# Một số lỗi thường gặp

# Access Control Flaw

# Authentication Flaws

# Sql Injection

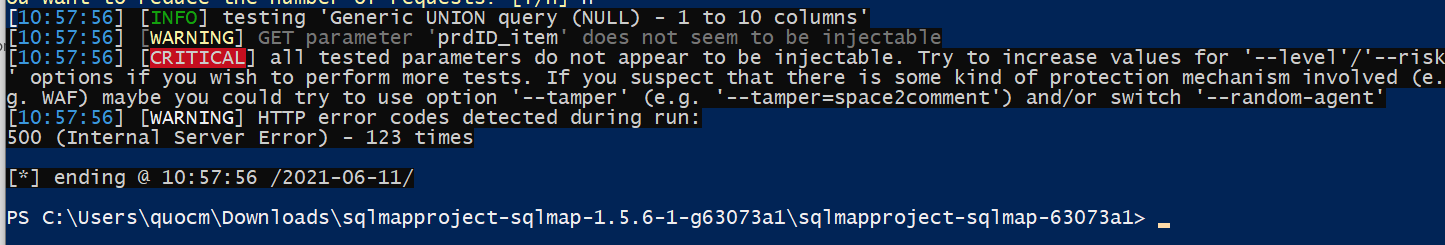
* SQL (Structured Query Language) hay ngôn ngữ truy vấn mang tính cấu trúc, là một loại ngôn ngữ máy tính phổ biến để tạo, sửa, và lấy dữ liệu từ một hệ quản trị cơ sở dữ liệu quan hệ.
* SQL Injection hoặc SQLi xảy ra khi có một lỗ hổng database-backed trên trang web cho phép Attacker truy vấn hoặc tấn công database của trang web. Các cuộc tấn công SQLi thường được nhắc đến vì mức độ nguy hiểm của chúng. Attacker có thể lấy toàn bộ dữ liệu trên website, thậm chí có thể chỉnh sửa chúng, thao túng và thậm chí có thể tự tạo một admin đăng nhập vào cơ sở dữ liệu.

**Nguyên nhân :**

* Dữ liệu đầu vào từ người dùng hoặc từ các nguồn khác không được kiểm tra hoặc kiểm tra không kỹ lưỡng
* Ứng dụng sử dụng các câu lệnh SQL động, trong đó dữ liệu được kết nối với mã SQL gốc để tạo câu lệnh SQL hoàn chỉnh

**Nhận xét về project nhóm :**

Trong project nhóm Owasp Zap không tìm được bất kỳ lỗi SQL Injection nào.Nên nhóm sử dụng ‘sqlmap’ để tiếp tục quét lỗ hổng SQL của web và kết quả cũng tương tự Owasp không lỗi SQL injection nào được phát hiện.



Sqlmap báo rằng không thể thực hiện injection :



**Cách giải quyết :**

Trang web của nhóm không có lỗ hỗng SQL Injection vì :

* Sử dụng Hibernate
* Sử dụng Prepared Statements.

Hibernate là một thư viện ORM (Object Relational Mapping) mã nguồn mở giúp lập trình viên viết ứng dụng Java có thể map các objects (pojo) với hệ quản trị cơ sở dữ liệu quan hệ, và hỗ trợ thực hiện các khái niệm lập trình hướng đối tượng với cớ dữ liệu quan hệ.

Hibernate cung cấp các câu lệnh truy vấn tương tự SQL, HQL của Hibernate hỗ trợ đầy đủ các truy vấn đa hình như, HQL “hiểu” các khái niệm như kế thừa (inheritance), đa hình (polymorphysm), và liên kết (association).

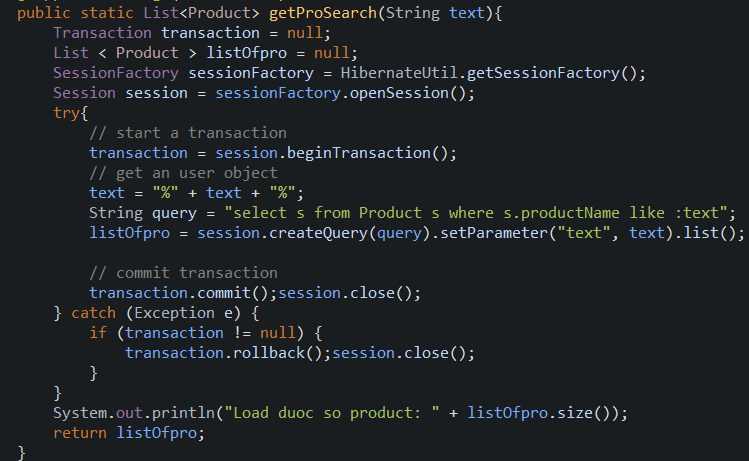
Code sử dụng Hibernate là độc lập với hệ quản trị cơ sở dữ liệu, nghĩa là ta không cần thay đổi câu lệnh HQL khi ta chuyển từ hệ quản trị CSDL MySQL sang Oracle, hay các hệ quản trị CSDL khác… Do đó rất dễ để ta thay đổi CSDL quan hệ, đơn giản bằng cách thay đổi thông tin cấu hình hệ quản trị CSDL trong file cấu hình.

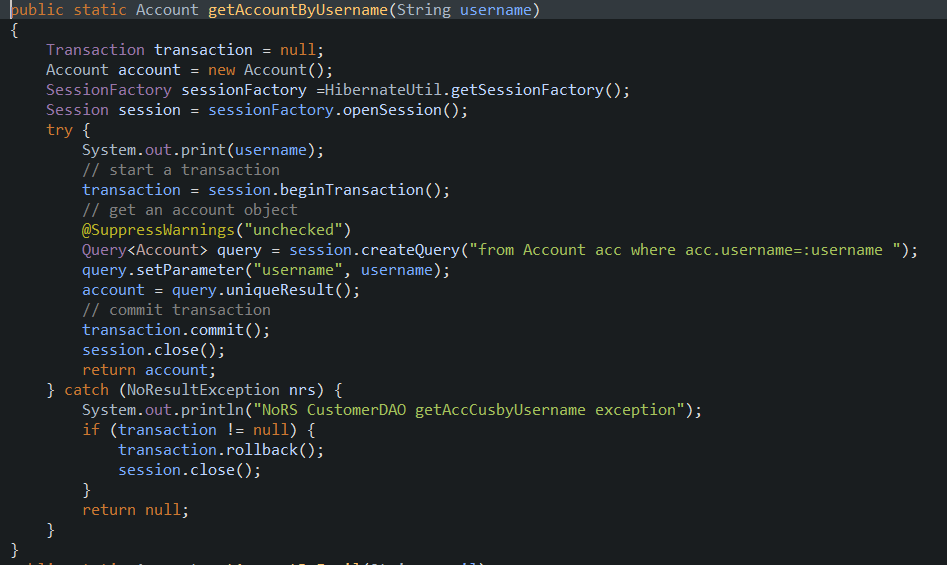
Hibernate không làm website miễn nhiễm với SQL Injection nhưng sử dụng nó với Prepared Statements thì sẽ có thể ngăn chặn SQL Injection.

Prepared Statements bảo rằng kẻ tấn công không thể thay đổi mục đích của một truy vấn, ngay cả khi các lệnh SQL được kẻ tấn công chèn vào. Ví dụ, nếu kẻ tấn công nhập userID như sau (tom 'or' 1 '=' 1), truy vấn được tham số hóa sẽ không dễ bị tấn công và thay vào đó sẽ tìm kiếm tên người dùng khớp với toàn bộ chuỗi (tom 'or' 1 '=' 1).

Prepared Statements được sử dụng trong Hibernate bằng cách sử dụng hàm createQuery() và setParameter().

**Một số ví dụ sử dụng trong project :**





# Cross-Site Scripting

# Một số lỗi Owasp Zap quét được

# X-Frame-Options Header Not Set, X-Content-Type-Options Header Missing và Web Browser XSS Protection Not Enabled(3 lỗi header)

## Nguyên nhân

X-Frame-Options là một header trong những HTTP Security Headers.Việc không set giá trị cho header này sẽ dẫn đến lỗi bảo mật X-Frame-Options Header Not Set.

X-Content-Type-Options là một header trong những HTTP Security Headers . Việc bị thiếu cho phép trình duyệt thực hiện sniffing kiểu MIME khi tiêu đề Content-Type không được đặt hoặc giá trị của nó có vẻ không phù hợp dẫn đến lỗi bảo mật X-Content-Type-Options Header Missing.

X-XSS-Protection là 1 trong số các HTTP Header.X-XSS-Protection là một tính năng của trình duyệt Chrome và Internet Explorer, được thiết kế để chống lại reflected XSS attacks .Việc không thiết lập header này dẫn đến lỗi bảo mật Web Browser XSS Protection Not Enabled

## Mô tả

X-Frame-Options HTTP có thể được sử dụng để cho biết liệu trình duyệt có được phép hiển thị một trang trong <frame>, <iframe>, <embed> hoặc <object> hay không. Các trang web có thể sử dụng điều này để tránh các cuộc tấn công click-jacking , bằng cách đảm bảo rằng nội dung web không được nhúng bởi các trang web khác.

X-Frame-Options Header có 3 giá trị:

* X-Frame-Options: DENY (không cho đặt trang vào iframe).
* X-Frame-Options: SAMEORIGIN (Trang chỉ có thể được hiển thị ở trong một iframe tạo bởi ai đó ở cùng một nguồn (same origin) với nó).
* X-Frame-Options: ALLOW-FROM https://example.com/ (Chỉ định đường dẫn url có thể đặt trang này vào iframe).

X-Frame-Options Header dùng để ngăn chặn clickjacking ( một thuật ngữ diễn tả việc lừa người sử dụng click chuột vào một liên kết nhìn bề ngoài có vẻ “trong sạch” trong các trang web, tuy nhiên qua cú click chuột đó hacker có thể lấy được các thông tin bí mật của người sử dụng hay kiểm soát máy tính của họ).

X-Content-Type-Options nhằm để chỉ ra rằng các loại MIME(Multipurpose Internet Mail Extensions) được trong các Content-type Header không được thay đổi và phải tuân theo.

Khi server trả lời request nó phải thiết lập content-type cho response để trình duyệt có thể biết được nó đang nhận được gì.Tuy nhiên một số developer vẫn chưa biết cách thiết lập content-type và vì thế trình duyệt sẽ thực hiện sniffing kiểu MIME để xem nội dung truyền đến là gì và hiển thị ra.

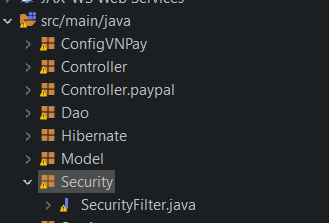
X-XSS Protection được thiết kế để chống lại reflected” XSS attacks - khi mà kẻ tấn công gửi mã độc hại như là một phần của request (reflected nghĩa là những gì request lên sẽ được phản ánh lại qua nội dung trang web).Có các giá trị sau :

* X-XSS-Protection: 0 : tắt chế độ này
* X-XSS-Protection: 1 : lọc bỏ các đoạn script ở trong request, nhưng vẫn hiển thị trang web
* X-XSS-Protection: 1; mode=block : khi được kích hoạt sẽ vô hiệu hoàn toàn việc hiển thị trang web

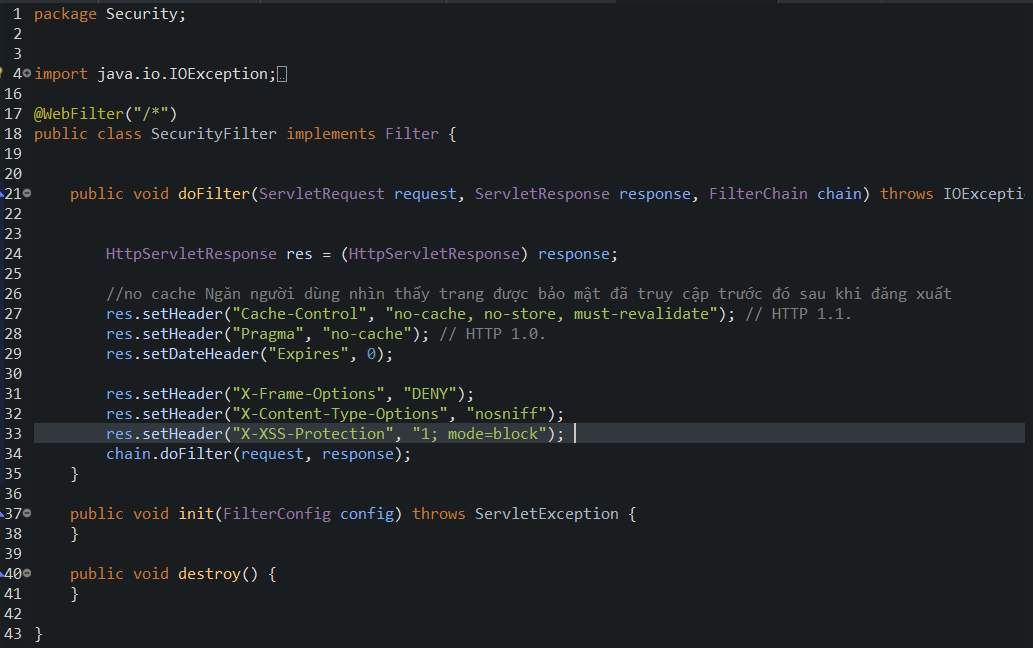
## Cách giải quyết

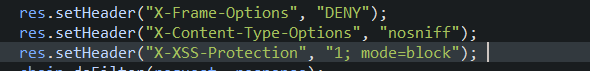
Để giải quyết vấn đề này ta sẽ implement 1 class filter cho toàn bộ website như sau:

1/Tạo 1 package Security và 1 class Filter trong package với tên là SecurityFilter



2/Cài đặt class SecurityFilter như sau

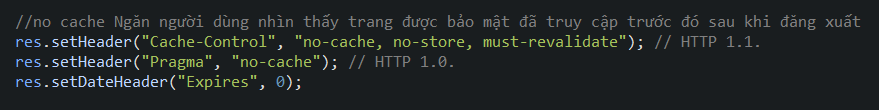




res.setHeader("X-Frame-Options", "DENY"); 🡪 để set X-Frame-Options cho response.

res.setHeader("X-Content-Type-Options", "nosniff"); 🡪 dùng để thêm x-content-type-option header.

res.setHeader("X-XSS-Protection", "1; mode=block"); 🡪 dùng để thêm x-xss-protection header.



Dùng thiết lập cache header nhằm ngăn người dùng nhìn thấy trang được bảo mật đã truy cập trước đó sau khi đăng xuất.

# Cross-Domain JavaScript Source File Inclusion

## Nguyên nhân

Cross-Domain JavaScript Source File Inclusion là cảnh báo bảo mật có thể ảnh hưởng đến ứng dụng web chạy một hoặc nhiều tệp Javascript từ miền của bên thứ ba.

## Mô tả

Nếu bên thứ ba cố ý hoặc vô ý giữ một nội dung độc hại, nội dung đó có thể được thêm vào và thực thi trên ứng dụng web của nạn nhân.

Khi Javascript bên ngoài không được xác thực. Nó có thể dẫn đến rò rỉ dữ liệu người dùng.

## Cách giải quyết

Đảm bảo bên thứ ba cung cấp các file javascript là đáng tin cậy.

Trang web sử dụng các file javascript như sau :



Google, CloudFlare và FontAwesome đều là những nguồn tin cậy.

# Cookie Without SameSite Attribute

## Nguyên nhân

Một cookie đã được đặt mà không có thuộc tính SameSite, có nghĩa là cookie có thể được gửi đi do yêu cầu "cross-site". Điều này dẫn đến lỗi bảo mật Cookie Without SameSite Attribute

## Mô tả

Cookie là một trong những phương pháp có sẵn bởi trình duyệt, dùng để lưu thông tin ngay trên client. Thông tin này được sử dụng phục vụ cho các trang Web nhằm tối ưu trải nghiệm cho cá nhân người dùng đang ở trên trình duyệt.

SameSite Cookie là một thuộc tính thêm vào trong Cooke để yêu cầu chủ sở hữu trang web ghi rõ ràng nhãn cookie của Web khác, nhờ đó chỉ có thể chia sẻ Cookie với các trang web này. Như vậy một Website lừa đảo không thể giả mạo người dùng vì không thể lấy được Cookie.

Bằng Samesite Cookie, việc chống tấn công CSFR sẽ dễ dàng và hiiệu quả hơn. Trước tiên người làm hệ thống sẽ giảm chi phí do không phải lập trình bổ sung tính năng tạo và tương tác thông qua token ở cả server và client. Thứ hai là hiệu năng của hệ thống được nâng cao khi không phải sinh và đối chiếu token.

## Cách giải quyết

**SameSite** là một thuộc tính của Set-Cookie trong HTTP response header giúp bạn ngăn cấm một site khác truy cập vào cookie hoặc cho phép trang chụ thể. SameSite nhận giá trị thuộc 1 trong 3 (None, Strict, LaX)

**Đặt** **SameSite = Strict**

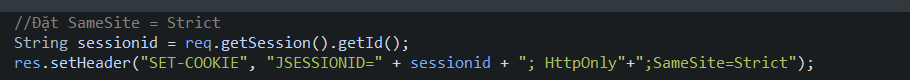
Đúng như tên gọi của nó, giá trị của Strict phòng ngừa các web giả mạo rất mạnh mẽ. Trình duyệt sẽ ngăn dữ liệu cookie chuyển giữa các tên miền chéo khác trong mọi trường hợp. Phần lớn các trang web dịch vụ tài chính, ngân hàng, tiền điện tử sẽ chọn đặt giá trị SameSite của họ là  Strict. SameSite = Strict sẽ lấy đi sự tiện dụng mà người dùng đã quen với khả năng chia sẻ thông tin của người dùng giữa các site, nhưng an toàn là trên hết.

Cài đặt như sau :

Tại lớp SecurityFilter



Thêm đoạn code sau :



String sessionid = req.getSession().getId(); 🡪 lấy session ID hiện tại

res.setHeader("SET-COOKIE", "JSESSIONID=" + sessionid + "; HttpOnly"+";SameSite=Strict");🡪thiết lập SameSite = Strict.

# Absence of Anti-CSRF Tokens

# Loosely Scoped Cookie

## Nguyên nhân

Cookie có thể được xác định phạm vi theo miền(domain) hoặc đường dẫn(path).Không thiết lập hoặc thiết lập phạm vi đến nguồn không tin cậy sẽ dẫn đến lỗi bảo mật Loosely Scoped Cookie

## Mô tả

Một cookie với kích thước 4KB, bao gồm 7 thành phần chính:  
1. Name  
2. Value  
3. Expires (hạn sử dụng)  
4. Path (đường dẫn đến nơi cookie có hiệu lực, “/” có nghĩa là cookie có giá trị ở bất cứ đường dẫn nào)  
5. Domain  
6. Secure  
7. HttpOnly  
Hai thành phần đầu tiên (name và value) yêu cầu bắt buộc phải có.

Thuộc tính miền(domain)

Thuộc tính Domain chỉ định máy chủ nào được phép nhận cookie. Nếu không được chỉ định, nó sẽ mặc định cho cùng một máy chủ đặt cookie, không bao gồm các miền phụ. Nếu Miền được chỉ định, thì các miền phụ luôn được bao gồm. Do đó, việc chỉ định Tên miền sẽ ít hạn chế hơn việc bỏ qua nó. Tuy nhiên, nó có thể hữu ích khi các miền phụ cần chia sẻ thông tin về người dùng.

Ví dụ: nếu Domain = mozilla.org được đặt, thì cookie sẽ có sẵn trên các tên miền phụ như developer.mozilla.org.

Thuộc tính đường dẫn(path)

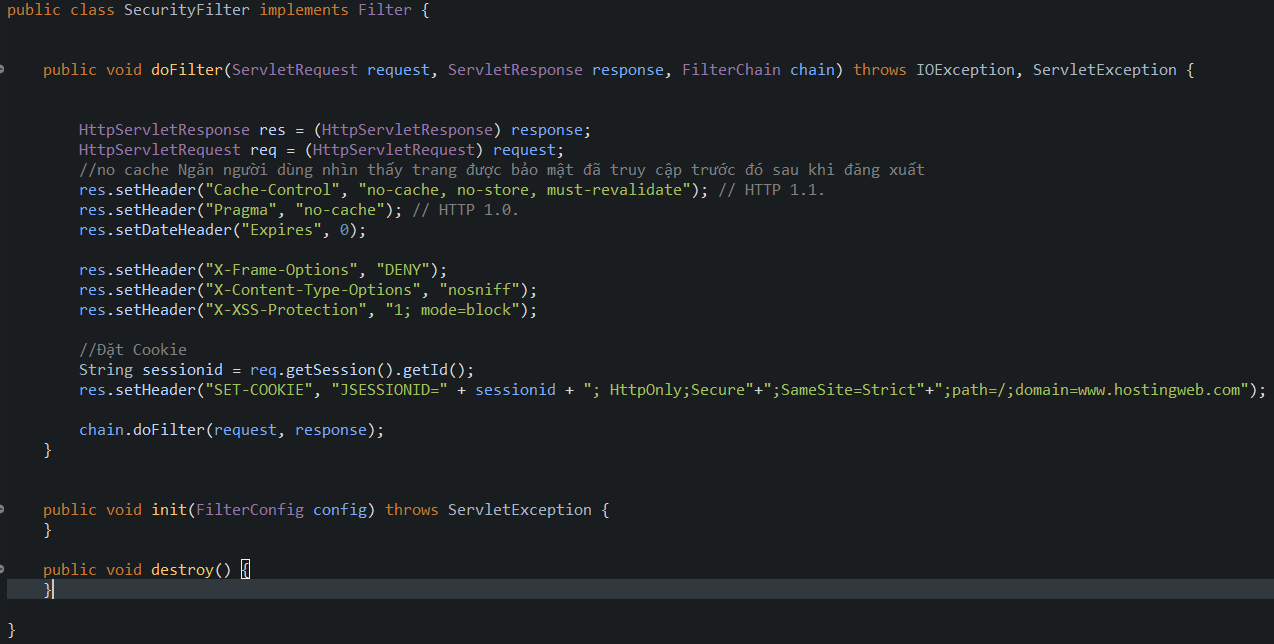
Thuộc tính Path cho biết một đường dẫn URL phải tồn tại trong URL được yêu cầu để gửi tiêu đề Cookie. Ký tự% x2F ("/") được coi là dấu phân tách thư mục và các thư mục con cũng khớp với nhau.

## Cách giải quyết

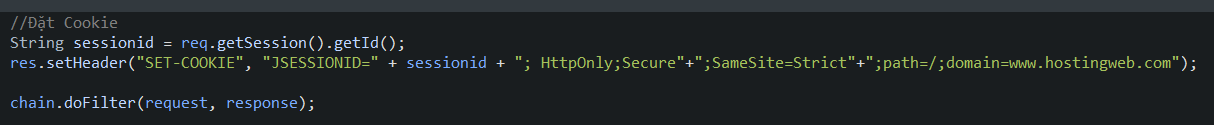
Luôn xác định phạm vi cookie đến trang miền đáng tin cậy.

Trong project nhóm làm như sau :

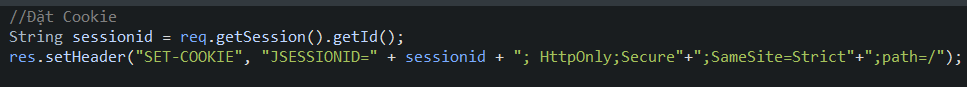
Thiết lập lớp Filter như sau



Đối với khi deploy lên hosting



Đối với khi chạy localhost



Tại localhost ta phải bỏ qua thuộc tính domain và để trình duyệt tự chọn, nếu sử dụng sẽ có thể làm trình duyệt bỏ qua header set-cookie.

# Timestamp Disclosure - Unix

## Nguyên nhân

Trang web có sử dụng timestamp sẽ nhận được cảnh báo này.

## Mô tả

Dấu thời gian do máy chủ ứng dụng hoặc máy chủ web tiết lộ có thể được sử dụng để truy xuất thông tin nhạy cảm khác, ví dụ: khi được sử dụng dưới dạng SALT hoặc mã thông báo(token) trong quá trình xác thực hoặc mã hóa.

## Cách giải quyết

Kiểm tra trang web rằng dữ liệu dấu thời gian không nhạy cảm và không thể tổng hợp dữ liệu để tiết lộ các mẫu có thể khai thác.

Trong project có sử dụng timestamp nhưng nó không sử dụng để xác thực hay truy xuất và cũng không tiết lộ thông tin bảo mật nào nên có thể bỏ qua lỗi bảo mật này.

# Session ID in URL Rewrite

## Nguyên nhân

URL Rewrite được sử dụng để theo dõi ID phiên của người dùng. ID phiên có thể được tiết lộ thông qua tiêu đề giới thiệu trên nhiều trang web. Ngoài ra, ID phiên có thể được lưu trữ trong lịch sử trình duyệt hoặc nhật ký máy chủ.Dẫn tới lỗi bảo mật Session ID in URL Rewrite

## Mô tả

URL Rewrite là một quá trình sửa đổi URL trong khi tải một trang. Nó được sử dụng để theo dõi các phiên của người dùng bằng cách giúp máy chủ có thể nhận ra các yêu cầu từ một phiên. Có nhiều trường hợp trong đó điều này có thể dẫn đến session fixation. Session fixation là một cuộc tấn công trong đó kẻ tấn công có thể tiếp quản phiên và thực hiện các cuộc tấn công nhờ vào session đã chiếm được và xác thực.

## Cách giải quyết

Đối với nội dung an toàn, hãy đặt ID phiên vào cookie. Để an toàn hơn nữa, hãy xem xét sử dụng kết hợp cookie và ghi lại URL.

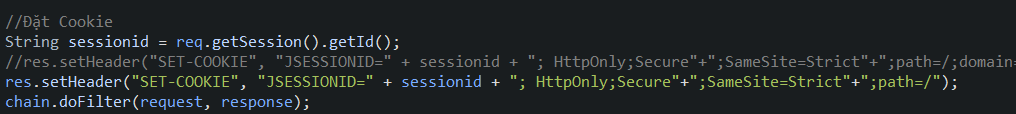
Đảm bảo sử dụng HTTPS trên trang web của bạn.

Lưu trữ ID phiên trong cookie

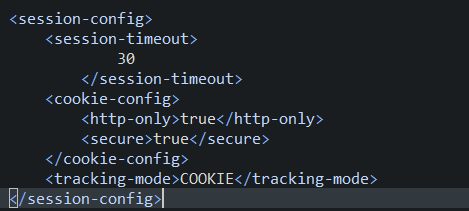
Để bảo mật hơn nữa, hãy sử dụng kết hợp cookie và ghi lại URL.

Trong project nhóm giải quyết như sau :

Ghi phiên vào cookie



Thêm đoạn xml sau vào web.xml



Với



dùng để tắt URL rewriting và hướng dẫn trình duyệt sử dụng chính sách "Chỉ cookie" để duy trì phiên HTTP.