Securing your WebSite, is it that difficult ?

Introdução

João Pedro Pereira
@joaoppereira
#codebits2010
Novembro 2010

Agenda

- Validações
  - ~~Client Side~~
  - Server Side
- Vectores de Ataque
- (Business) Logic Vulnerabilities

⇑

em 5min

Não confiar em ninguém !

Alguns principios...

- Nada é 100% seguro...
- A segurança deve estar em equilibrio com a usabilidade.
- A segurança não é um *add-on*, deve fazer parte do processo de desenho.

Alguns principios... de desenho

- Vários níveis de protecção.
- Apenas conceder os privilégios necessários!
- A complexidade facilita o aparecimento de falhas
- "Blacklist approaches are quite fragile" - OWASP

Blacklist Approach

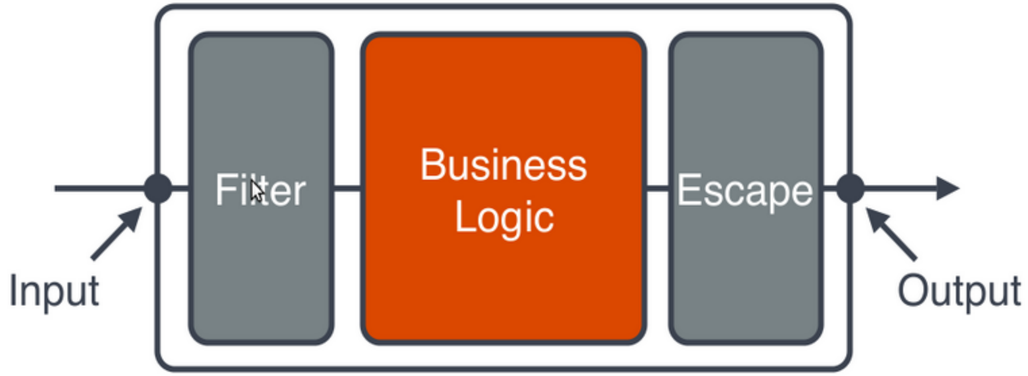
```
function verify($str) {
$badwords = array('javascript','expression','vbscript','script',
'applet','alert','document','write','cookie','window','<?php','<?',
'onblur','onchange','onclick','onfocus','onload','onmouseover',
'onmouseup','onmousedown','onselect','onsubmit','onunload','onkeydown',
'onkeyup','onresize','xmlns','basefont','base','behavior','blink',
'body','embed','form','frameset','frame','head','html','iframe','input',
'layer','link','meta','object','style','textarea','xml','cmd',
'passthru','eval','exec','escapeshellarg','escapeshellcmd','proc_open',
'proc_nice','shell_exec','system','fopen','fsockopen','file',
'file_get_contents','readfile','unlink','\\"\/','<','<','img',
'ALTER DATABASE','ALTER TABLE','CREATE DATABASE','CREATE TABLE',
'DROP DATABASE','DROP TABLE','RENAME TABLE','DELETE','REPLACE',
'SELECT','TRUNCATE','CREATE PROCEDURE','ALTER PROCEDURE',[.....]]);

for($x = 0; $x < count($badwords); $x++) {
        preg_match("/".$badwords[$x]."/", $str, $matches);
        if(count($matches) > 0) return false;
        else return true;
} }
```

Escapando às Blacklists

Not Good	Nice
Plain Text	encoded attacks!
alert('xss')	prompt('xss')
<script>	<ScRIPT src=//0x.lv?
' OR 1=1 --	' OR 2=2 --
UNION SELECT	UNION ALL SELECT
/etc/passwd	/lol/../etc/lol/../passwd

Mais sobre BlackLists e como fugir a elas nos Slides de XSS.

Alguns principios...de validação



- **FIEO (Filter Input, Escape Output)**
  - Filtrar Input
    - Dados entram válidos?
  - Filtrar Output
    - Dados que saem são *seguros*!

Validações :: Filtrar Input

- (Blind) SQL Injection / Database Hacks
- Path Traversal / Command Injection
- Cross Site Request Forgery (CSRF) / Session Hijacking (Fixation)

Validações :: Filtrar Output

- HTML Injection
- Cross Site Scripting (XSS)
  - Browser Exploitation

Validar o Input pode ser simples quando os campos podem ser processadas através de expressões regulares:

```
/^([-a-z0-9_-])+$/i          /* alfanumérico + travessões */
/^[0-9]+$/                     /* numeros naturais */
/^(((?:\.)(?!\.))|\w)+$/     /* nomes de ficheiros válidos */
```

Mas pode tornar-se complicado em situações que vamos receber texto e que até se aceita HTML... É preciso ter cuidado com tags "perigosas" e praticamente com todos os métodos pois podem também trazer problemas.

REGEX Powaa !

```
if( $idade > 0 and $idade < 120 ) { /* code */
$query = "... AND `idade` = '$idade'";
} else die('buh');
```

Cuidado ! 19 UNION ALL SELECT ... Result: **TRUE**!
Mais seguro:

```
$idade < '120' //$idade < 120
```

Melhor ainda seria com REGEX !

Telefone:

Email:

Username:

Ver Código

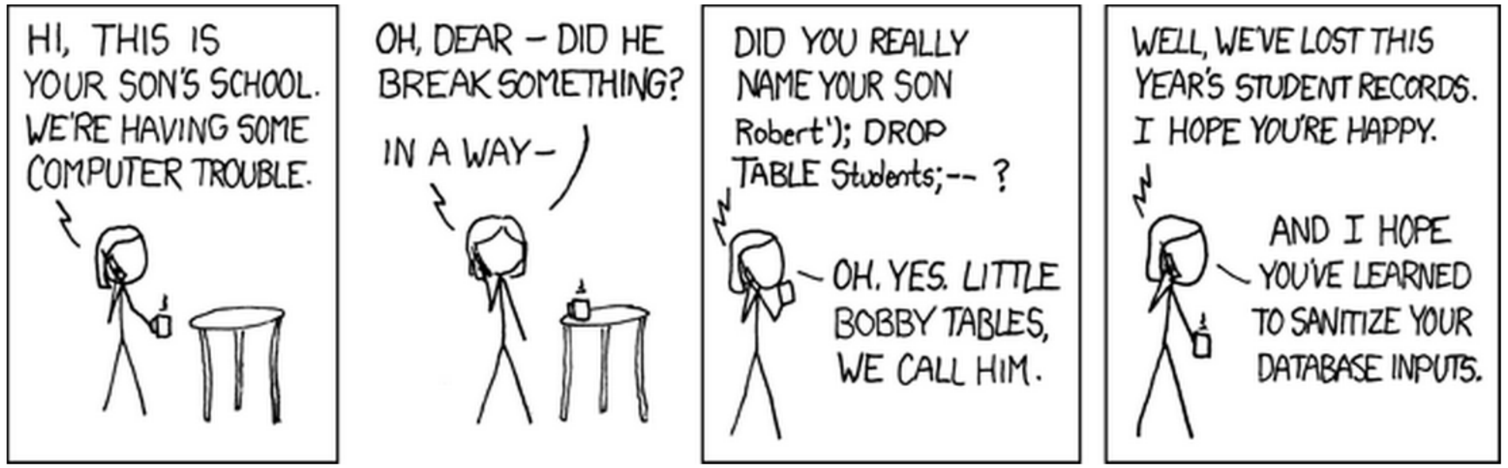
```php
<?php

foreach($_GET as $key=>$value) {
if( $key == 'Telefone' ) {
        echo (preg_match("/^[0-9]{9}$/i", $value)) ? 'ok' : 'bad';
} else if ($key == 'Email' ) {
        echo (preg_match("/^([a-z0-9\+_\-]+)(\.[a-z0-9\+_\-]+)*@([a-z0-9\-]+\.)+[a-z]{2,6}
} else if( $key == 'Username' ) {
        echo (preg_match("/^([-a-z0-9_-])+$/i", $value)) ? 'ok' : 'bad';
}
}

?>
```

E casos mais complexos ?

Pode dar problemas com:

- SQL Injection
- Cross Site Scripting (XSS)

SQL Injection



Procedimento Correcto: Resultado

```
SELECT * FROM users WHERE `login`='$_POST["user"]' AND `password`='$_POST["password"]'
```

Request: url/?user=' or 1=1 --&password=codebits

```
SELECT * FROM users WHERE `login`='' OR 1=1 --
```

Ups... Resultado

Ver Código

```php
//connect to the database
$db = mysql_connect( $dbHost, $dbUser, $dbPass ) or die ( 'Error connecting to database.'

mysql_select_db( $dbDatabase, $db ) or die ( 'Couldn\'t select the database.' );
$login = $_GET['user'];
$pass = $_GET['password'];
$result=mysql_query( "select * from teste where login='$login' AND password='$pass'", $db

//check that at least one row was returned
echo urldecode( $_SERVER['REQUEST_URI'] );

$rowCheck = mysql_num_rows( $result );
if( $rowCheck > 0 ){
while( $row = mysql_fetch_array( $result ) ){

        //start the session and register a variable
        //successful login code will go here...
        echo '<br/>Wellcome to InsecureApp!';
        echo '<br/>Account Settings';
        echo '<br/>username:password<br/>';
        echo $row[0] . ':' . $row[1] . '<br/>';
}

} else { //if nothing is returned by the query, unsuccessful login code goes here...
        echo '<br />Incorrect login name or password. Please try again.';
}
```

SQL Injection :: Ameaças e Protecções

- aceder/manipular/eliminar dados
- acesso não autorizado
- DoS
- executar código na máquina

- Proteger
  - validar as entradas
    - regular expressions
    - whitelist's
  - *escape* de caracteres periogos
  - dar o minimo de privilégios possíveis
  - prepared statements

Blind SQL Injection

O que é? Como explorar?

- 1º - distinguir querys verdadeiras de falsas
  - 1' AND 1=2 --
  - 1' OR 1=1 --
- 2º - recolher informações
  - ' UNION ALL SELECT IF(SUBSTRING (password,1,1) =CHAR(49), BENCHMARK(1000000,ENCODE('sapo','codebits 2010')),null),1 FROM teste WHERE login='lol' --

Blind SQL Injection (cont.)

- 2º - recolher informações
  - ' UNION ALL SELECT sad,1 FROM asdf -- &password=codebits
  - ' UNION ALL SELECT IF(SUBSTRING(password,1,1)= CHAR(49),BENCHMARK(1000000, ENCODE('sapo','codebits 2010')),null),1 FROM teste WHERE login='lol' --
  - ' AND ASCII(SUBSTRING(login,1,1)=**n** AND 1=1 --
  - ' AND LENGTH(login)=**n** AND 1=1 --
- Don't fuck your brain !
  - SQL Map

SQL Injection :: Prepared Statements

```php
$dbh = new PDO('mysql:host=127.0.0.1;dbname=codebits_talk',
                $dbUser, $dbPass);

$stmt = $dbh->prepare("SELECT * FROM `teste`
                        WHERE `login` = '?' AND `password` = '?'");
if ($stmt->execute(array($_GET['login'], $_GET['password']))) {
  while ($row = $stmt->fetch()) {
    print_r($row);
  }
}
```

Cross Site Scripting (XSS)

- Impacto

  Demo?

  - browser pwned
  - roubo de dados da vítima
  - DDoS
  - depende da imaginação do atacante...
- Proteger
  - validar as entradas
    - regular expressions
    - whitelist's
  - *escape* de caracteres periogos

Truques para Escapar às BlackLists (1)

```
<object><param name="src"
value="javascript:alert(0)"></param></object>

<object data="javascript:alert(0)">
```

Creditos: Alex. K (kuza55)

Truques para Escapar às BlackLists (2)

```
<isindex type=image onerror=alert(1) src=>
```

```
<isindex action=javascript:alert(1)
type=image>
```

Creditos: Gareth Heyes

Truques para Escapar às BlackLists (3)

```
<x:script
xmlns:x="http://www.w3.org/1999/xhtml">alert('
</x:script>
```

Conteúdo mostrado como text/xml | text/xml-xhtml executa JavaScript usando namespaces de HTML e xHTML.

XSS :: Proteger - Escaping

Canonicalize / Normalize

- reduzir uma string, possívelmente codificada, e convertê-la até a sua forma mais simples;

(Algumas) Formas de Escrever "<"

%3C

0x3C

0xC0 0xBC

0xE0 0x80 0xBC

XSS :: Proteger - Sanitizing HTML

Não re-inventar a roda !

- HTML Purifier
- AntiSamy


E os CSRF pá?

CSRF / XSRF :: Proteger

- random token para cada pedido
- ^ guardados em sessão (server side)
- e em hidden fields no formulários
- é só comparar valores
- vulnerabilidades XSS estragam tudo

O que se faz no Codebits enquanto espera pelo SecurityContest part2 ?
Karaoke, Guitar Hero,... e que tal desenvolver um filtro para CSRF ?

Ver Código

```php
<?php
/*
$csrf_protect = new CSRFProtect();
$csrf_protect->enable();
*/
class CSRFProtect {
    var $csrf_field = '_csrf_token';

    function gen_token( ) {
        return sha1( session_id( ) );
    }

    function error( $msg ) {
        die( 'Logged' );
        /* implement log using the $msg */
    }

    function to_protect( ) {
        // Only protect if the user has a session
        $session_id = session_id( );
        if ( empty( $session_id ) ) return false;
        return true;
    }

    function enable( ) {
        if ( !$this->to protect( ) ) return;
```

```
function enable( ) {
    if ( !$this->to_protect( ) ) return;

    $csrf_token = $this->gen_token( );
    // If it's a POST, check the token matches
    if ( !empty( $_POST ) ) {
        $form_token = $_POST[$this->csrf_field];
        if ( $form_token != $csrf_token ) {
            $this->error( "$form_token != $csrf_token" );
            return;
        }
    }
    // On POST or GET, we still need to add the token to any forms
    ob_start( array( $this, 'ob_callback' ) );
}

function ob_callback( $html ) {
    $token = $this->gen_token( );

    $hidden = '<div style="display: none;"></div>';
    $hidden .= '<input type="hidden" name="';
    $hidden .= $this->csrf_field . '" value="' . $token;
    $hidden .= '"/>');
    $hidden .= '</div>';

    return preg_replace('/(<form\W[^>]*\bmethod=(\'|"|)POST(\'|"|)\b[^>]*>)/i','\\1'.
}
```

ClickJacking

- FrameBusting
- ~~Verificar Referer~~
- X-Frames-Options (DENY | SAMEORIGIN)
- Content Security Policy (em v. Beta @ Firefox 3.7)

ClickJacking :: FrameBusting

- Protege a página de ser vítima de clickjacking
- Só funciona se a página não sofrer de XSS

```
if (top != self) {
        top.location = self.location;
}
```

ClickJacking :: X-Frames-Options

- Limitações
  - **HTTP HEADER**
  - sites com múltiplos dominios
  - proxies
- Exemplo de Implementação (Apache)

```
Header always append X-Frame-Options SAMEORIGIN
```

(Business) Logic Vulnerabilities

- User Enumeration
- Brute-Force
- Remember / Reset Password
- Logout & Gestão de Cache

And now...

Still thinking that Securing your WebSite is that difficult ?

:)