information

Review

A Comparative Study of Web Content Management Systems

Jose-Manuel Martinez-Caro [1,*], Antonio-Jose Aledo-Hernandez [1], Antonio Guillen-Perez [1], Ramon Sanchez-Iborra [2] and Maria-Dolores Cano [1]

[1] Department of Information Technologies and Communications, Universidad Politécnica de Cartagena (UPCT), Edif. Cuartel de Antigones, Plaza del Hospital 1, 30202 Cartagena, Spain; antonioj.aledoh@gmail.com (A.-J.A.-H.); agp4@alu.upct.es (A.G.-P.); mdolores.cano@upct.es (M.-D.C.)

[2] Department of Information and Communications Engineering, Universidad de Murcia (UM), Avda. Teniente Flomesta, 5, 30003 Murcia, Spain; ramonsanchez@um.es

* Correspondence: jmmc0@alu.upct.es; Tel.: +34-968-328-871

Received: 15 December 2017; Accepted: 25 January 2018; Published: 27 January 2018

Abstract: Web Content Management Systems (WCMS) play an increasingly important role in the Internet's evolution. They are software platforms that facilitate the implementation of a web site or an e-commerce and are gaining popularity due to its flexibility and ease of use. In this work, we explain from a tutorial perspective how to manage WCMS and what can be achieved by using them. With this aim, we select the most popular open-source WCMS; namely, Joomla!, WordPress, and Drupal. Then, we implement three websites that are equal in terms of requirements, visual aspect, and functionality, one for each WCMS. Through a qualitative comparative analysis, we show the advantages and drawbacks of each solution, and the complexity associated. On the other hand, security concerns can arise if WCMS are not appropriately used. Due to the key position that they occupy in today's Internet, we perform a basic security analysis of the three implement websites in the second part of this work. Specifically, we explain vulnerabilities, security enhancements, which errors should not be done, and which WCMS is initially safer.

Keywords: web content management systems; websites; web pages; security; joomla!; wordpress; drupal; internet

1. Introduction

The Internet is more alive than ever: the number of websites on the Internet has already exceeded one billion [1], the number of Internet users has achieved almost four billion [2], and the penetration rate is more than 51.7% [2]. By observing the growth trend during recent years (see Figure 1), we could come to the conclusion that this evolution is entering a permanent phase, i.e., a linear instead of an exponential increase. However, if we check other statistics, we find that 60% of Small and Medium Businesses (SMB) do not have a corporate website [3]. As a consequence, there is still much room for growth, and Web Content Management Systems (WCMS) facilitate this task. WCMS are software platforms generally used when a website is needed, commonly requiring different user roles, but when at the same time there is a lack of web programming knowledge [4–7]. As a tool, WCMS are booming, being very helpful for beginners in web development or for small business managers, because websites can easily be implemented at a relative low cost [8,9]. One example could be a newspaper editorial, where journalists are interested in launching an online edition. In this case, journalists may not know enough about web programming to develop their website, having only office software skills. It is in this scenario that WCMS have great potential. The *open-source* WCMS, also called second-generation WCMS, are platforms often based on PHP (PHP Hypertext Preprocessor) and usually fed by communities of users who contribute novel solutions and new

functionalities [11]. The basic WCMS structure has the following parts: (i) the files of the content manager; (ii) a hosting provider to store the files of the content manager; and (iii) a linked database, e.g., MySQL (My Structured Query Language), to store website information. A WCMS provides an administration or development area called the *back-end*, where articles, functionalities, or any other aspect can be added, deleted, or modified. On the other hand, the visible part of a website, i.e., what a visitor *sees*, is called the *front-end*.

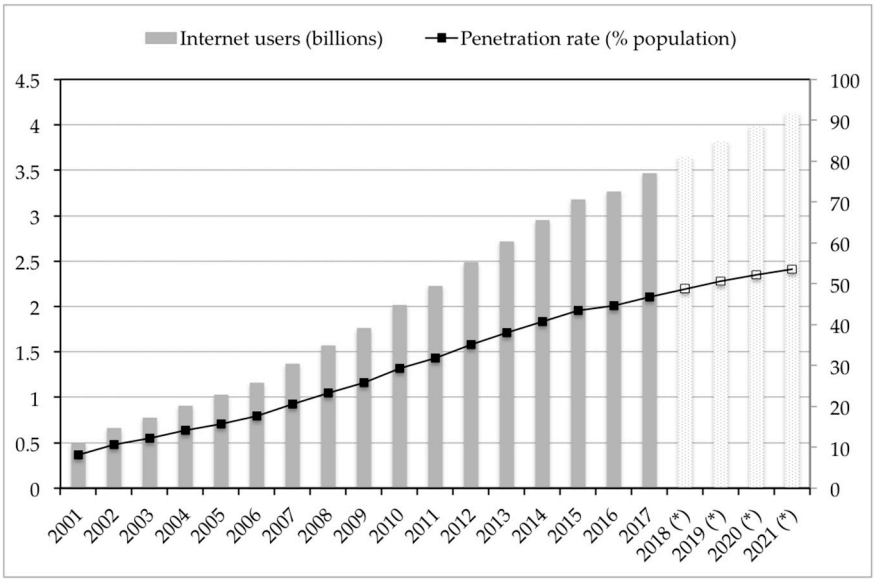

Figure 1. Internet users and penetration worldwide, 2001–2021 (billions, % of population); data shown as * are expected values [10].

Security vulnerabilities are a weak point of any system at the design and implementation levels, and WCMS are not an exception [9]. Bugs and flaws are both software problems differing in terms of the level at which they are presented (implementation and design, respectively); in any case, they are dangerous, and should be analyzed [12]. In a simplified way, an attack consists of two phases: discovery, which is longer in time, and exploitation, once the weakness of the WCMS is known [13]. A secure application ensures authentication, confidentiality, integrity, and availability. Depending on the service we are focusing on, we could put aside some of these security characteristic (e.g., confidentiality). However, WCMS are a key piece of today's Internet, and have become a target for attackers [14,15]. If a WCMS has security vulnerabilities, it may become inaccessible [16], with the corresponding negative effects. Usual attack consequences are: confidential data destruction, data modification, misuse of a web-server for illegal activities, and Denial-of-Service (DoS), among others.

Due to the key position that WCMS occupy in today's Internet, and the security concerns that can arise with an inappropriate use or configuration, the goal of this article is twofold. Firstly, we extend the work carried out in [17], explaining how to manage a WCMS from a tutorial perspective, and what can be achieved with its use. In order to do so, we have chosen the best WCMS in terms of popularity and performance [18–21]; namely, WordPress [22], Joomla! [23], and Drupal [24]. Regarding popularity, while only 23.6% of web pages were created with a WCMS in January 2011, this figure increased to 48.8% in January 2018 [19]. In 2011, 13.1% of the websites were implemented with WordPress, 2.6% with Joomla!, and 1.4% with Drupal. In 2018, the most used WCMS is WordPress (29.3%), the second preferred option is Joomla (3.2%), and the third one is Drupal (2.3%). In other words, WordPress has a market share of 60% in January 2018, followed by Joomla! and Drupal with 6.5% and 4.6%, respectively [18]. In terms of performance, the works done in [20,21] compared several WCMS and both studies emphasized that WordPress, Joomla!, and Drupal were the most efficient ones, because they achieved a better load time and good static content, showed the highest number of installations,

presented better documentation support, etc. Although Joomla! and Drupal have lost some relevance, they are still widely used as shown in [25], what justifies choosing them for this comparative work. The methodology followed for the comparison consists of creating three equal web sites, each one with a different WCMS. The advantages and drawbacks of each selected WCMS, as well as their complexity, will be discussed. Secondly, we carry out a basic security analysis of each implemented website, reporting vulnerabilities, how to bestow security on them, which mistakes can be avoided, and which WCMS is initially safer.

The rest of the paper is organized as follows. Section 2 includes the related work. In Section 3, all details about the selected WCMS are discussed. Section 4 provides a guide to how to create a website using Joomla!, Drupal, and WordPress, as well as a qualitative comparison. Section 5 presents the outcomes of the basic security analysis. The document ends with the conclusion.

2. Related Work

There are several works in the related literature addressing operation, performance, and security in WCMS. According to [7] and [8], WCMS stood out because a quick design is possible by using frameworks and templates. Additionally, both studies highlighted the integration of information and knowledge and high levels of efficiency and usability as the main advantages of WCMS. Patel et al. compared, in [20,26], several WCMS, trying to identify the proper scenario for each WCMS. The authors in [21] introduced a comparative study of seven WCMS, including Drupal, Joomla!, and WordPress. However, the comparison was based only on implementing a website with each WCMS and on determining if a pre-established functionality, defined as a previous requirement, was available or not. In [27], an exhaustive analysis of WCMS was presented, but from a purely qualitative perspective. They addressed other properties of WCMS, such as the effect of the web-server on the performance (e.g., Apache, Nginx, etc.) according to available statistics.

On the other hand, most WCMS vulnerabilities are found in plugins and extensions (80%), and once any vulnerability is known, hackers could be able to inject malware on web-servers [9]. The studies done in [16,27] introduced and analyzed security in WCMS. In particular, authors identified in [16] the vulnerabilities and attacks that WCMS are exposed to, and proposed possible countermeasures. As a case study, they used Joomla! and Drupal (with earlier versions than those used in this work), and carried out simple penetration tests using the tools WebScarab [28] and TamperData [29]. As the main result, the authors concluded that even though security mechanisms were provided, both WCMS could easily become victims of attacks. An interesting tool called ZenIDS was presented in [30] to identify malicious activities in PHP applications and to protect them. To perform this task, the process distinguishes two phases: feature learning, and monitoring. Finally, authors in [31] computed a web-server's risk factors when cybercriminals use Apache, Nginx, or Microsoft IIS, among others. They also included the effect of running the most popular WCMS platforms with up-to-date versions on those web-servers, finding that recent versions were more compromised than less-popular WCMS platforms or outdated versions. The more add-ons or extensions that are included in a WCMS platform, the more the negative impact on security. As will be discussed later, our work updates [16] and contributes in a practical way to the results obtained in [27].

3. Web Content Management System under Study

Joomla! [23] is one of the most popular WCMS for creating dynamic websites [25]. Joomla! is compatible with MySQL, SQL Server, and PostgreSQL databases. One of the key features of Joomla! is that it offers the widest range of functionalities, such as picture galleries, forums, chats, blogs, news, etc. Likewise, Drupal [24] is also aimed at creating dynamic websites. It presents high compatibility with different databases. Security, fast loading, and a wide variety of user roles are the main Drupal features; as an example of the last characteristic, Drupal allows us to limit access of a specific user such that they are only able to modify the properties of a certain functionality, and even to modify only particular parameters of that functionality. On the other hand, WordPress [22] was initially oriented

towards creating blogs, but has evolved to provide web applications and e-commerce solutions. One key feature of WordPress is its great Search Engine Optimization (SEO) positioning. The reason for this is that WordPress has many plugins to improve a quick inclusion in search engines, compared with other WCMS [32]. Moreover, it is possible to create a simple blog for free under the platform subdomain (*.wordpress.com*). Extensions and modules can also be used in Joomla! and Drupal to achieve a better SEO, but these options do not have such a high impact as WordPress. All WCMS platforms offer to improve SEO manually. At the time of carrying out this work, the latest available versions were: Joomla! 3.6.5, WordPress 4.6.2, and Drupal 7.54. Tables 1–3 include a brief comparison of the features of each version.

There are many differences among the three selected WCMS. There are many possible scenarios but, generally speaking, websites start with a reduced size and they do not involve too many characteristics, modules, or contents. Over time, the increase in website visitors, new developed content, sales volume, etc., generates the need to add new modules and extensions to the website, giving it more capabilities. WordPress offers many extensions to insert a specific functionality, but it does not have the wide range of functionalities that Joomla! has. Additionally, WordPress includes the option to add integrated extensions to the management environment. Lastly, Drupal is the most complex WCMS in terms of extending its capabilities, because a specific functionality can have dependencies from different libraries or functions, making this extension process long and tedious. The same applies to the functionality range, where Joomla! and WordPress have a large number of functions to extend the website. In contrast, this option is more limited with a Drupal implementation. Repositories are very important for updating WCMS, implementing new functionalities, and improving the existing ones. These repositories can be distributed, which is the case of Joomla! and WordPress, or centralized, such as in the case of Drupal. The largest user communities belong to Joomla! and WordPress, and hence they have a very active user community and excellent documentation. On the other hand, the Drupal user community is limited, because it is more complex to implement and manage, having, as a result, a limited documentation compared to the others. In contrast, Drupal's complexity allows a higher diversity of user roles, which is an advantage compared to Joomla! and WordPress. Table 4 summarizes the main features of the three selected WCMS. The comparison has been done based on available official documentation, previous studies from the related literature, and the experimental work carried out in this paper.

Table 1. WCMS: Comparison of Joomla! versions and features.

Version	New Features
1.0	UTF-8 Built-in, Database Drivers, Official Supported Components Already in the Core, Plugin Framework, FTP Uploading of Extensions, Visible to Administrators when the Website is Offline, SQL Injection and XSS Solutions, HTTPS, Include CHANGELOG.php File for More Information, Media Manager Support for XCF, ODG, ODT, ODS, ODP File Formats and New Menu to Clear All Caches
1.5	Support GIF Images, LDAP Security Fix, SEO Improvement, RSS Feed, Solve XSS, DoS and SQL Injection Vulnerabilities in Back-end and Front-end Editing
2.5 (1.6 & 1.7)	Smart Search Engine, New Database Supports, Captcha by Default, URLs and Images Fields, Admin Notification—New User Creation, Notes into Users and Menu Items, Customized Text Filter, News Feed Flexible Sequence, Translation Edition from Language Manager, Automatic Offline Website after Installation, Customized Offline Website Image, Status Bar, Online User Status, Native ZIP Support, New SEO System Plugin and New Debugging Plugin Interface
3.6.5	Responsive Built-in Feature, Bootstrap, Installation Process Simplified in Only 3 Steps, reCAPTCHA, Content Version Control, Extension Finder, PHP 7 Support, Article Manager—Better Organized and more Available Options, Drag-and-Drop Images, One-click Extension Installation, Update Notifications via Email, Log Folder, Two-Step Authentication and Higher Password Security, ACL into Menu, JLayout Integration, Inverse Cache, Back-end Menu Manager and Custom Fields

Table 2. WCMS: Comparison of WordPress versions and features.

Version	New Features
0.7	New Administration Interface, Private Posts and Geographical Data Support
1.0	Multiple Categories, Comment Moderation, User Creation from Admin Page and Edit Page and Comment link
1.2	Plugin Architecture, Sub-Categories, Post Preview, Unlimited Update Services, Custom Fields, Directory Flexibility, Encrypted Password, Comment Management Tools and Solve Login Problems
1.5	Security Issues (XSS and SQL Injection), Templates, Site Customization and "Save and Continue" Button
2.0	Redesigned Backend, Faster Posting, Image and File Uploading, New User Roles and Capabilities, Database Versioning and Theme and Header Customization
2.1	AutoSave, Spell-Checking into Editor, New Search Engine, Redesigned Login Screen and More Efficient Database Code
2.2	New Widgets, Infinite Comment Stream and Speed Optimization
2.3	Native Tagging Support, New Update Notification, Canonical URLs and Pending Review Feature for Multi-Author Blogs
2.4	More Widgets and Cleaner, Faster and Less Cluttered Dashboard
2.8	Faster to Use, Ease of Installation, Redesigned Widgets Interfaces and Screen Options on Every Page
2.9	Built-in Image Editor, Easier Video Embeds, Global Undo/"trash" Feature and Update and Compatibility Checking
3.0	Lighter Interface, Contextual Help on Every Screen, 1217 Bug Fixes and Feature Enhancements
3.1	Redesigned Linking Workflow, Admin Bar, Post Formats Support, New WCMS Capabilities, New Network Admin and Advanced Taxonomy and Custom Fields Queries
3.2	Refreshed Dashboard Design, New Post Editor Design—Distraction Free and Rotating Header Images
3.3	Drag-and-Drop Uploader and Pointer Tips
3.4	Theme Customizer and Thirty-part Embedded Box
3.5	Re-imagined Flow for Uploading Multimedia Content and Dashboard Style Refresh (Retina Ready)
3.6	Revamped Revisions, Post-Locking, Augmented AutoSave, HTML Media Player and Menu Editor Easier to Understand and Use
3.7	Maintenance and Security Updates While Sleep, Better Global Support and Stronger Password Recommendations
3.8	Modern Aesthetic, Clean Typography, Refined Contrast, High Definition at High Speed, Admin Color Schemes and Smoother Widget Experience
3.9	Improved Visual Editing, Edit Images Easily and Gallery Previews
4.0	New Multimedia Management, Ease of Use of Embedded Multimedia and New Plugin Search Engine
4.1	Auto-Detected Language and Plugin Recommendations Section
4.2	Easier Way to Share Content and Extend Character Support
4.3	Menus in the Customizer, Formatting Shortcuts, Stronger Password Generation when New User is Generated
4.4	Responsive Images, Embed Everything and REST API Infrastructure
4.5	Live Responsive Previews, Custom Logos and Smart Image Resizing
4.6.2	Native Fonts, Inline Link Checker and Content Recovery

Table 3. WCMS: Comparison of Drupal versions and features.

Version	New Features
1.0	Initial Release
2.0	FAQ Module, Multi-Lingual Website Option, Multiple Vhosts, Search Functionality in Administration Pages, Multiple Directories, Sections and Section Manager
3.0	Book, CVS, Help, Page, Moderate, Statistics, System, Poll, Blog and Access Modules
4.0	Blogger API, Tracker and Weblogs Modules and Support for External SMTP Libraries
4.1	Throttle, Profile and Taxonomy Module and Pager Support to the Main Page and Offline Mode
4.2	Support for Clean URLs and Better Installation Instructions
4.3	Support for Configurable URLs, Multiple Sessions per User, Anonymous Session, Mass Node Operations and Optimization of Many SQL Queries
4.4	Automatic Disabling Module or Blocks under Heavy Load and Improve Memory and Footprint Performance
4.5	Reorganize the Navigation Menu, Add Recent Comment Block, Tabs and SubTabs, Possible to Track Forum Topic and Support for Uploading Documents, Database Connections and Using Multiple Inputs Formats
4.6	PHP 5 Compliance, Add Flow Control Mechanism and Categories to RSS Feeds, Contact Module and Security Issues (XSS, DoS, and CSRF)
4.7	Free Tagging Support, Auto-Complete Forms (AJAX), Resizable Text Fields (JS), IP Black-List, Customizable Result Ranking, Support for External URLs and New Security Issues
5	Retooled Administration Page, Web-Based Installer and New Security Issues
6	New, Faster and Better Menu System and Email Notification to Approved, Blocked or Delete Users
7.54	Support for SQLite Database Engine, Limited Login Attempts to Prevent Brute-Force Password Guessing, Drag-and-Drop Positioning for Input Format, Language and Pool Listing, Administration Role, Stronger Password Validator and Time Zone

Table 4. Qualitative comparison of Joomla!, WordPress, and Drupal.

Feature	Joomla!	Drupal	WordPress
Main content type	Websites, online apps	Blog	Blog, e-commerce, online apps
Extension availability	High	Middle	High
Functionality range	High	Middle	High
Extension repository	Distributed	Centralized	Distributed
Documentation	Excellent	Good	Excellent
User community	Very active	Limited	Very active
Ease of use	Simple	Complex	Simple
User role personalization	Middle	Very High	Middle
Manual SEO positioning	Yes	Yes	Yes
Automatic SEO positioning	Extensions	Modules	Plugins and tools

4. Creating a Website with Joomla!, Drupal, and WordPress

To compare the three WCMS under study, and to provide a good understanding of their operation, the same website is going to be created using Joomla!, then Drupal, and finally WordPress. The website should follow the graphical distribution shown in Figure 2 and should have the functionalities enumerated below:

- A homepage slider or banner, based on JavaScript.
- A login module, allowing user registration and creating private areas on the website.
- Social network integration; Twitter and Facebook.
- A multi-language module, content translation based on Google Translator.
- A search module, to find indexed content in the website.
- A contact form.
- Videos.
- Maps.
- A downloads section, customized multi-user downloads.
- A newsletter, so users are aware of recent news by mail
- Events, as a way to place important news into the website homepage.

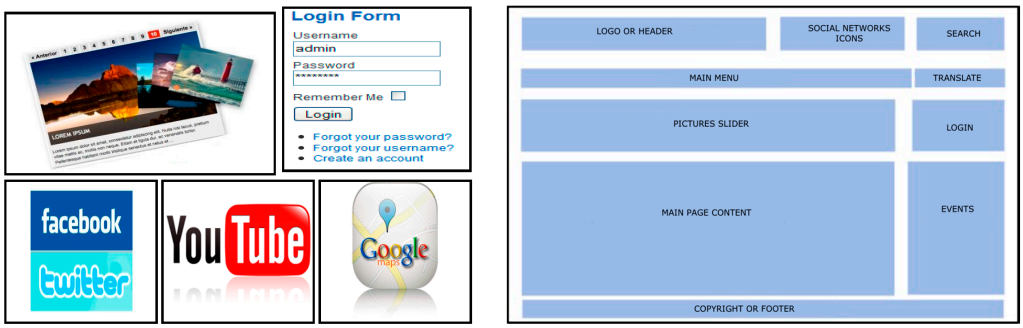

Figure 2. (**a**) Some desirable elements to integrate into the three versions of the website; (**b**) Functionalities provision and graphical appearance.

The first step in starting the website implementation is the installation process. In order to do so, the recommended steps are the same for all WCMS:

- Hire a hosting provider service (e.g., 1and1.com) that includes a database (e.g., MySQL).
- Create the WCMS database.
- Download the WCMS installation package from the official website and extract the files into the virtual directory given by the hosting provider.
- Install the WCMS using the installation wizard, which links with the database.

On the other hand, the WCMS configuration and customization process usually follows the next pattern. Firstly, it is necessary to download or to design the website template, e.g., with a template editor such as Artisteer [33]. Then, the template should be assigned to the website, for instance using the template manager. Secondly, it will be necessary to look for extensions to add the required functionalities. Once found, these extensions need to be enabled and located on the website. Before setting up the template, it is important to check available sections or areas so that those functionalities will be placed in the desired position. Finally, content should be added using the provided editors. Each WCMS usually has its own editor tool for inserting multimedia content or text. Table 5 summarizes the similarities and differences found during the implementation of the websites using Joomla!, WordPress, and Drupal. The Yoo Downtown template was selected for Joomla! and WordPress, whereas the AT commerce template was chosen for Drupal. Note that the selected templates incorporate their own framework for changing block positions and size, page width, fonts, colors, etc. This is highly useful, because it allows a high level of customization, and therefore, a similar visual aspect can be achieved with the three WCMS. Artisteer [33] is a software tool whose goal is to customize the website in a very intuitive way, modifying the visual aspect as required. Once the changes have been done, the user can

export the template as a .zip file ready to be installed in the web-content manager. Regarding header and footer, the Artical module for Joomla! transforms an article made with the editor into a module to be placed wherever the designer considers appropriate on the website. Drupal and WordPress natively integrate this option.

Table 5. Detected similarities and differences between WCMS in implementing website functionalities.

Functionality	Joomla!	Drupal	WordPress
Own template developer	Artisteer	Artisteer	Artisteer
Used template	Yoo Downtown	AT commerce	Yoo Downtown
Header and footer	Artical module	Incorporated into template	Incorporated into template
Picture slider	Nivo	Incorporated into template	Incorporated into template
Social networks	ITPsocialbuttons	Linksalpha website provides social code.	Linksalpha website provides social code.
Translation	GTranslate	GTranslate	GTranslate
Events	JNews	Recent content section	Permanent links widget
Download manager	Jdownloads	File Downloader	Wordpress Download Manager
Newsletter	Mailchimp Website	Mailchimp Website	Mailchimp Website
Contact form	CKForms	Incorporated into kernel	CformsII
Module developer	Jumi	-	-
Entries editor	JCE	Own	Own
YouTube, Twitter and Maps	Inserting HTML code provided by official websites	Inserting HTML code provided by official websites	Inserting HTML code provided by official websites
Search and login	Incorporated into kernel	Incorporated into kernel	Incorporated into kernel

A home-page slider is a good choice for implementing a website with an attractive look. The slider with Drupal needs an additional library called *jquery.cycle.all.js* to work properly. With WordPress, it is necessary to include the returned configuration code into the post (or page) to show the slider. For both WCMS, the picture dimensions and the route where the pictures are located need to be set. On the other hand, Nivo [34] is the module used to incorporate the slider into the Joomla! website. Regarding the social-networks buttons, there are websites such as Linksalpha [35] that return the necessary code to be inserted into a new website section, which has been very useful with Drupal and WordPress. As before, Joomla! does not share the same implementation process, opting in this case for the ITPsocialbuttons module [36]. As expected, when users click on one of these buttons, they will be forwarded to the corresponding social network account to publish a website recommendation. Focusing on the web users, they should be able to choose the website language (for all web content). To do so, we have employed the Google Translator [37] module. By using this module, it is only necessary to indicate which languages will be offered and the default website language. Google Translator is widely considered a powerful tool able to provide a desirable service in the three WCMS under study. Multimedia resources are needed to build an appealing website. The easiest method to insert multimedia from YouTube, Twitter, or Maps is pasting the embedded HTML code provided by official websites into the corresponding website section. Other modules included by default in the website kernel are search and login, with each WCMS having its own implementation. It is highly recommended to read the documentation before implementing the module for each platform.

Events and events management are also of interest in our website. Each WCMS uses a different module or widget to incorporate the events. The Joomla! module has been configured in a way whereby it is possible to select which article identifiers should be highlighted as the most important ones. WordPress uses a permanent link widget to paste the links for interesting events; please note that when a new page is created in WordPress, a permanent link is always returned. However, Drupal uses a so-called *recent content section* to manage which pages from the website are also going to be considered as events. The download manager tool employed by Joomla! is Jdownloads [38]. Mailchimp [39] is the chosen method to create the newsletter on WordPress, Joomla!, and Drupal. Mailchimp websites return predefined code text to be inserted into the website; hence, users can subscribe to the latest news via e-mail. The contact form is the section that requires more configuration and customization because labels, text-boxes, text-areas, submit, and reset buttons should be implemented, as well as

setting configuration parameters, such as the e-mail recipient of the queries. We decided to exchange the default Joomla! editor for one more comprehensive and complete, which allows multimedia integration; namely, JCE. Finally, the corresponding articles are selected to appear on the homepage of the three websites. Based on this work, and from a qualitative perspective, Joomla! offers the most intuitive solution in terms of content management administration and functionalities, Drupal presents the highest complexity in management, and WordPress has the advantage of providing functionalities from its own back-end.

5. Basic Security Analysis

As we mentioned in Section 1, WCMS are becoming a common target for attackers, raising important security concerns. Possible WCMS threats are [40,41]:

- *Data manipulation*: violating data integrity, e.g., Structured Query Language (SQL) injection and parameter manipulation.
- *Confidential data*: when an unauthorized person has access to sorted data, e.g., SQL injection and Cross-Site Scripting (XSS).
- *Phishing*: a special confidential data-gathering method using forms and *spam* mails.
- *Spam*: using email addresses published on the website.
- Execution of code, run scripts, or programs on a web-server using WCMS vulnerabilities.

Analyzing the information shown in Figure 3 [42], JavaScript and PHP are the programming languages with the most vulnerabilities, generating a high amount of security weaknesses. Technologies and frameworks based on Java are less vulnerable; in cases of vulnerability, this is usually because of poor component or framework patching. Packages such as JQuery in PHP and JavaScript are also a source of vulnerabilities. Despite being the most used web-server software, Apache [43] can present several vulnerabilities due to poor configurations and patching policies. At the application layer, 61% of attacks are made through a web-browser. 86% of those web-browser attacks correspond to an XSS attack, thus constituting a vast majority. In an XSS attack, a mischievous user finds a means to insert a malicious code fragment into the website [16]. That is, an XSS attack injects a malicious sequence of commands into a trusted website executed on the visitor's web-browser (without the visitor's knowledge), and therefore, the attacker has access to sensitive user data, such as session tokens and cookies, stored in the browser [44]. Some variations of the XSS attack are the following:

- *Reflected XSS attack*: This attack uses other routes to reach the victims, such as email messages with crafted links or other websites, which reflect the attack back to the user's web-browser. The script is executed by the web-browser because it comes from a "*trusted* server". This type of attack is also known as *Non-Persistent* or *Type-II XSS* [45].
- *Stored XSS attack*: The malicious script is stored somewhere on the web-server (e.g., a database, a forum message, logs, comments, etc.) and is sent to the victim when it requests the query. This attack type is also called *Persistent* or *Type-I XSS* [45].
- *DOM-Based* (Document Object Model) *attack*: In contrast to previous types, in this one, the injection is performed by the user into the web-page when the server script processes user data and injects it back into the website [46].

Another common attack is SQL injection [47]. In this case, a malicious user accesses a website database, and is able to modify it. Databases are fundamental for implementing websites using WCMS. Databases store large volumes of information, in many cases valuable information, and are a common target for malicious users. To perpetrate an SQL injection attack, the website must have an input to write an SQL query. Then, the website should include the attacker input data as an SQL statement into the application without any verification, so this statement could be an SQL query and be run against a database server [48,49]. If the attack is successful, confidential and sensitive data could be

read, modified, or even deleted. Moreover, the attacker could execute administration operations on the database and obtain the database structure information. SQL injection errors occur when the executed program is from a non-verified source or SQL queries are formed dynamically [47]. While an SQL injection is directed to the query function that interacts with the database, XSS attacks take advantage of the output HTML function that sends data to the browser. In any case, most attacks can result in a total system-wide compromise [42].

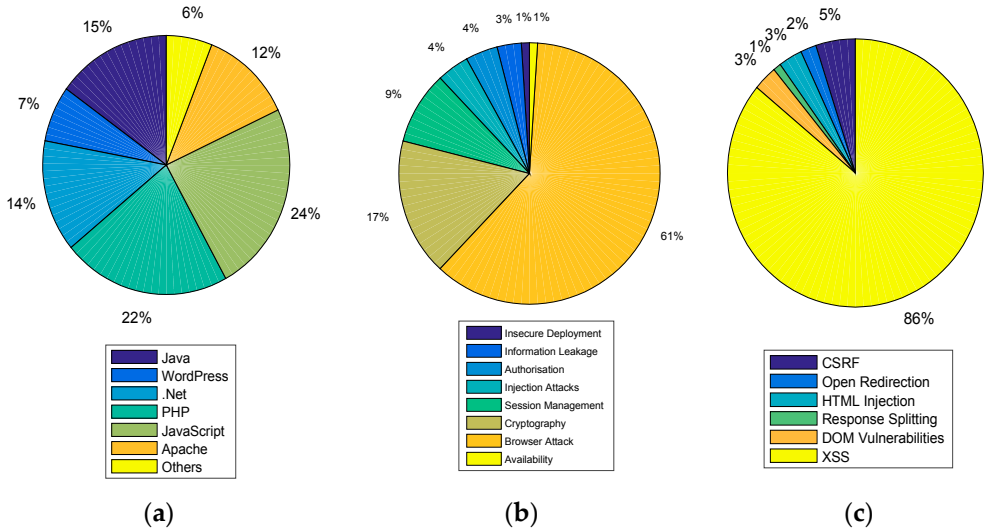

Figure 3. (**a**) Likelihood of a vulnerability being discovered by framework or language; (**b**) Likelihood of a vulnerability being discovered at the application layer; (**c**) Distribution of types of browser attacks. Data extracted from [42].

In this scenario, there are several security recommendations that should always be followed when working with WCMS:

- Make regular WCMS backups (files and database).
- Hire professional hosting providers, which are safer against SQL injection attacks.
- Use the most recent versions of WCMS and plugins.
- Employ specific security plugins, such as JHackGuard for Joomla!, that provide extra security.
- Limit the access to administration files and folders.
- Remove the installation script (*install.php* on Drupal and *installation* folder on Joomla!).
- Modify default passwords and define safe user roles.
- Enable *captcha* for unregistered users avoiding *spam*.
- *Hide* email addresses to avoid unsolicited *spam.*
- Activate URLs-friendly.
- Change the default global website parameters configuration.
- Change the default database prefix during the installation process (if possible).
- Avoid showing sensitive information about the WCMS in the front-end.

Nevertheless, complying with these suggestions is not enough. Vulnerability assessment and penetration testing, a process also known as *pentesting*, have become a valuable tool for evaluating security in a wide variety of systems and devices by simulating attacks. More specifically, the aim of a vulnerability assessment is scanning, i.e., to perform network discovery, network port, and service identification for a system or device, trying to find inadequate security measures and vulnerabilities. On the other hand, penetration testing goes a step further, being defined by the National Institute of Standards and Technology (NIST) as the tester being able to "simulate the actions of a given class

of attacker by using a defined set of documentation (i.e., the documentation representative of what that class of attacker is likely to possess) and working under other specific constraints to attempt to circumvent the security features of an information system" [50]. Depending on the previous knowledge that the tester has about the target, three pentesting categories can be identified; namely, white-box, gray-box, and black-box testing. The *color* of the test is inversely proportional to the amount of knowledge the tester has about the system being evaluated. For instance, in a black-box test no knowledge is assumed about the implementation or configuration of the target, whereas in a white-box test a high amount of information is known (e.g., operating system, source code, etc.).

In this work, we performed a basic security analysis on the websites implemented with Joomla! and Drupal. The website implemented with WordPress was not included in this preliminary analysis, and was left for future work. To carry out the security analysis, we used the Acunetix software [51], an intuitive and automated tool for auditing website security. Acunetix automatically crawls the specified website, carrying out both black-box and gray-box hacking to discover threatening vulnerabilities that can compromise a website and its data. The testing process was as follows. We employed Windows as our operating system. Once Acunetix was installed, the first step was choosing the option *New Scan*, and the program launched a wizard to enter all the testing configuration parameters, e.g., scan type, target, crawling options, scan options, and login. In the scan type window, we opted to scan a single website and introduced the website URL. In the second step, we added some information about the target, such as operating system, web-server, server banner, etc., although most parameters were automatically completed. The crawling configuration was set by default in this scan. In the Scan Options section, we selected the scanning profile with a list of vulnerabilities that the target will be scanned for; in our study, this was XSS and SQL injection. The selected scanning mode was Heuristic, enabling Port Scanning and AcuSensor Technology. The Login section asked about HTTP or HTML authentication to login and scan the protected site. It was necessary to show the scanner how to log in to the website protected area with a username and password. In the last stage of the scan wizard, we could check the scan summary with the scan profile and target information. Once we clicked on finish, the scan was carried out and a complete report was obtained.

The results attained for SQL injection and XSS are shown in Table 6. Analyzing the obtained results, Joomla! and Drupal present low or very low risk, unlike the outcomes obtained in [16]. Testing SQL injection in the website developed with Joomla!, there were 133 total alerts, and their risk level was low or 1; whereas testing XSS, we obtained a total of 122 alerts with the same risk level. In contrast, the website implemented with Drupal presented fewer alerts (78 for SQL injection and 9 for XSS) with a similar risk level. Therefore, all alerts were low-risk warnings, and were similar in nature. Most of them corresponded to content, links, or extensions that had been deleted or uninstalled from the WCMS during the project development and, because of that, the content appears as non-indexed. In that case, Acunetix recommends removing it manually. On the other hand, tests showed a login module alert, since the auto-complete option is enabled, and Acunetix recommends disabling it. Lastly, results for Joomla! indicated that the *Jdownloads* extension allowed users to upload files, which could be a dangerous action. However, this was a design requirement for our website.

Table 6. Acunetix alerts.

	Joomla!	**Drupal**
SQL injection	Total alerts: 133 Risk level: 1 or low	Total alerts: 78 Risk level: 1 or low
XSS	Total alerts: 122 Risk level: 1 or low	Total alerts: 9 Risk level: 0

In addition to the information gathered with the security test, there are other factors that have an effect on our website security. Both Joomla! and Drupal have a large and active user and developer community, always ready to include security enhancements into available versions, and also to report

errors and potential solutions to solve system deficiencies and vulnerabilities. New versions of Joomla! and Drupal correct the errors and vulnerabilities of previous versions. However, both WCMS should give more recommendations to be followed during the installation process, specifically regarding critical folders and files to protect *a posteriori*; above all, a special emphasis should be placed on stating clearly that it is not secure to use the default admin user and the default database prefixes. Instead of this, the documentation (usually) only recommends deleting installation files after the installation process. Joomla! and Drupal have third-party additional modules developed to bestow extra security on websites. One example of such modules is the Taxonomy Access Control, which provides extra security by employing different user roles. Another example is Marco's SQL Injection module for Joomla!, which protects against SQL injection and website file inclusion. Indeed, both WCMS have directives against SQL injection and XSS attacks. Additionally, they have an internal recognition system for checking the uploaded files extensions. Similarly, all the implemented structures and APIs are protected against XSS. Furthermore, third-party components or modules can be set to allow website visitors to upload files such as contact forms, a download manager, photo galleries, etc., having their own security configuration measures ranging from using *captcha* to filtering website visitors by IP address. These modules always register warnings in the general log, because they are external to the WCMS and thus may have some incompatibility, system update, or modification as a result of the version of the module. In addition, these modules and components usually notify the available updates. It is always recommended to update them, so many problems can be solved (or avoided), and the performance can be enhanced. Another security measure is *spam* protection to avoid malicious users or machines attacking the website using this technique. In particular, Joomla! has a weak point in the new user registration process, and Drupal recommends the use of *captcha* in this situation. Lastly, Drupal alerts should be disabled; otherwise, the security weaknesses will be shown. This is not a problem in Joomla!, because warning information is not reachable by website users.

6. Conclusions

WCMS provide a flexible way to show content online, being easy to use and manage. Indeed, nearly 50% of Internet web pages are nowadays implemented using WCMS. Two of the reasons for this success are that they are intended for a very wide audience, without a need for great computer skills, and that they offer a large collection of functionalities. In this work, we have shown how to create three websites with the same graphical design and functionalities using three different WCMS, namely, Joomla!, Drupal, and WordPress. These WCMS are open-source, can be downloaded from official websites, and can be installed using a wizard. The only required resources are a database and a hosting provider. Based on our qualitative study, Joomla! was the most intuitive WCMS, with wider options for functionalities. Likewise, it had the largest and most active user community. Drupal was the most complex in terms of management, but seemed more robust in terms of security and users' roles. Finally, WordPress had the advantage of providing functionalities from its back-end, offered a free hosting option, was not as complex as Drupal, and was the best in SEO positioning. One initial requirement for the three websites was having the same visual aspect. To achieve this goal, we used Artisteer. This software allowed us to create the templates and to obtain the same aesthetic result for the three websites. In the second part of this work, we addressed security in WCMS. We performed a basic security analysis on the websites implemented with Joomla! and Drupal. Specifically, we used the software Acunetix to carry out a gray-box hacking focused on SQL injection and Cross-site scripting attacks. Before the pentesting, we collected and applied several security measures, e.g., the use of *captcha*, IP filtering, events logging, etc. Our findings showed that the tested versions of Joomla! and Drupal were robust against these attacks. However, new threats will constantly emerge, from phishing to specific malware targeted to PHP. As a consequence, we believe that WCSM will continue as a leading-edge research topic in the forthcoming years, with an emphasis on the development of specific security methods for prevention, detection, and recovery in this type of platform.

Acknowledgments: This research was supported by the AEI/FEDER, UE project grant TEC2016-76465-C2-1-R (AIM).

Author Contributions: Maria-Dolores Cano conceived and designed the experiments. Antonio Guillen-Perez and Ramon Sanchez-Iborra contributed with the selection of tools, initial testing, and the state of the art. Antonio-Jose Aledo-Hernandez performed the experiments. Jose-Manuel Martinez-Caro analyzed the data and. together with Maria-Dolores Cano, wrote the paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Netcraft. Available online: https://www.netcraft.com/ (accessed on 14 December 2017).
2. World Internet Users Statistics. Available online: http://www.internetworldstats.com/stats.htm (accessed on 14 December 2017).
3. *How Very Small Businesses Are Utilizing the Internet Today and Future Future Expectations*; GoDaddy LLC & Redshift: Scottsdale, AZ, USA, 2015.
4. Cody, W.F.; Kreulen, J.T.; Krishna, V.; Spangler, W.S. The Integration of Business Intelligence and Knowledge Management. *IBM Syst. J.* **2002**, *41*, 697–713. [CrossRef]
5. Bergstedt, S.; Wiegreffe, S.; Wittmann, J.; Moller, D. Content Management Systems and E-Learning Systems -a Symbiosis? In Proceedings of the 3rd IEEE International Conference on Advanced Technologies, Athens, Greece, 9–11 July 2003; pp. 155–159.
6. McDaniel, R.; Fanfarelli, J.R.; Lindgren, R. Creative Content Management: Importance, Novelty, and Affect as Design Heuristics for Learning Management Systems. *IEEE Trans. Prof. Commun.* **2017**, *60*, 183–200. [CrossRef]
7. Wan, S.; Li, D.; Gao, J. Exploring the Advantages of Content Management Systems for Managing Engineering Knowledge in Product-Service Systems. *Procedia CIRP* **2016**, *56*, 446–450. [CrossRef]
8. Bianco, F.; Michelino, F. The Role of Content Management Systems in Publishing Firms. *Int. J. Inf. Manag.* **2010**, *30*, 117–124. [CrossRef]
9. Shteiman, B. Why CMS Platforms Are Breeding Security Vulnerabilities. *Netw. Secur.* **2014**, *2014*, 7–9. [CrossRef]
10. Internet Users and Penetration Worldwide. 2016-2021—Emarketer. Available online: http://www.emarketer.com/Chart/Internet-Users-Penetration-Worldwide-2016-2021-billions-of-population-change/206259 (accessed on 14 December 2017).
11. Barker, D. *Web Content Management: Systems, Features and Best Practices*, 1st ed.; O'Reilly Media: Sebastopol, CA, USA, 2016, ISBN 9781491908129.
12. McGraw, G. *Software Security: Building Security in*; Addison-Wesley: Boston, MA, USA, 2006, ISBN 0321356705.
13. Hoglund, G.; McGraw, G. *Exploiting Software: How to Break Code*; Addison-Wesley: Boston, MA, USA, 2004; ISBN 0201786958.
14. Symantec. Available online: https://www.symantec.com/security-center/threat-report (accessed on 3 December 2017).
15. Jonsson, E. Towards an Integrated Conceptual Model of Security and Dependability. In Proceedings of the First International Conference on Availability, Reliability and Security (ARES'06), Vienna, Austria, 20–22 April 2006.
16. Meike, M.; Sametinger, J.; Wiesauer, A. Security in Open Source Web Content Management Systems. *IEEE Secur. Priv. Mag.* **2009**, *7*, 44–51. [CrossRef]
17. Aledo-Hernández, A.J.; Guillen-Pérez, A.; Martínez-Caro, J.-M.; Sánchez-Iborra, R.; Cano, M.-D. Sistemas de Gestión de Contenidos Web: Uso Y Estudio Comparativo de Su Seguridad. In Proceedings of the XIII Jornadas de Ingeniería Telemática (JITEL 2017), Valencia, Spain, 27–29 September 2017. (In Spanish)
18. Usage Statistics and Market Share of Content Management Systems for Websites, January 2018. Available online: https://w3techs.com/technologies/overview/content_management/all (accessed on 8 January 2018).
19. Historical Yearly Trends in the Usage of Content Management Systems, January 2018. Available online: https://w3techs.com/technologies/history_overview/content_management/all/y (accessed on 11 January 2018).

20. Patel, S.K.; Rathod, V.; Prajapati, J.B. Performance Analysis of Content Management Systems—Joomla, Drupal, and WordPress. *Int. J. Comput. Appl.* **2011**, *21*, 39–43. [CrossRef]
21. Mirdha, A.; Jain, A.; Shah, K. Comparative Analysis of Open Source Content Management Systems. In Proceedings of the 2014 IEEE International Conference on Computational Intelligence and Computing Research, Coimbatore, India, 18–20 December 2014; pp. 1–4.
22. WorPress. Available online: http://www.wordpress.com (accessed on 1 December 2017).
23. Joomla. Available online: http://www.joomla.org (accessed on 1 December 2017).
24. Drupal. Available online: http://www.drupal.org (accessed on 1 December 2017).
25. W3Techs. Web Technology Surveys. Available online: https://w3techs.com/technologies/overview/content_management/all (accessed on 1 December 2017).
26. Patel, S.K.; Rathod, V.R.; Parikh, S. Joomla, Drupal and WordPress—A Statistical Comparison of Open Source CMS. In Proceedings of the 2011 3rd International Conference on Trendz in Information Sciences and Computing (TISC), Chennai, India, 8–9 December 2011; pp. 182–187. [CrossRef]
27. Jerkovic, H.; Vranesic, P.; Dadic, S. Securing Web Content and Services in Open Source Content Management Systems. In Proceedings of the 2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, Croatia, 30 May–3 June 2016; pp. 1402–1407.
28. Proyecto Webscarab OWASP. Available online: https://www.owasp.org/index.php/Proyecto_WebScarab_OWASP (accessed on 1 December 2017).
29. Tamper Data: The Firefox Add-on. Available online: https://www.lifewire.com/firefox-addon-that-hackers-dont-want-you-to-know-about-2487289 (accessed on 1 December 2017).
30. Hawkins, B.; Demsky, B. ZenIDS: Introspective Intrusion Detection for PHP Applications. In Proceedings of the 39th International Conference on Software Engineering, Buenos Aires, Argentina, 20–28 May 2017; pp. 232–243. [CrossRef]
31. Vasek, M.; Wadleigh, J.; Moore, T. Hacking Is Not Random: A Case-Control Study of Webserver Compromise Risk. *IEEE Trans. Dependable Secur. Comput.* **2015**, *13*, 206–219. [CrossRef]
32. Shivakumar, S.K. *Enterprise Content and Search Management for Building Digital Platforms*; John Wiley & Sons: Hoboken, NJ, USA, 2016, ISBN 1119206812.
33. Artisteer. Available online: http://www.artisteer.com (accessed on 1 December 2017).
34. Vinaora Nivo Slider—Joomla! Extension. Available online: https://extensions.joomla.org/extension/vinaora-nivo-slider/ (accessed on 3 December 2017).
35. Social Media Buttons and Management—LinksAlpha.com. Available online: https://www.linksalpha.com/ (accessed on 3 December 2017).
36. ITP Social Buttons—Joomla! Extension Directory. Available online: https://extensions.joomla.org/extension/itpsocial-buttons/ (accessed on 3 December 2017).
37. Google Translate. Available online: https://translate.google.com/ (accessed on 3 December 2017).
38. jDownloads! Download Manager for Joomla! Available online: http://www.jdownloads.com/ (accessed on 3 December 2017).
39. Mailchimp. Available online: http://www.mailchimp.com (accessed on 1 November 2017).
40. Newman, R.C. Cybercrime, Identity Theft, and Fraud. In Proceedings of the 3rd Annual Conference on Information Security Curriculum Development, Kennesaw, Georgia, 22–23 September 2006; ACM Press: New York, NY, USA, 2006; p. 68.
41. Tanenbaum, A.S.; van Steen, M. *Distributed Systems: Principles and Paradigms*; Prentice-Hall: Upper Saddle River, NJ, USA, 2002.
42. *2016 Vulnerability Statistics Report*; EdgescanTM Portal: Dublin, Ireland, 2016.
43. Apache. Available online: https://www.apache.org/ (accessed on 14 December 2017).
44. Yusof, I.; Pathan, A.-S.K. Mitigating Cross-Site Scripting Attacks with a Content Security Policy. *Computer* **2016**, *49*, 56–63. [CrossRef]
45. Cross-Site Scripting (XSS)—OWASP. Available online: https://www.owasp.org/index.php/Cross-site_Scripting_(XSS) (accessed on 3 December 2017).
46. CWE—CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting') (3.0). Available online: http://cwe.mitre.org/data/definitions/79.html (accessed on 3 December 2017).
47. SQL Injection—OWASP. Available online: https://www.owasp.org/index.php/SQL_Injection (accessed on 3 December 2017).

48. SQL Injection (SQLi)—Acunetix. Available online: https://www.acunetix.com/websitesecurity/sql-injection/ (accessed on 3 December 2017).
49. Shar, L.K.; Tan, H.B.K. Defeating SQL Injection. *Computer* **2013**, *46*, 69–77. [CrossRef]
50. National Institute of Standars and Technology. Assessing Security and Privacy Controls in Federal Information Systems and Organizations: Building Effective Assessment Plans. *NIST Spec. Publ. 800-53A* **2014**, *4*, 1–487. [CrossRef]
51. Acunetix. Available online: https://www.acunetix.com/ (accessed on 3 December 2017).