**Broken Access Control**

**1, Access Control là gì ?**

Access Control hay kiểm soát truy cập đề cập đến một hệ thống kiểm soát quyền truy cập vào thông tin hoặc chức năng. Trong môi trường ứng dụng web , việc kiểm soát truy cập thường liên quan hay sử dụng các yếu tố sau :

- Authetication (xác thực) giúp định danh người dùng để xác định họ là ai

- Session Management (Quản lý phiên) giúp xác định các request HTTP được thưc hiện bởi ai

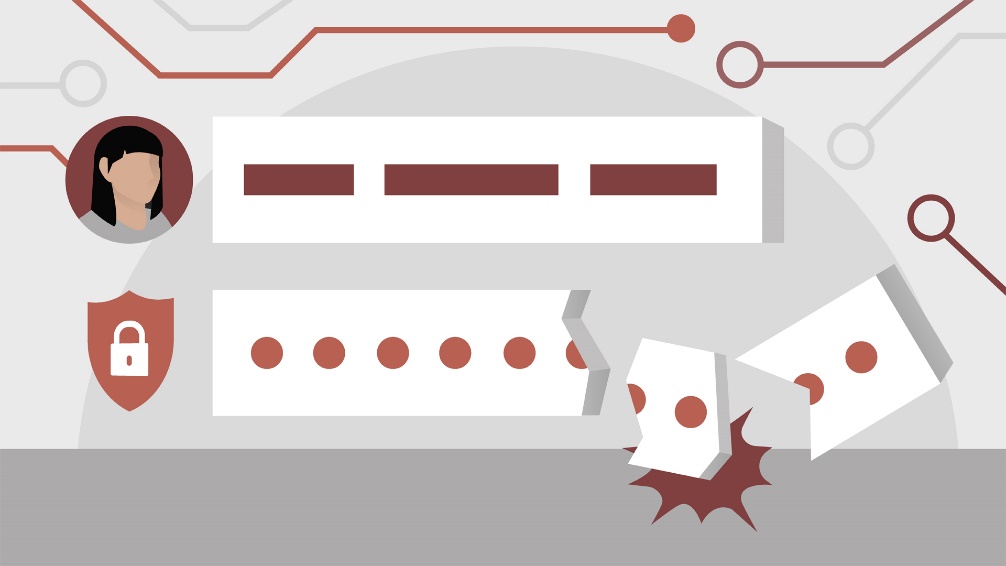
→ Từ đó các cơ chế Kiểm soát truy cập xác định liệu người dùng có được phép thực hiện hành động mà họ đang cố gắng thực hiện hay không.

**2, Từ góc độ ngừời dùng thì kiểm soát truy cập có thể chia là 3 dạng sau :**

* Kiểm soát truy cập phương dọc ([Vertical](https://portswigger.net/web-security/access-control#vertical-access-controls)): Cơ chế kiểm soát truy cập dọc thường được áp dụng cho việc quản lý các chức năng của ngừơi dùng. Phân theo vai trò và quy định các chức năng tương ứng với mỗi vai trò khác nhau. Ví dụ như Admin có full quyền thêm, sửa ,xóa còn user chỉ có 1 trong 3 quyền trên.
* Kiểm soát truy cập ngang hàng ([Horizontal](https://portswigger.net/web-security/access-control#horizontal-access-controls)) là cơ chế hạn chế quyền truy cập vào tài nguyên đối với những người dùng được phép truy cập cụ thể vào các tài nguyên đó. Giả sử như người dùng A chỉ được phép truy cập vào tài khoản của A chứ không được phép truy cập vào tài khoản của B.
* Kiểm soát truy cập phụ thuộc vào ngữ cảnh hạn chế quyền truy cập vào chức năng và tài nguyên dựa trên trạng thái của ứng dụng hoặc hoạt động của người dùng. Ví dụ: Người dùng có quyền chỉnh sửa thông tin giỏ hàng trước khi thanh toán, nhưng sau khi thanh toán xong thì sẽ không còn quyền chỉnh sửa giỏ hàng đó nữa.

**3, Vấn đề**

Access Control chứa lỗ hổng cho phép kẻ tấn công bỏ qua ủy quyền (authorization) và thực hiện các tác vụ như thể là người dùng có đặc quyền, chẳng hạn như quản trị viên (admin), nó được gọi là Broken access control .

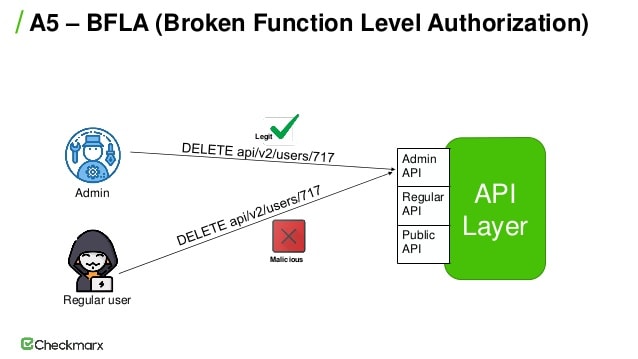


Broken access controls được đánh giá là một lỗ hổng bảo mật phổ biến và thường rất nghiêm trọng. Việc thiết kế và quản lý các phương pháp kiểm soát truy cập là một vấn đề phức tạp và cần áp dụng các ràng buộc kinh doanh, tổ chức và pháp lý một cách linh hoạt . Các quyết định thiết kế kiểm soát truy cập phải được thực hiện bởi con người chứ không phải công nghệ và khả năng xảy ra sai sót là rất cao.

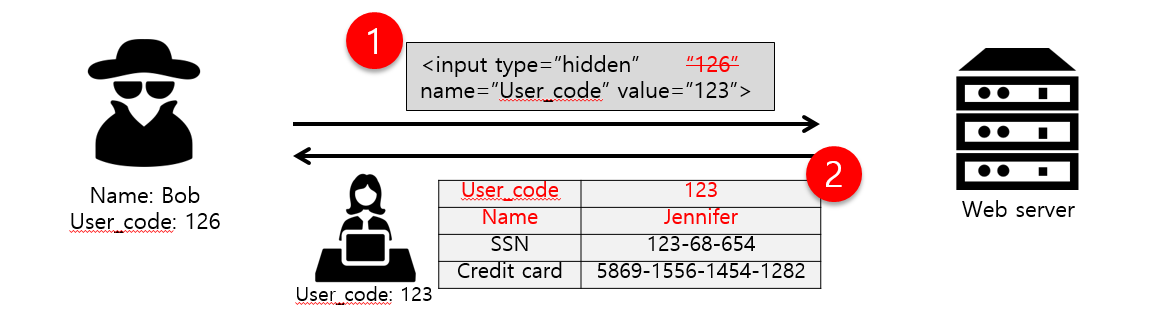
**4, Tác động**

Việc hacker phá vỡ quy tắc của các dạng kiểm soát truy cập này sẽ dẫn đến các tác động cũng khác nhau:

- Kiểm soát truy cập phương dọc lỗi sẽ dẫn đến khả năng bị leo thang đặc quyền – Privilege escalation : cho phép một người dùng bình thường có thể thực hiện các chức năng của ADMIN. Các lỗi kiểu này còn được gọi với tên là Broken function level authentication (BFLA)



- Kiểm soát ngang hàng lỗi có thể dẫn đến lỗi Tham chiếu đối tượng trực tiếp không an toàn - IDOR (INSECURE DIRECT OBJECT REFERENCE) cho phép ngừời dùng có thể tham chiếu đến các thông tin của các đối tượng khác. Các lỗi kiểu này còn được gọi là Broken object level authentication (BOLA).



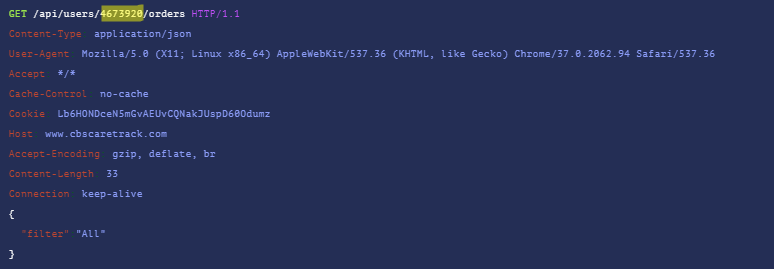
- Kiểm soát truy cập phụ thuộc vào ngữ cảnh bị lỗi thường được coi như business logic vulnerabilities nhưng nguyên nhân vẫn là do broken access control hoặc lack of authentication

**Ví dụ khai thác lỗ hổng**

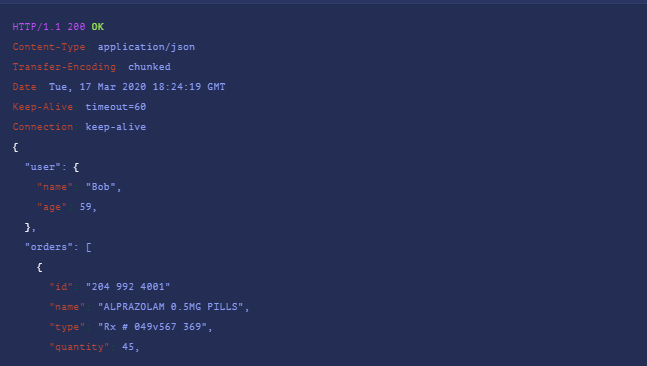
1, Broken object level authentication – IDOR

Tình huống: Ta có một ứng dụng mua đặt mua hàng hóa và có 1 user có username Bob với mã định danh userid trong ứng dụng của anh ấy là **4673920**

Để xem lịch sử đặt hàng ứng dụng sẽ tạo một request HTTP (/api/users/{userid}/orders để lấy dữ liệu . Trong đó ,giá trị {userid} sẽ được lấy từ userid của user thực hiện request.



Như hình bên trên, user Bob đang muốn xem lịch sử đặt hàng của anh ấy và kết quả sẽ trả về như hình bên dưới :



Có 2 thông tin được trả về là thông tin “user” và “order” Cùng xem phía backend đã xử lý như thế nào ?

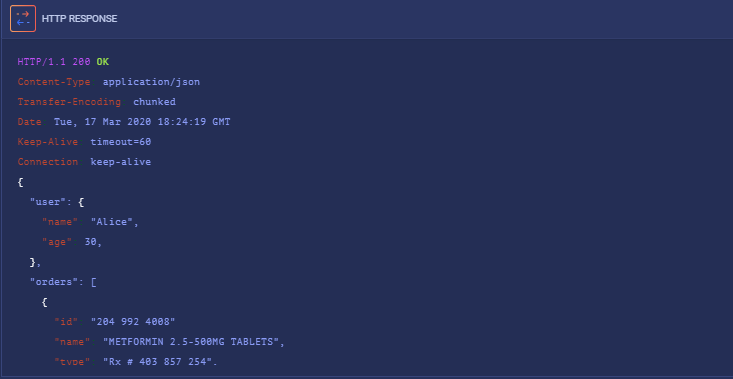


Miêu tả:

* Một phương thức getOrder() sẽ cung cấp thông tin user và thông tin đơn hàng tương ứng với user đó qua giá trị của biến userId
* Tại bước 2, sẽ gọi 2 hàm OrderManager và UserManager để thực hiện lấy thông tin qua thuộc tính get của nó. Biến userId sẽ là đầu vào của 2 hàm này. Nếu giá trị userid hợp lệ sẽ trả về về kết quả ở đoạn 3.
* Cùng xem vấn đề ở đây là gì ? chúng ta có thể thấy không có bất sự kiểm soát nào ở đây. Gía trị của userId thì có thể được kiểm soát bởi client. Điều gì sẽ xảy ra nếu Bob thay đổi giá trị userId của anh ấy thành một userId của người khác ?



Kết quả là ra ngay lịch sử đặt hàng của bạn Alice

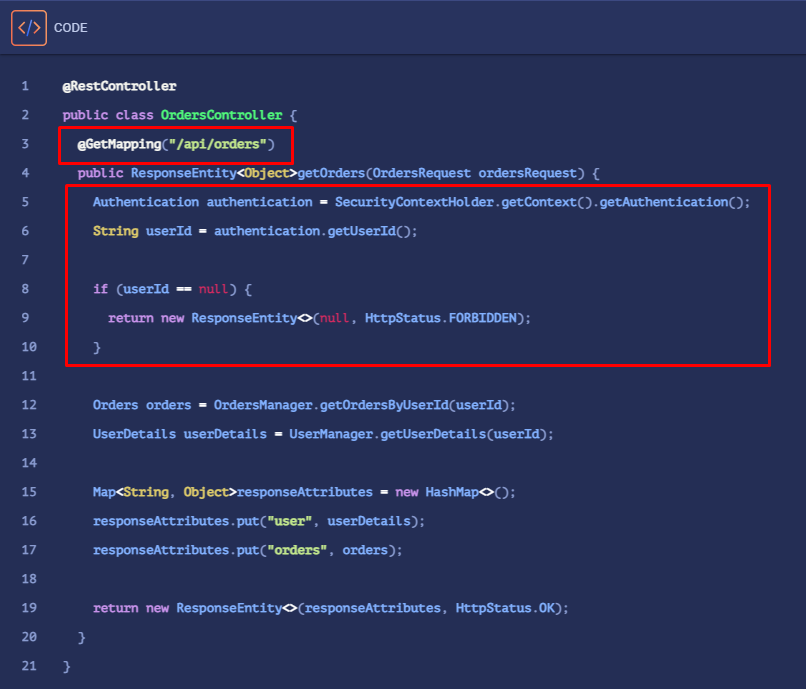


Như vậy user Bob có thể đọc xem/lấy được thông tin lịch sử đơn hàng của user Alice một cách dễ dàng và bất hợp pháp.

Hậu quả sẽ như thế nào nếu lúc này Bob là một hacker xấu xa hay resource bị lỗi broken access control lúc này chứa các thông tin bí mật như thẻ thanh toán, mật khẩu của Alice .

Cách khắc phục lỗ hổng ?

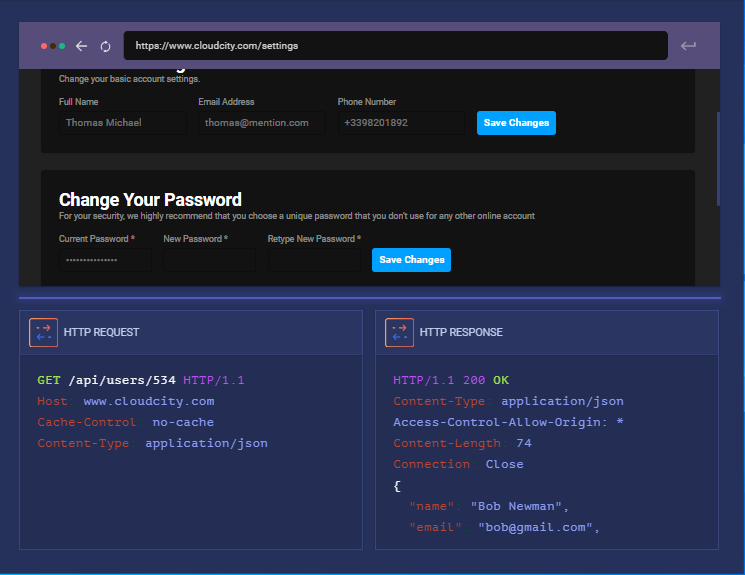
* Thứ nhất là lỗ hổng BOLA thường dựa vào giá trị định danh (uuid, userid, id, groupid … ) của người dùng để kiểm tra và khai thác do đó che được giá trị khỏi tầm mắt của hacker là điều ưu tiên đầu tiên. Do đó thay vì để các giá trị này ở trên URL thì có thể vất chúng trong cookie, Authentication token - JWT.
* Bổ sung chức năng kiểm soát truy cập chặt chẽ. Như ở hình đới đây, phương thức getOrder() đã sử dụng hàm SecurityContectHolder() để lấy giá trị Authentication – Gía trị sẽ được kiểm soát bởi server khó thể thay đổi tùy ý nên attacker sẽ không thể truy cập vào đối tượng khác.



Ví dụ 2: Broken function level authentication

Tình huống: Ta có một ứng dụng khác với user Bob và một userId mới là 534. User Bob lúc này muốn xem thông tin tài khoản trong ứng dụng của anh ấy.

Ưng dụng lúc này sẽ thực hiện một GET request gửi đến API (/api/user/534) để lấy thông tin thông qua userId .



Và lúc này Bob muốn thử kiểm tra xem API này có dính lỗi IDOR như ví dụ bên trên không :



Kết quả Bob nhận được là 403 Forbidden !!! Như vậy là không có lỗi Broken object level authentication ở đây. Không từ bỏ ở đây , Bob quyết định thử kiếm lỗi bằng cách khác đó là thay đổi Methob của request.

Chúng ta đều biết rằng với RESTFul API thì một API endpoint thì có thể có nhiều Methob được cho phép thực hiện ví dụ như GET (lấy dữ liệu), POST (gửi dữ liệu lên) , PUT (Cập nhật tài nguyên), DELETE (Xóa) ….. Điều thứ 2 nữa chúng ta cũng biết là thông thường việc xóa USER trong ứng dụng chỉ có ADMIN – quản trị viên mới có quyền này. Thế thì làm sao để một user bình thường Bob có thể làm được điều này ?

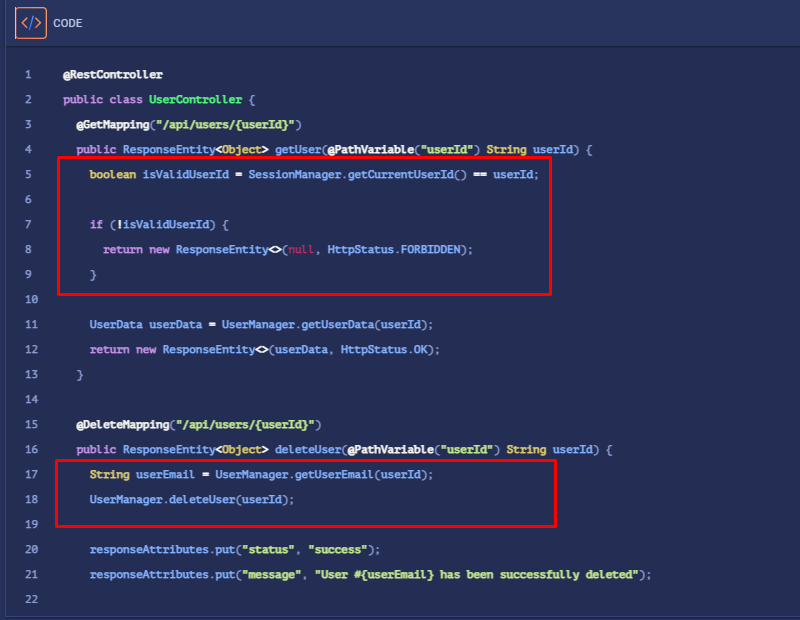


BOOM!!! Tài khoản Alice đã bị xóa. Như vậy chỉ cần thay đổi Methob của request là đã giúp Bob thực hiện chức năng xóa user của ADMIN.

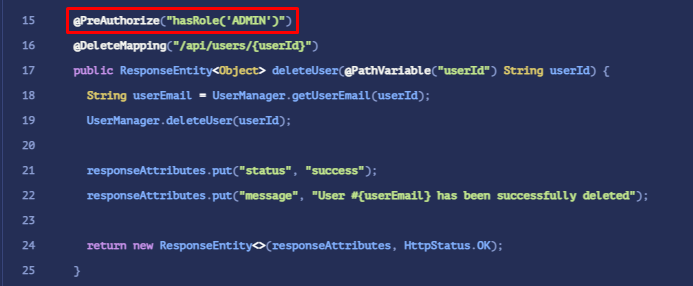
Và đây gọi là Broken function level Authentication hay Broken Acess Control.

Bây giờ cùng lướt xem phía BE đã xử lý như thế nào mà dẫn đến vấn đề này .

Miêu tả: Điều dễ thấy đó là ở chức năng getUser() thì có hàm kiểm tra userid được lấy ở URL với userId trong session cookie hay nói cách khác là có liên kết/đồng bộ userId với phiên người dùng. Còn ở chức năng deleteUser thì không có hàm kiểm tra nào được triển khai. Chỉ cần một userId hợp lệ là có thể xóa, lỗi thì không khác gì BOLA nhưng bản chất nó là BFLA



Cách khắc phục như thế nào ? Đơn giản là bổ xung thêm kiểm tra người dùng đó có vai trò ‘ADMIN’ trước khi thực hiện hay không .



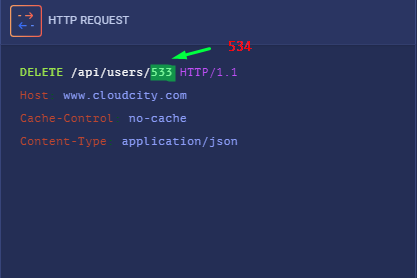
Các câu hỏi :

1, Trong ví dụ 2 , giá trị userid đã được thay đổi thành giá trị của user muốn xóa thì endpoint này có coi là có cả lỗi Broken Objection level Authentication hay không ?

* Câu trả lời là không. Nó chỉ tính là BFLA . Bởi vì chức năng bị dính lỗi chứ không phải là object đó bị lỗi kiểm soát truy cập.

2, Thế có trường hợp nào thì ví dụ 2 thì được coi là BOLA không ?

* Câu trả lời là có. Ví dụ như user Bob và user Alice đều có chức năng xóa tài khoản nhưng theo cơ chế kiểm soát thì chỉ được phép xóa (delete/disable) tài khoản của chính mình. Gỉa sử endpoint này có dĩnh lỗi BAC cho phép Bob xóa tài khoản của Alice bằng cách thay giá trị của userId của Bob thành Alice.



Thì lúc này kiểm soát truy cập đến đối tượng Alice đã bị phá vỡ -> BOLA