**Protegendo o acesso as páginas, refatorando e extraindo a lógica**

Legal, fizemos um formulário onde o usuário coloca o usuário e senha e mostramos se ele fez login com sucesso (através de um cookie) ou não. Mas queremos mostrar o formulário de produto somente se ele estiver logado!

Como fazer isso? Verificando se o usuário está logado antes de mostrar o formulário. Então vamos no nosso**produto-formulario.php** e verificamos se o usuário não tem o cookie de usuário logado:

if(!isset($\_COOKIE["usuario\_logado"])) {

Header("Location: index.php?falhaDeSeguranca=true");

die();

}

E no nosso **index.php** devemos verificar se o parâmetro GET está setado:

<?php

if(isset($\_GET["falhaDeSeguranca"])) {

?>

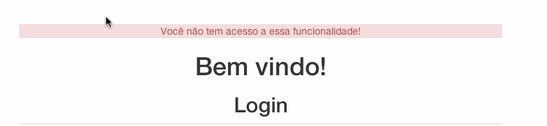
<p class="alert-danger">Você não tem acesso a esta funcionalidade!</p>

<?php

}

?>

Perfeito, deslogados tentamos acessar o formulário de produto e recebemos a mensagem:



Mas tem algo de estranho. Protegemos o nosso formulário, mas o que alguém maldoso poderia fazer? Poderia criar um arquivo html que tem um formulário igual ao de adicionar, mas salvo na própria máquina. Ele pode enviar esse formulário para a URI **adiciona-produto.php**. Ele pode acessar o html pois é um html qualquer na máquina dele, e quando fizer o post, ele vai para nosso arquivo, que adiciona. A falha de segurança é que o **adiciona-produto** não verificou se o usuário está logado, e é muito importante proteger a lógica no **adiciona-produto.php**:

if(!isset($\_COOKIE["usuario\_logado"])) {

Header("Location: index.php?falhaDeSeguranca=true");

die();

}

Agora sim, se ele acessar o formulário ou o adiciona-produto, estamos protegidos. Claro, ficou um copy e paste meio feinho, o que podemos fazer? Extrair o código, extrair a lógica para uma função **verificaUsuario\***:

function verificaUsuario() {

if(!usuarioEstaLogado()) {

header("Location: index.php?falhaDeSeguranca=true");

die();

}

}

Mas se esta função ficar em cada um dos arquivos, ainda não removemos os dois copy e paste. Devemos extrair este código para algum lugar comum, um arquivo que contenha todas as lógicas de um usuário logado. Chamaremos esse arquivo de **logica-usuario.php**:

<?php

function verificaUsuario() {

if(!usuarioEstaLogado()) {

header("Location: index.php?falhaDeSeguranca=true");

die();

}

}

E em nosso formulário e adiciona-produto:

include("logica-usuario.php");

verificaUsuario();

Já que estamos refatorando e melhorando nosso código, vamos trazer todo o código de lógica de usuário logado para cá. O próximo passo é a extração deste usuário logado, que aparece em diversos lugares. No **index.php**perguntamos se o usuário está logado, então extraímos para uma função que diz se ele está logado ou não:

function usuarioEstaLogado() {

return isset($\_COOKIE["usuario\_logado"]);

}

E também acessamos em diversos lugares o usuário logado:

function usuarioLogado() {

return $\_COOKIE["usuario\_logado"];

}

Podemos alterar agora nosso **index.php** para não acessar mais a variável cookie. Mas ainda temos um único lugar que acessa o cookie fora de nosso arquivo **logica-usuario**. Quando o usuário se loga temos o acesso dentro do**login.php**. No caso de sucesso queremos agora chamar um método que loga o usuário:

if($usuario == null) {

header("Location: index.php?login=0");

} else {

logaUsuario($usuario["email"]);

header("Location: index.php?login=1");

}

die();

Para isso implementamos a função:

function logaUsuario($email) {

setcookie("usuario\_logado", $email);

}

No **login.php** e no **index.php** usamos o **logica-usuario**:

include("logica-usuario.php");

Pronto! Toda nossa lógica de login está concentrada aqui. Quando precisarmos alterar algo basta utilizar este arquivo. Nossa próxima preocupação é que ainda armazenamos dados em nosso cliente e não podemos confiar nele, como veremos em breve.