WordPress

#WordPressSeguro

Guia prático para implementações de segurança em WordPress

Versão 1. Março de 2015

Guia prático para implementações de Segurança em WordPress

Sobre este guia prático

Criamos a iniciativa **#WordPressSeguro** para que ao longo de 2015 possamos demonstrar que o WordPress é Seguro. Inseguro é você. Realizaremos uma série de atividades entre palestras, publicações em portais, blogs e revistas e disponibilizaremos um guia prático para implementações de Segurança em WordPress

Este guia será atualizado periodicamente e enviado as pessoas interessadas no assunto. Estamos abertos e gostaríamos muito da sua contribuição. Acompanhe essa iniciativa através da página #WordPressSeguro no site da Apiki.

Garantia e direitos de uso

Este guia contém sugestões e códigos para implementações de segurança em WordPress. Use por sua conta e risco.

Backups

Antes de iniciar as implementações faça backup. Se possível defina rotinas e processos claros para a realização das suas cópias de segurança dos arquivos e banco de dados. E considere também a redundância, ou seja, ter cópias em diferentes locais seguros.

Nossa sugestão de plugin gratuito para essa tarefa é o BackWPup.

Atualizações

Manter o WP sempre rodando a última versão e também seus plugins e temas é um primeiro passo para estar com a casa em ordem e com um procedimento bem executado.

Prefira as atualizações via repositório de arquivo como Git integrado a um processo de *continuous delivery* ao contrário de atualizações através do *Dashboard* da aplicação. Com essa prática é possível validar as atualizações em outros ambientes antes de aplicar as mudanças em produção. Além disso, reverter para uma versão anterior será possível e prático caso o processo falhe ou aconteça algo desagradável.

Banco de dados

1. Não utilize o prefixo "wp_" nas tabelas no banco de dados. Dê preferência a prefixos como "wp_HaSHEs_", ou seja, "wp_" seguido de um hash único;
2. Renomear o usuário "admin", caso ele esteja em uso.

Debug

Ativar o recurso de debug permitirá a gravação dos alertas e erros que ocorrem em sua aplicação. Manter uma rotina de revisão é aconselhado para análise e correções.

Evite que os erros e alertas sejam exibidos e delete o arquivo com determinada periodicidade, em alguns casos ele fica bem grande.

O código abaixo deve ser incluído no arquivo wp-config.php.

```
define( 'WP_DEBUG', true );
define( 'WP_DEBUG_LOG', true );
@ini_set( 'log_errors', 'On' );
define( 'WP_DEBUG_DISPLAY', false );
@ini_set( 'display_errors', 'Off' );
```

Com o debug ativo será gerado o arquivo debug.log que se localiza em /wp-content/debug.log.

wp-config.php

O arquivo wp-config.php é o maestro que rege uma aplicação em WordPress. Ele deve ser bem tratado e potencializado com constantes que elevam o nível de segurança. Alguns conselhos.

1. Manter o arquivo um nível acima do diretório público;
2. Usar a permissão 400 (readonly) ou 600 (para permissão de escrita);
3. No arquivo .htaccess fazer uso de diretiva para proteção #1;
4. Fazer uso de algumas constantes do WordPress #2;
5. Considerar o uso e atualização das Chaves únicas de autenticação e salts. https://api.wordpress.org/secret-key/1.1/salt/ #3.

#1

```
<files wp-config.php>
order allow,deny
deny from all
</files>
```

#2

```
define( 'DISALLOW_FILE_EDIT', true );
```
ou
```
define( 'DISALLOW_FILE_MODS', true );
```

Com a constante DISALLOW_FILE_EDIT não será possibilitado ao usuário editar os arquivos de plugins e temas através da interface do WordPress. Já a constante DISALLOW_FILE_MODS além da mesma funcionalidade evita a instalação e atualização do core, plugins e temas.

#3

```
define('AUTH_KEY',
'H etc[q/4]g.G,u3NuQ5nj1Zx7J|C5{KFrW*G@!Il+4iUtS&v oU+b|EN<,t!&+T');
define('SECURE_AUTH_KEY',  'uPV(8/C/H qGtk.Y>G9GKRzAIKQ-?Y+
7^/BTQH+8ZY77|<*#%p3W6/JZ#dmm,73');
define('LOGGED_IN_KEY',
'#As7X$}Nt51+8{HCF{n+(YCLv.t3,Jr.C5+rc=k=&YK<ND.5n0s<V8 f5q01Q^9r');
define('NONCE_KEY',
'S]3z`P3B.Nr!s{.c;`{J|0v2FGBS+Dm7lZu]iA}Zy7R o+9@J~-bjDJs+8X~jJr ');
define('AUTH_SALT',        'Wpu IKvy>VSWpHijfvh#+][Vp+yr[1N9Fo
rY#!3(Rf4!c4kNqc:+!B!|/+#6BwL');
define('SECURE_AUTH_SALT', '{TBKIVfQd[NMdS6?`[C;-Vu|@J^zXNWq[IP7RBB[a
cvc~@z#Z5%v*7|R6@+Pekd');
define('LOGGED_IN_SALT',
'.WqaZf8/T,c>Rs&EB~i1GWGmJYZt};+AHR+t3)7+Ty0L|<n[)IH!-WEeCGjqL8Jd');
define('NONCE_SALT',       '-f3&2DI93=; 7Gs
}N`CK8MYR?u[: T5^Z@#78n<NLX@v70|?Es90Hlyvyl6.4,');
```

Não utilize os códigos acima. Gere um exclusivo e único através da url https://api.wordpress.org/secret-key/1.1/salt/

Permissões de arquivos e pastas

1. Permissão 644 para os arquivos #1;
2. Permissão 755 para os diretórios #2;
3. Remover arquivos sem usuários #3;
4. Permissão 400 ou 600 (para o caso de algum plugin precisar da permissão de escrita) no wp-config;
5. Permissão 600 para o arquivo debug.log

#1

```
find /path/to/wp-base-folder-install/ -type f ! -perm 644 -exec
chmod 644 {} \;
```

#2

```
find /path/to/wp-base-folder-install/ -type d ! -perm 755 -exec
chmod 755 {} \;
```

#3

```
find /path/to/wp-base-folder-install/ -nouser -exec rm {} \;
```

.htaccess

O arquivo de configuracão .htaccess pode abrigar instruções para garantir uma maior segurança. Os conselhos são:

1. Evitar a listagem dos arquivos nos diretórios #1;
2. Proteger o arquivo debug.log #2;
3. Proteger o arquivo wp-config.php #3;
4. Segurança para o "wp-includes" bloqueando os arquivos que devem ser somente incluídos pelo PHP #4;

#1

```
Options -Indexes
```

#2

```
<Files debug.log>
    Order allow,deny
    Deny from all
</Files>
```

#3

```
<files wp-config.php>
      order allow,deny
      deny from all
</files>
```

#4

```
# Block the include-only files.
<IfModule mod_rewrite.c>
RewriteEngine On
RewriteBase /
RewriteRule ^wp-admin/includes/ - [F,L]
RewriteRule !^wp-includes/ - [S=3]
RewriteRule ^wp-includes/[^/]+\.php$ - [F,L]
RewriteRule ^wp-includes/js/tinymce/langs/.+\.php - [F,L]
RewriteRule ^wp-includes/theme-compat/ - [F,L]
</IfModule>

# BEGIN WordPress
```

Exclusão de arquivos do core

Alguns arquivos do WordPress expõe a versão e detalhes da plataforma em uso e outros são necessários para o processo de instalação. Depois de instalado, ambos arquivos devem ser excluídos. A lista é pequena:

1. /wp-config-sample.php
2. /readme.html
3. /license.txt
4. /wp-admin/install.php

Robots.txt

Diga não ao Google. É importante evitar a indexação de conteúdo sensíveis do seu site. Os códigos abaixo devem ser colocados no arquivo robots.txt para dizer ao Google e aos demais buscadores o que eles não devem indexar.

```
User-agent: *

Disallow: /feed/
Disallow: /trackback/
Disallow: /wp-admin/
Disallow: /wp-content/
Disallow: /wp-includes/
Disallow: /xmlrpc.php
Disallow: /wp-
```

Evita SPAM e postagem fora do ambiente do site

Spam é uma praga e podemos implementar iniciativas contra ele para ficarmos resguardados dos lixos e mensagens indesejadas trazidas pelos spammers.

Os códigos abaixo devem ser colocados no arquivo .htaccess para quem utiliza Apache ou no arquivo de configuração do NGINX para quem o utiliza como servidor web. Esses códigos param os ataques no mecanismo de login e comentários do WordPress se caso seja acionado por um domínio diferente da instalação.

Troque "exemplo-dominio.com" pelo endereço correto do seu domínio.

```
# Apache
<IfModule mod_rewrite.c>
      RewriteEngine On
      RewriteCond %{REQUEST_METHOD} POST
      RewriteCond %{REQUEST_URI} .(wp-comments-post|wp-login)\.php*
      RewriteCond %{HTTP_REFERER} !.*exemplo-dominio.com.* [OR]
      RewriteCond %{HTTP_USER_AGENT} ^$
      RewriteRule (.*) http://%{REMOTE_ADDR}/$ [R=301,L]
</ifModule>

# NIGINX
location ~* (wp-comments-posts|wp-login)\.php$ {
        if ($http_referer !~ ^(http://exemplo-dominio.com) ) {
          return 405;
        }
    }
```

WordPress

#WordPressSeguro

https://apiki.com/wordpress-seguro/