02

Protegendo o servidor

Capítulo 5 - Protegendo o servidor

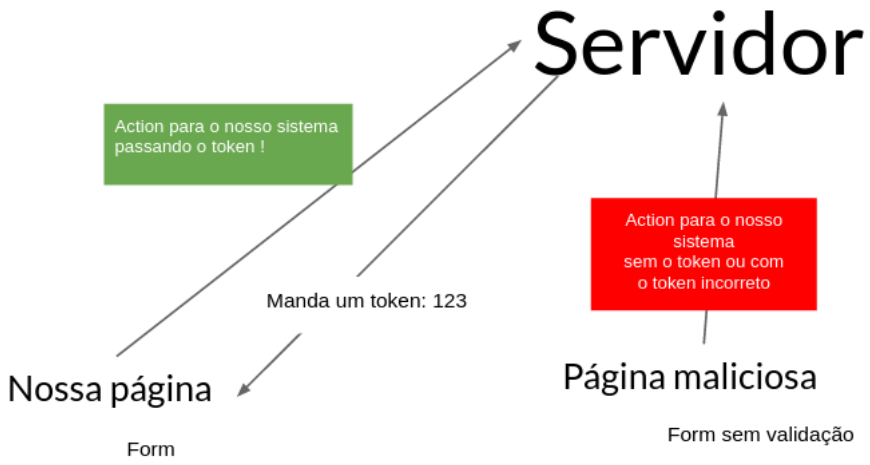
Nossa aplicação já possui diversos outras features agora, inclusive está funcionando em outros idiomas. Vamos agora falar se outra coisa importante.

A nossa aplicação feita através do Cake está rodando no servidor e, se vamos cadastrar um produto utilizamos o formulário, preenchemos seus campos, uma validação é executada, etc e, no fim, o formulário tem uma *Action* para o sistema.

Porém imaginemos a seguinte situação: um usuário malicioso, o qual não possui login e senha, sabe qual é essa *action*, qual script deve ser executado. Ele cria uma página maliciosa com um formulário idêntico ao nosso, mas sem validação e com a mesma *action* para o servidor.

O servidor receberá essa requisição da página maliciosa e irá aceitar, pois são os mesmos dados do nosso formulário. Serão cadastrados produtos que não deveriam, serão feitas transferências com contas de terceiros, etc. Precisamos de uma solução para que o servidor garanta que os dados recebidos vieram da página que ele gerou.

Fazemos esse controle utilizando *Tokens*. O servidor envia um Token para a página que, por sua vez, ao completar a *Action*, envia de volta o mesmo Token. O servidor verifica se o Token recebido é o mesmo que ele anteriormente havia gerado. Já a página maliciosa não receberá esse Token, logo vai gerar uma Action que chegará ao servidor sem Token e não será processado.



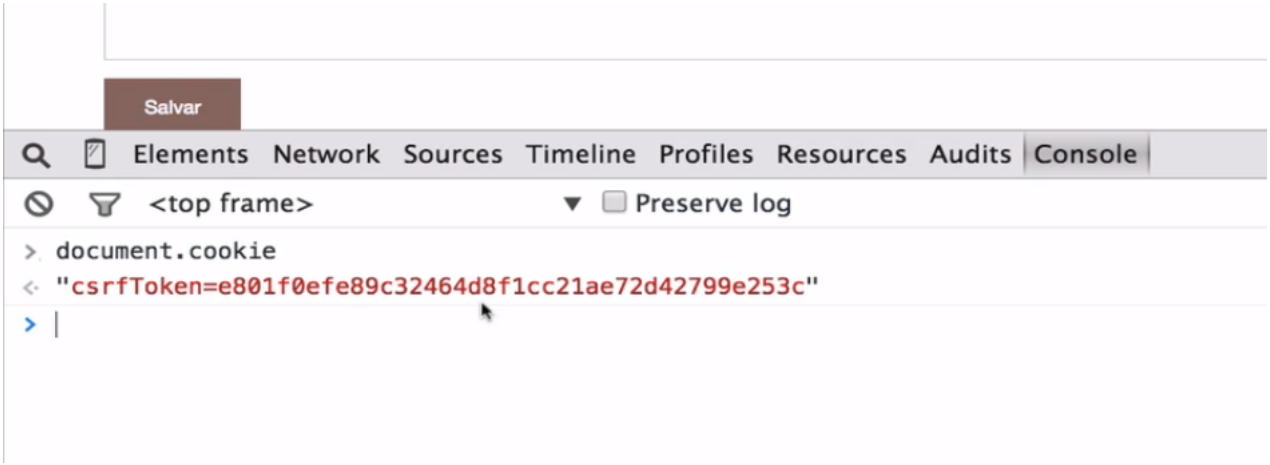
Dessa forma o servidor consegue diferenciar quais requisições estão chegando da página própria do sistema ou quais foram forjadas. A esse tipo de ataque damos o nome de **CSRF**, Cross Site Request Forgery.

Vamos implementar, então, esta feature de segurança. Queremos garantir que o ProdutosController utilize o nosso Token. No Cake PHP, para trabalharmos com CSRF, basta carregarmos um componente:

```
public function initialize() {
    ...

    $this->loadComponent('Csrf');
}
```

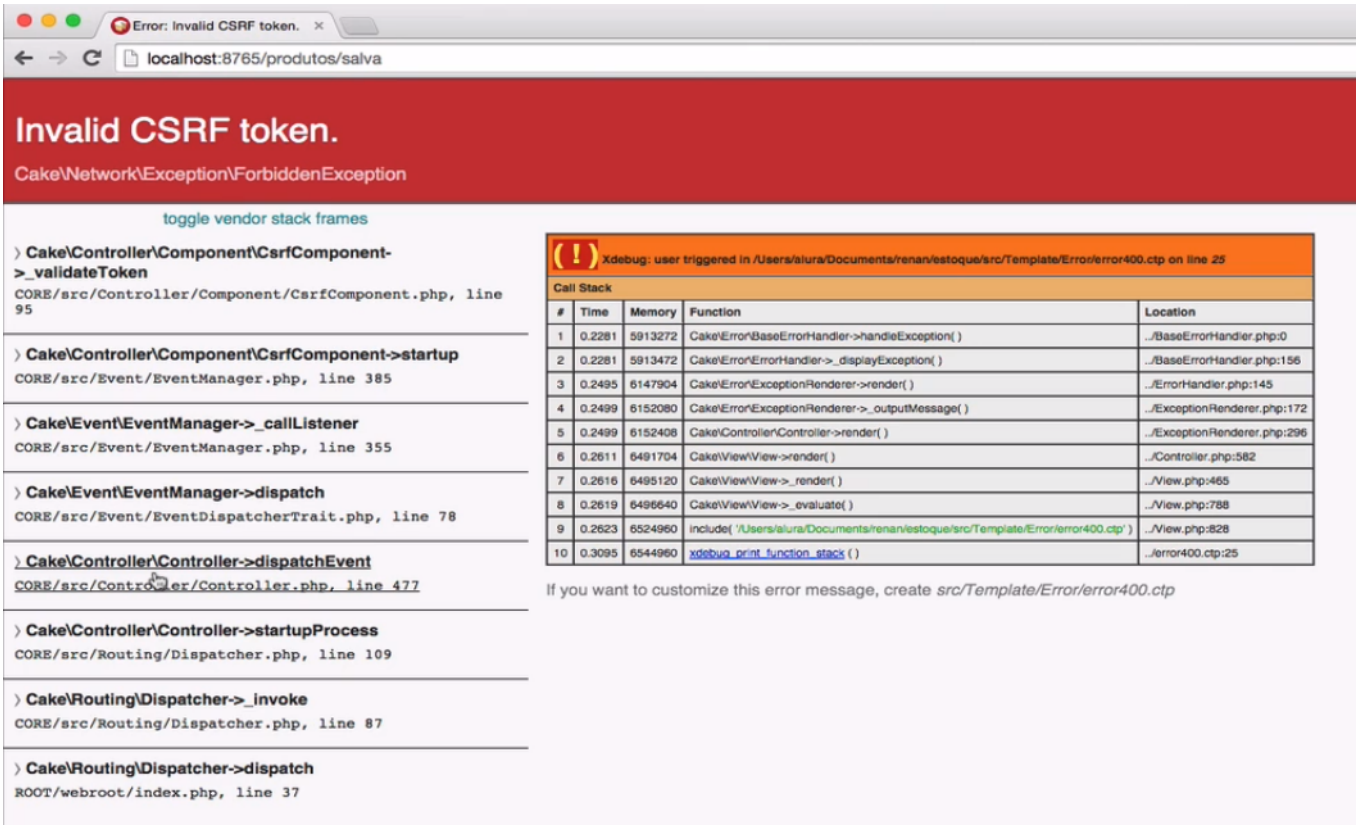
Agora, vamos salvar um novo produto dando uma olhada no console para visualizarmos os cookies que teremos dentro da aplicação. Para isso usamos Javascript:



Assim visualizamos o Token que garante a requisição. Vamos adicionar um Token inválido para ver o que acontece. Fazemos:

```
document.cookie = "csrfToken=[Token inválido]
```

Adicionando um novo produto, o navegador nos mostra o seguinte erro:



Dessa maneira protegemos nossas páginas contra CSRF.