng bảo mật sau này.



Mục tiêu : Khai thác lỗi thiết kế của web

Tạo tài khoản mới và đăng nhập nhận vé đến khi hết vé (60 vé)



Mức độ ảnh hưởng:

Tạo ra nhiều lỗ hổng bảo mật: Thiết kế không an toàn có thể dẫn đến việc các lỗ hổng bảo mật dễ dàng bị khai thác, dẫn đến các cuộc tấn công mạng, đánh cắp dữ liệu, hoặc làm gián đoạn hệ thống.

Thiệt hại tài chính và uy tín: Các cuộc tấn công xuất phát từ insecure design có thể gây ra thiệt hại tài chính lớn cho doanh nghiệp, đồng thời làm suy giảm uy tín của tổ chức.

Chi phí sửa chữa cao: Khi phát hiện lỗ hổng trong giai đoạn sử dụng, việc sửa chữa một thiết kế không an toàn thường đòi hỏi nhiều công sức và chi phí, đặc biệt là khi phải thay đổi cấu trúc cơ bản của hệ thống.

Cách ngăn ngừa Insecure Design

1. **Thiết kế bảo mật ngay từ đầu (Security by Design)**: Bảo mật phải là một phần trong quá trình thiết kế và phát triển ứng dụng. Các nhà phát triển nên tuân thủ các nguyên tắc bảo mật và thực hành tốt nhất trong giai đoạn thiết kế, không nên xem bảo mật là một yếu tố phụ trợ sau khi phát triển.
2. **Xác thực và phân quyền chặt chẽ**: Cần phải có các cơ chế xác thực mạnh mẽ (như multi-factor authentication) và phân quyền rõ ràng, đảm bảo rằng mỗi người dùng chỉ có quyền truy cập vào những tài nguyên mà họ được phép.
3. **Mã hóa dữ liệu nhạy cảm**: Tất cả dữ liệu nhạy cảm phải được mã hóa, cả khi lưu trữ và truyền tải, để ngăn ngừa việc rò rỉ thông tin nếu bị tấn công.
4. **Kiểm tra và lọc dữ liệu đầu vào**: Sử dụng các cơ chế kiểm tra và lọc dữ liệu đầu vào để ngăn chặn các cuộc tấn công phổ biến như **SQL Injection**, **XSS**, và **Command Injection**.
5. **Kiểm tra bảo mật thường xuyên**: Thực hiện các kiểm tra bảo mật thường xuyên, bao gồm cả kiểm tra code, kiểm tra thiết kế, và thực hiện các bài test xâm nhập (penetration testing) để phát hiện và khắc phục các lỗ hổng.
6. **Sử dụng các tiêu chuẩn bảo mật**: Áp dụng các tiêu chuẩn bảo mật như **OWASP Top 10** hoặc các quy chuẩn bảo mật ngành khác để hướng dẫn quá trình thiết kế và phát triển.

labprototype-pollution-client-side-prototype-pollution-via-browser-apis

Link video: Youtub

Định nghĩa: **Client-Side Prototype Pollution** là một lỗ hổng bảo mật cho phép kẻ tấn công sửa đổi hoặc tiêm các thuộc tính vào prototype của các đối tượng JavaScript, ảnh hưởng đến tất cả các đối tượng con kế thừa từ prototype này. Khi khai thác qua **Browser APIs**, lỗ hổng này có thể gây ra nhiều hậu quả nghiêm trọng trong các ứng dụng web.

Web Security Academy | Client-side prototype pollution via browser APIs

Back to lab description

LAB Not solved

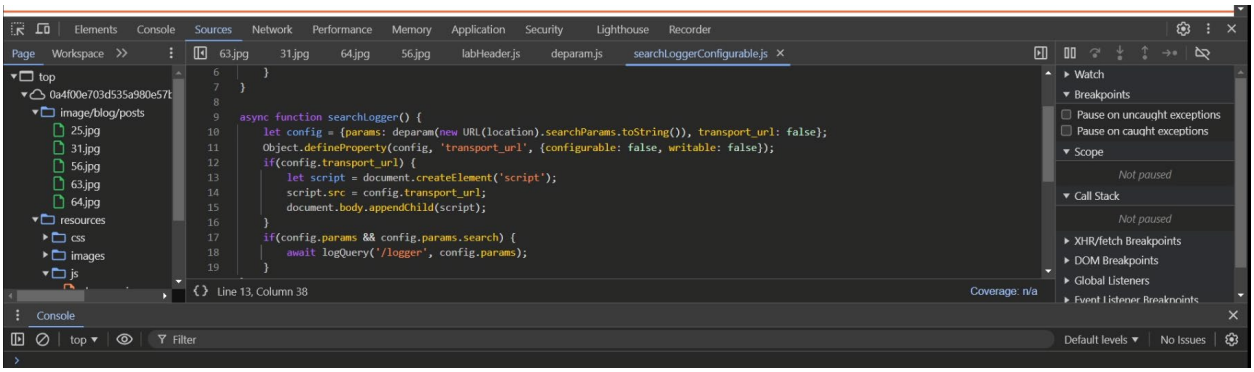
Home | My account | Submit feedback

WE LIKE TO BLOG

Search the blog... Search

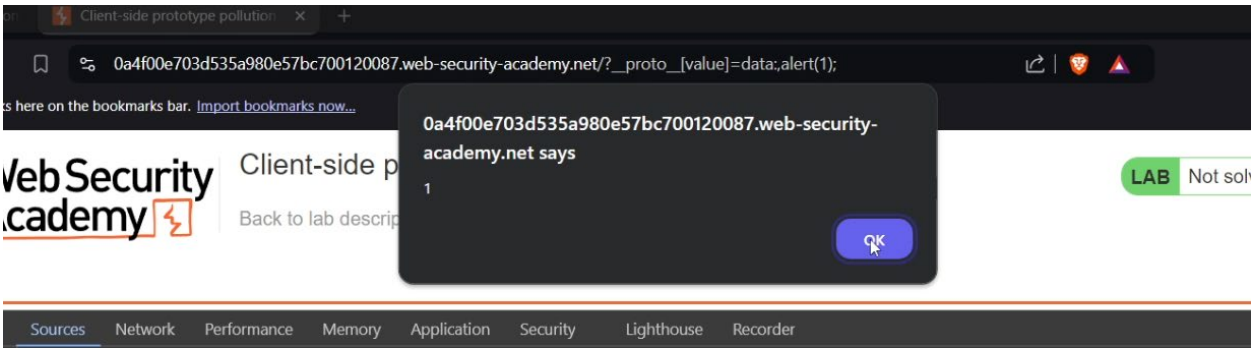
Mục tiêu: Tìm và thay đổi thuộc tính trong Object.prototype để tạo ra một event alert trên web

Dùng F12 để xem source



Quan sát thấy dòng sử dụng Object.defineProperty() để khiến cho transport_url unwritable và unconfigurable. Tuy nhiên là nó chưa định nghĩa phần value.

Thêm chuỗi /?__proto__[value]=data:,alert(1); vào câu truy vấn



Mức độ ảnh hưởng:

Cross-Site Scripting (XSS): Kẻ tấn công tiêm mã JavaScript độc hại vào ứng dụng và thực thi trên trình duyệt nạn nhân, đánh cắp thông tin nhạy cảm như cookie, thông tin đăng nhập, hoặc chiếm quyền điều khiển phiên.

Thay đổi hành vi ứng dụng: Kẻ tấn công thay đổi hành vi của các đối tượng JavaScript, gây ra lỗi logic hoặc phá hỏng các chức năng quan trọng như xác thực, phân quyền.

Khai thác API trình duyệt: Các API như localStorage, sessionStorage, và XMLHttpRequest có thể bị khai thác để đánh cắp dữ liệu hoặc thực hiện hành động bất hợp pháp trên tài khoản người dùng.

Gián đoạn và lỗi dịch vụ: Prototype Pollution có thể làm gián đoạn ứng dụng, gây lỗi hoặc dẫn đến từ chối dịch vụ (DoS).

Cách ngăn ngừa Prototype Pollution

1. **Kiểm tra và lọc dữ liệu đầu vào**:
    - Luôn kiểm tra và xác thực dữ liệu đầu vào từ người dùng, đặc biệt là khi làm việc với các đối tượng phức tạp như JSON. Không nên cho phép người dùng gửi dữ liệu có thuộc tính __proto__ hoặc constructor.
2. **Sử dụng các thư viện an toàn**:
    - Hãy chắc chắn rằng các thư viện và framework mà bạn sử dụng đã được cập nhật phiên bản mới nhất để vá các lỗ hổng liên quan đến prototype pollution.
3. **Sử dụng các phương pháp an toàn để sao chép đối tượng**:

- Thay vì sử dụng các phương thức như Object.assign() hoặc $.extend() (jQuery) để sao chép đối tượng, hãy sử dụng các phương thức an toàn hơn như Object.create(null) để tạo các đối tượng không kế thừa từ Object.prototype.

4. **Cấu hình bảo mật chặt chẽ cho các hệ thống quản lý mã**:
    - Trong trường hợp sử dụng các hệ thống template hay API để nhận dữ liệu từ người dùng, hãy kiểm tra và thiết lập cấu hình bảo mật chặt chẽ để ngăn ngừa kẻ tấn công khai thác lỗ hổng này.
5. **Sử dụng các công cụ phát hiện lỗ hổng**:
    - Sử dụng các công cụ kiểm tra bảo mật hoặc các framework bảo mật để phát hiện sớm các lỗ hổng **Prototype Pollution** trong mã nguồn hoặc thư viện.