Segurança em PHP

Desenvolva programas PHP com alto nível de segurança e aprenda como manter os servidores web livres de ameaças

Márcio Pessoa

Novatec

CAPÍTULO 1

Conceitos gerais

No primeiro capítulo serão abordados alguns conceitos de desenvolvimento visando à segurança do projeto de um modo geral. Não serão explorados aspectos específicos do PHP; o objetivo deste capítulo é explicar de forma simples os princípios de segurança em um projeto, independente de linguagens de programação ou sistemas operacionais.

Ao concluir este capítulo, o leitor estará familiarizado com os seguintes tópicos:

- Conceitos básicos de segurança relacionados à programação.
- Definição de usabilidade de um sistema.
- Razões para realizar a programação visando a usuários ilegítimos.
- Considerações sobre desempenho.
- Limites de acesso.

1.1 Segurança

Levando em consideração as boas linguagens de programação, elas por si só não podem ser consideradas seguras ou inseguras; a segurança de sua aplicação depende, sobretudo, de todo o projeto e, mais especificamente, do código do programa.

Nenhum esforço isolado é suficiente para evitar problemas de segurança de computadores. Com toda certeza o leitor já deparou com frases como:

- "Nossa política de segurança é à prova de falhas."
- "A tecnologia usada neste DVD para evitar pirataria é absolutamente confiável."
- "Este sistema é totalmente seguro!"

Certamente não é necessário mencionar que essas frases foram muito mal sucedidas, pois a segurança não pode ser medida de forma absoluta; a segurança de um sistema deve ser aferida, não devendo ser tratada como uma característica do mesmo.

A segurança de um sistema deve ser balanceada com o custo e a usabilidade do projeto. De modo geral, um projeto com algum nível de segurança implementado tem um custo superior a um projeto sem nenhum requinte de segurança. Esses custos dizem respeito à contratação de pessoal especializado, maior tempo para confecção do projeto e outros itens que acarretam maior custo financeiro:

- Custo
  - Tempo de projeto, pesquisa, programação e testes.
  - Profissionais mais bem-qualificados.
  - Hardware específico ou mais poderoso.
  - Mais banda de internet para suportar maior interação com o usuário e/ou criptografia.
- Usabilidade
  - Facilitar ou complicar.
  - Cartões de acesso (tokens).
  - Tempo de sessão.
  - Senhas, chaves etc.

Para não tornar o projeto um bicho de sete cabeças, o nível de segurança desejado deve ser definido com cautela para não tornar o projeto economicamente impraticável ou até mesmo tão complicado que não possa ser utilizado.

Outro item muito importante a ser levado em consideração é que a segurança deve fazer parte do design do projeto. Muitos programas têm sua parte de segurança implementada depois que está pronto. Isso, além de implicar na existência de uma implementação problemática dos recursos de segurança, implica aumentar muito o tempo do projeto e conseqüentemente os custos deste.

O último, porém não menos importante, recurso que citarei é a informação – é fundamental estar sempre atualizado. Existem inúmeros recursos disponíveis na internet, que vão desde websites especializados em segurança até listas de discussão.

1.2 Usabilidade *versus* segurança

Quanto mais recursos de segurança forem implementados no sistema, mais complicado será o seu uso.

Existe um tipo de sistema muito popular que pode exemplificar isso muito bem – o Internet Banking. Tal sistema precisa ser muito fácil de usar, mas necessita contar com recursos de segurança muito confiáveis.

Os bancos têm usado os mais diversos e criativos artifícios de segurança para garantir a legitimidade dos usuários autenticados no sistema. Para tanto, usam recursos como:

- Login (geralmente número da agência e da conta)
- Senha
- Palavra secreta ou contra-senha
- Número sorteado do cartão de chaves de acesso (token)
- Posição da seqüência dos números de segurança impressos no cartão de débito

Certamente tem se tornado cada vez mais complicado trabalhar com Internet Banking, o que tem feito que as pessoas prefiram ir até as agências bancárias para não terem de aprender a usar ou por não confiarem nessa tecnologia.

Alguns bancos até fizeram softwares específicos para os clientes acessarem os serviços remotamente, mas isso exigia muito mais do que simplesmente distribuir um disquete com o programa; envolvia custos com suporte técnico e treinamento dos clientes para trabalhar com um programa que nem sempre era amigável.

O desenvolvedor precisa usar de sensatez ao implementar recursos de segurança em seu sistema. Leve em consideração os seguintes itens:

- Valor da informação manipulada
- Perfil do usuário
  - Faixa etária
  - Nível de instrução
  - Finalidade do sistema (lazer ou trabalho)
- Hardware usado

1.3 Postura de defesa

É importante manter uma constante postura de defesa, a qual consiste em procurar manter informações protegidas. Ainda que uma informação não tenha relevância ou não seja confidencial, terceiros não devem tomar conhecimento dela para não tirar proveito de alguma forma. Isso é conhecido também como método da ignorância – quanto menos alguém sabe sobre você, mais privacidade você terá e, conseqüentemente, mais segurança.

Em termos práticos isso se resume em atitudes muito simples, afinal, se você não ganha nada em divulgar o sistema operacional que usa, não há então necessidade de divulgar. O mesmo se aplica à linguagem de programação, bancos de dados e demais recursos.

1.4 Desempenho

Ao desenvolver programas, esteja certo de que sempre é possível fazer melhor. Um bom programa deve fazer tudo o que foi planejado e ainda ter as seguintes características:

- Consumir o mínimo de tempo de CPU para ser executado.
- Alocar a menor quantidade possível de memória.
- Gravar todos os dados necessários usando o menor espaço em disco possível.
- Utilizar a rede interna e a internet de forma econômica e racional.

Procurar medir os recursos que são utilizados para execução do programa é uma prática extremamente importante, pois já temos em mente que os programas devem sempre consumir o mínimo de recursos do sistema. Dessa forma, o processador terá tempo livre para executar outras tarefas.

Se uma página web dinâmica consome poucos recursos do servidor para ser gerada, então o servidor poderá atender tranqüilamente múltiplas requisições.

Na concepção de um projeto é importante usar temporizadores, trechos de código que devem ser inseridos no corpo do script para medir o tempo total de execução do programa e, além disso, é de suma importância aferir os recursos disponíveis no servidor. Isso pode ser feito com a ajuda de softwares específicos para essa tarefa. No Apêndice A, será possível encontrar uma visão geral de alguns métodos de temporização e, no Apêndice B, serão abordadas algumas técnicas de monitoração.

1.5 Usuários ilegítimos

Geralmente programadores definem a estrutura de um programa baseando-se nos usuários legítimos do sistema. É necessário mudar esse conceito. O programador que escreve sistemas com responsabilidade deve desenhar a estrutura do sistema pensando sempre em primeiro lugar no usuário ilegítimo e posteriormente no usuário real do sistema.

De fato é uma técnica muito simples, mas, depois que o projeto é definido dessa forma, a programação ganha muito em velocidade, e, como já sabemos, tempo é dinheiro.

1.6 Filtro de dados

Tão importante quanto beber água para evitar a desidratação é filtrar as variáveis de entrada do programa, pois tudo que vem de alguma fonte externa deve ser minuciosamente inspecionado. Dados digitados pelos usuários, coletados de arquivos, enviados por um formulário, ou até interações feitas com servidores de bancos de dados devem passar por um crivo.

Os filtros de dados podem ser considerados como a forma mais significativa de implementar segurança em uma aplicação.

Um deslize muito comum que faz programadores perderem os cabelos é realizar checagens de consistência de dados apenas usando JavaScript. Esses scripts são excelentes para melhorar a interface com o usuário, em que os campos de um formulário podem ser testados sem haver o recarregamento da página. O problema é que o JavaScript pode ser desativado no browser, ou ainda o formulário pode ser preenchido por um "robô" (programa que busca por informações em um formulário) funcionando como browser e passando despercebido pelo JavaScript. Dessa forma, é importante realizar a checagem de consistência de dados no servidor também.

1.7 Manipulação de erros

Tão comum quanto ver aves no céu é ver websites de grandes instituições exibindo informações confidenciais quando apresentam problemas. Todos que navegam pela internet já tiveram a experiência de entrar em um site e o mesmo estar indisponível.

A falha nesse caso não é o fato de estar indisponível, afinal problemas realmente acontecem. A gravidade consiste em, quando da ocorrência de problemas, informações como endereço do servidor de banco de dados, variáveis do sistema e outras informações serem exibidas pelo browser, de modo que pessoas mal intencionadas podem tirar proveito dessas informações para prejudicar o seu sistema.

Por essa razão, é importante manipular exceções geradas pelo sistema e garantir que mensagens de erro sejam registradas somente em arquivos de log.

É de suma importância então permanecer atento aos logs gerados pelo sistema, pelo servidor web e pelos outros recursos do sistema operacional.

1.8 Limitador de acessos

Um limitador de acesso não é propriamente um programa ou uma ferramenta; trata-se de um ou mais recursos que devem ser implementados em um projeto com foco em segurança.

Um típico exemplo de um limitador de acesso é uma função no sistema que reage a tentativas mal sucedidas de login por um usuário, o que pode ser caracterizado como uma possibilidade de quebrar uma senha. Esse tipo de limitador de acesso é conhecido também como *rate limit*.

Outro exemplo clássico é realizar a limitação de acessos por endereço IP. Isso pode ser feito usando regras de firewall ou com o auxílio de ACLs implementadas no sistema.