SUMÁRIO

1. SEGURANÇA PHP: ESCOPO DO SISTEMA DE ARQUIVOS 1

2. PHP SEGURO: TRATAMENTO DO SISTEMA DE ARQUIVOS 1

2.1. ATUALIZAR O PHP 1

2.2. CONFIGURAR O PHP.INI 1

2.2.1. DIRETIVA DISABLE_FUNCTIONS 1

2.2.2. DIRETIVA DISABLE_ERROS 1

2.2.3. DIRETIVA DOC_ROOT 2

2.2.4. DIRETIVA OPEN_BASEDIR 2

2.2.5. EXPOSE_PHP 2

2.3. CONFIGURAÇÕES DO HTACCESS 2

2.3.1. ACESSO A PASTAS SENSÍVEIS VIA HTTP 2

2.3.2. O ARQUIVO PHPINFO 2

2.3.3. FORÇAR TRÁFEGO COM SSL 3

2.4. UTILIZAR SERVIDORES CLOUD 3

1. SEGURANÇA PHP: ESCOPO DO SISTEMA DE ARQUIVOS

Embora a segurança do servidor seja frequentemente considerada além do alcance de atacantes, há ampla oportunidade para os desenvolvedores tomarem medidas preventivas para ajudar a garantir a segurança de seus sistemas.

Embora grande parte da segurança do PHP se preocupe em proteger o aplicativo ou a camada de dados contra abusos, no escopo do código, a segurança dos arquivos que compõem o próprio aplicativo costuma ser negligenciada.

Frequentemente, a ameaça mais perigosa para qualquer aplicativo é uma ameaça interna. Ao considerar a segurança do aplicativo PHP, é importante considerar não apenas um usuário mal-intencionado da web, mas também um usuário mal-intencionado no sistema de arquivos. Sem a proteção adequada, os próprios arquivos PHP podem expor dados ou ser violados pelos usuários do sistema de arquivos.

2. PHP SEGURO: TRATAMENTO DO SISTEMA DE ARQUIVOS

2.1. ATUALIZAR O PHP

É importante atualizar regularmente sua versão do PHP porque as versões mais recentes geralmente contêm patches para problemas de segurança conhecidos. Se você não atualizar sua versão do PHP para a versão estável mais recente, os hackers podem explorar essas vulnerabilidades de segurança conhecidas com versões mais antigas.

2.2. CONFIGURAR O PHP.INI

Primeiramente, devemos atentar para as configurações oferecidas pelo PHP para tornar nosso aplicativo mais seguro.

2.2.1. DIRETIVA DISABLE_FUNCTIONS

Com essa diretiva, você pode desabilita funções especificas do php que podem gerar problemas de segurança, como por exemplo o fopen(),popen() e file(). Dessa forma: disable_functions=fopen,popen,file.

disable_functions=string

2.2.2. DIRETIVA DISABLE_ERROS

Deixe essa função habilitada só se estiver em ambiente de desenvolvimento, já que é indispensável saber detalhes sobres os erros de sua aplicação, mas quando for passar sua

aplicação para ambiente de produção deve desabilitar essa diretiva, e usar métodos alternativos para saber de erros gerados pela aplicação (salvar os erros em um arquivo de log).

disable_errors=On|Off

2.2.3. DIRETIVA DOC_ROOT

Essa diretiva pode ser configurada com um caminho que especifica o diretório raiz a partir do qual os arquivos PHP serão servidos. Se a diretiva doc_root estiver configurada com nada (vazia), ela será ignorada, e os scripts PHP serão executados exatamente conforme especifica a URL.

doc_root=string

2.2.4. DIRETIVA OPEN_BASEDIR

A diretiva open_basedir do PHP pode estabelecer um diretório base ao qual todas as operações com arquivos estarão restritas. Suponha que seu site esteja localizado na pasta home/www , para impedir que pessoas má intencionadas manipulem arquivos , tais como /etc/passwd por meio de alguns comandos simples do proprio PHP, você deve configurar essa diretiva da seguinte forma: open_basedir = “/home/www/”.

open_basedir=string

2.2.5. EXPOSE_PHP

É conveniente desabilitar essa opção, pois ela exibe detalhes sobre o PHP na assinatura do servidor.

expose_php=On|Off

2.3. CONFIGURAÇÕES DO HTACCESS

2.3.1. ACESSO A PASTAS SENSÍVEIS VIA HTTP

Para evitar esse problema, a melhor alternativa é proteger as pastas que contêm esses arquivos contra acessos externos. Para isso, basta colocar esses arquivos fora da pasta public_html ou adicionar uma ordem de negação de acesso as pastas no arquivo .htaccess.

2.3.2. O ARQUIVO PHPINFO

Mostra uma grande quantidade de informações sobre o estado atual do PHP. Isto inclui informações sobre as opções de compilação do PHP e extensões, a versão do PHP, informações do servidor e ambiente (se compilado como um módulo), o ambiente PHP, informação da

versão do SO, caminhos, valores principais e locais das opções de configuração, cabeçalhos HTTP e a licença do PHP.

Uma prática recomendada, devido à grande quantidade de informações contidas nesse arquivo, é deletá-lo do servidor. Se um atacante obtiver acesso ao sistema de arquivos, e especificamente a esse arquivo, pode, combinado com outras informações, explorar vulnerabilidades do sistema.

Outra prática, alternativa a essa, seria proteger o sistema de arquivos do servidor com uma senha, para proteger de acesso direto, como de um desenvolvedor mal intencionado em uma empresa, e também do acesso pela web, utilizando o arquivo de configuração .htacess.

```
<FilesMatch "/(phpinfo\.php|php\.ini)">

order allow,deny

deny from all

</FilesMatch>
```

2.3.3. FORÇAR TRÁFEGO COM SSL

Cada servidor deve ter um certificado SSL para transferir arquivos com segurança via HTTPS. Essa é uma medida imprescindível de segurança, principalmente para aqueles que acessam a aplicação.

No entanto, se tratando aqui da segurança do servidor da aplicação, é possível forçar, por meio do arquivo .htaccess, que os internautas utilizem HTTPS para ter acesso à aplicação web. Na prática, o que ocorre é um redirecionamento de HTTP para HTTPS.

```
RewriteEngine On

RewriteCond %{HTTPS} off

RewriteRule ^(.*)$ https://%{HTTP_HOST}%{REQUEST_URI} [L,R=301]
```

2.4. UTILIZAR SERVIDORES CLOUD

A hospedagem é a etapa final e primordial para qualquer aplicativo da web, já que você sempre cria o projeto em servidores PHP locais e os implanta em servidores ativos que oferecem hospedagem compartilhada, em nuvem ou dedicada.

Uma prática recomendada é a hospedagem em nuvem como DigitalOcean, Linode, AWS e Firebase. São plataformas rápidas, seguras, e que ainda oferecem uma série de recursos.