06

Evitando ataques do tipo CSRF

Transcrição

A aplicação aparenta estar funcionando normalmente, mas o que acontece se tentarmos realizar uma compra ou tentarmos pelo menos adicionar um produto ao carrinho?

O que antes funcionava, agora não funciona. Adicionar os componentes de segurança do *Spring* fez com que o formulário de adição de produtos no carrinho parasse de funcionar por algum motivo. O que será esse **CSRF**?

CSRF é uma sigla que representa a frase **Cross Site Request Forgery**, uma técnica de ataque a sites muito comum na web. A técnica representa o cenário de que um outro site está enviando dados para nossa aplicação, em vez do usuário diretamente. Geralmente, acontece com páginas clonadas, uma página falsa é apresentada ao usuário e este sem saber submete seus dados que por sua vez podem ser enviados ao servidor original ou podem ser feitas cópias em um servidor falso para posteriormente ter o uso indevido.

Por ser uma técnica de ataque muito comum e também muito perigosa, diversos frameworks de várias plataformas, já possuem recursos prontos para que usemos afim de evitar tais ocorrências. Com o *Spring* não é diferente.

A solução mais simples é criar um novo `input` no formulário com o valor `${_csrf.parameterName}` no atributo `name` e `${_csrf.token}`. Em nossa página de `detalhes.jsp` teriamos:

```
<form action="<c:url value='/carrinho/add' />" method="post" class="container">
    <ul id="variants" class="clearfix">
        <input type="hidden" name="produtoId" value="${produto.id}" />
        <c:forEach items="${produto.precos}" var="preco">
          <li class="buy-option">
           [...]
          </li>
        </c:forEach>
    </ul>
    <button type="submit" class="submit-image icon-basket-alt" alt="Compre Agora" title="Compre

    <input type="hidden" name="${_csrf.parameterName }" value="${_csrf.token }" />

  </form>
```

Com este código, o *Spring* irá gerar o `name` e o `value` de forma que ele saiba se o **POST** veio do formulário da página ou de algum outro lugar. O `_csrf.token` nada mais gera do que uma sequência de números verificadores que se alterados tornam-se inválidos gerando o seguinte erro:

Mas convenhamos de que podemos facilmente esquecer de criar o `input` de *token* manualmente toda vez que precisarmos criar um formulário em nossa aplicação. Para que evitemos esses casos de esquecimento, podemos usar uma segunda forma de criação de formulários que já vem com este campo configurado. Usando as tags de formulários do próprio *Spring*. Assim só precisamos alterar a tag `form` do formulário de compras da página `detalhes.jsp` .

```
<form:form servletRelativeAction="/carrinho/add" method="post" cssClass="container">
    <ul id="variants" class="clearfix">
        <input type="hidden" name="produtoId" value="${produto.id}" />
        <c:forEach items="${produto.precos}" var="preco">
          [...]
        </c:forEach>
    </ul>
    <button type="submit" class="submit-image icon-basket-alt" alt="Compre Agora" title="Compre
  </form:form>
```

Dessa forma não precisamos nos preocupar em criar um campo *CSRF* manualmente toda vez que formos adicionar um formulário em alguma página, o *Spring* já fará isso automaticamente. Perceba que utilizamos uma forma alternativa de criar o `action` do formulário através do `servletRelativeAction` ao invés do `mvcUrl` .