Vol. 11(1), pp. 1-12, January-June 2022
DOI 10.5897/JIIS2021.0120
Article Number: BF428EA68483
ISSN: 2141-6478
Copyright ©2022
Author(s) retain the copyright of this article
http://www.academicjournals.org/JIIS

ACADEMIC JOURNALS
expand your knowledge

Journal of Internet and Information Systems

Full Length Research Paper

Model for security controls in web content management system

Alex Maraga*, F. Mzee Awuor, and James Ogalo*

School of Information Science and Technology, Kisii University, Kisii, Kenya.

Received 7 June, 2021; Accepted 7 January, 2022

Web content management systems (WCMS) are systems used in creating, publishing, customizing and designing website services by web administrators toward delivering user-centric web applications and services. Such applications include Joomla, Drupal, and WordPress, which have found their usage in various institutions including universities and colleges, non-government and government institutions. While these WCMS provide easy access to web services to the users, they are vulnerable to security breaches and threats. This study sought to ascertain whether web administrators are aware of security concerns in WCMS. The objective of this paper was to identify widely used WCMS and the level of awareness of security breaches on these applications by web administrators. The study employed the census method and presented the results of 40 Web Administrators sampled from four public universities within Nairobi County. We then presented a security control model informed by the data analysis towards proactive mitigation of the potentials of WCMS security threats. The model sought to integrate security measures such as security awareness in the design of WCMS to curb threats related to SQL injections, XSS attackers and unauthorized access of information, and to assist the web administrator in choosing suitable WCMS applications that meet their users' preference.

Key words: Web content management systems, security awareness, web administrators, Drupal, WordPress, Joomla.

INTRODUCTION

Information systems security continues to be a growing concern for learning institutions such as universities as they embrace Internet Technologies to offer anytime anywhere learning experience to their learners such as massive online and open courses (MOOC), the traditional eLearning approaches among others. Notably, these learning solutions are more often built on open-source (WCMS) that are managed locally at these learning institutions by their dedicated personnel in their information technology (IT) departments, mostly web administrators. Being open-source, however, make these WCMS very susceptible to security threats. Thus, this study sought to establish the level of WCMS security awareness in open source WCMS among the web administrators, to identify the security threats and breaches common in these WCMS, and to derive a model to mitigate these security threats. We hope that this can provide a strategy for controlling these security vulnerabilities in WCMS from both the users and web application ends perspectives. It is noticeable that most

*Corresponding author. E-mail: amaraga24@gmail.com.

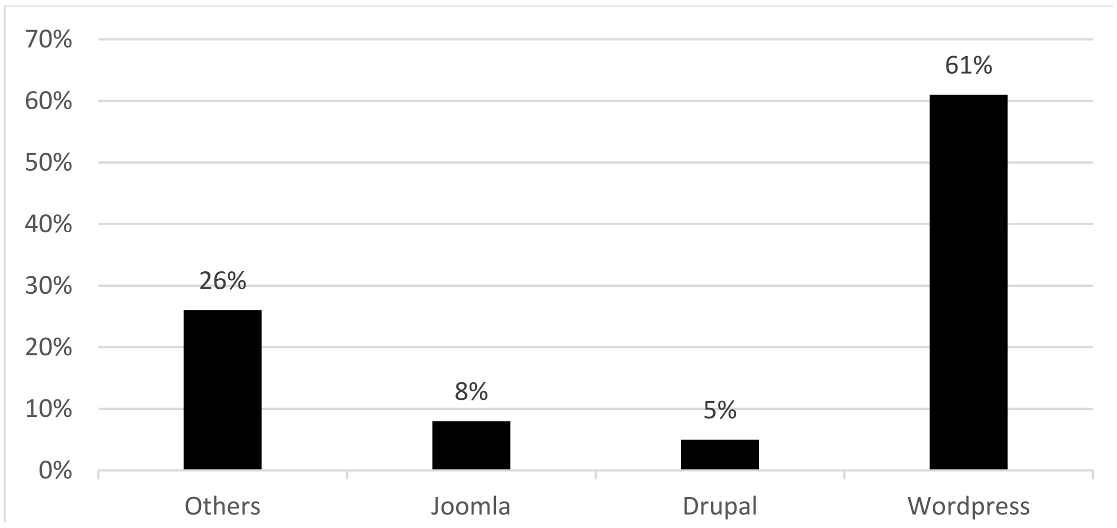

Figure 1. Current WCMS Market Share.

research has focused on general network and Internet security, software and application security, and computer system security; there is no much attention on WCMS security in particular while WCMS are key drivers for the beyond web 2.0 applications. For the sake of illustration, this paper focuses on the top three popular WCMS in Kenya (Martinez-Caro et al., 2018), that is, WordPress, Joomla, and Drupal. According to Cassetto (2014), the three most popular WCMS WordPress, Joomla, and Drupal have something in common globally that they are frequently used. Another unfortunate similarity is that they are the most targets for hackers. To satisfy globally the frequently used WCMS, Handova (2019) says that a lot of people assume that since Joomla, WordPress, and Drupal are highly recognized and popular, they must be robust in security. He gave WordPress as an example that has more than 14,000 known vulnerabilities at its core, plugins, and modules. He concluded that more than 90% of the top many websites are based on Drupal, Joomla, and WordPress (Figure 1).

For example, in an attempt for WordPress to keep its users up-to-date on security patches to safeguard the users, it usually notifies its users through the dashboard of the administrator panel such that the users cannot ignore or fail to see such notifications when they log into WordPress. This feature among others, arguably, has made WordPress to be ranked the most user-friendly WCMS (Filotrani, 2018). As noted by Williams et al. (2015), the installation of these new releases is always simple and just point-and-click away. As many users prefer WordPress to the other WCMS, this in itself makes it vulnerable to security threats (Mesa et al., 2018). Just as viruses, this large number of users and bulk installations make WordPress a key target of hackers. To this end, we can argue that security awareness of WCMS by web administrators is essential towards building in them the capacity to safeguard users of WCMS against security threats and vulnerabilities. Last year it was discovered that cybercriminals discovered WordPress security lapses, being over 170,000 (Cassetto, 2014). Also Infrastructure (2016) mentions that Joomla is one of the most widely used WCMS globally. It is PHP-based and allows the rapid placement of active content on websites. It is known for its simplicity of deployment and custom while offering widespread structures and plugins. But, like many other large packages, Joomla has experienced several vulnerabilities in recent years and, if left unpatched, can represent a risk for site owners, and any other Internet users.

According to Cyber Security Report (2016), the wrong choice of WCMS coupled with a lack of security awareness among technology users significantly impedes the maximum exploitation of WCMS applications in universities. Such challenges include lack of skills in cyber and digital security, lack of security awareness among the user community e.g., the culture of not giving much consideration to security threats while online among others (Piper et al., 2015). Another security challenge identified as facing WCMS is the fact that technology and cybersecurity landscapes change rapidly (Peltier, 2016). According to Peltier (2016), the common methods used for an attack on WCMS include ransomware, SQL injection, malware, denial of service, database transaction manipulation and cross-scripting (XSS) attacks.

Towards exploring the WCMS security awareness problem, this study employed the census method to collect both qualitative and quantitative data which were analyzed using both descriptive and content analysis. From the study findings, it can be concluded that most web administrators in public universities in Kenya (in the case of Nairobi County) use open-source WCMS (mostly

Drupal, followed by Joomla and finally WordPress). The choice of Joomla and WordPress is owed to their perceived ease of use and user-friendly interfaces while choice on Drupal is due to its perceived security features. Thus, it is notable that important features for a WCMS according to the web administrators revolve around their security and usability. Others include their ability to provide an opportunity for user customization and personalization, and a rich online user community for support and troubleshooting. Moreover, the study results also showed that all the surveyed web administrators had encountered or experienced or know a colleague who had experienced unauthorized entry into their WCMS application. Among these web administrators, the preferred security control measures to curb such incidences include those against SQL injections and parameter manipulation. In other words, the common security measures deployed mostly against such adversaries from gaining unauthorized access to confidential data include protection from SQL injections and XSS attacks and protection from unauthorized access to confidential data. Besides, an attacker gathering confidential data by sending emails to people and adversaries exploiting WCMS was identified as an important measure against the WCMS attack. Such measures are required to ensure confidentiality, integrity, and availability of information (CIA), in addition to the provision for backup and documentation. To this end, this paper also proposes a security model that integrates these WCMS security requirements in addition to a mechanism to promote security awareness among the WCMS. In a nutshell, the contribution of this paper can be summarized as follows:

1. We established the security threats and breaches in WCMS in the context of web administrators in public universities in developing nations, the case of Kenya, and evaluated the level of preference of the different WCMS applications among users. We endeavour to explore the underlying reasons that inform this preference and reported on these guidelines – that is, perceived security levels, ease of use, and freedom for customization and personalization.
2. Besides, we established the level of security awareness among the web administrators who use WCMS (that is, Drupal, Joomla, and WordPress) in the local public universities in Kenya and reported on how they safeguard their users against security threats and vulnerabilities.
3. Informed by these findings, we derived a security model towards making WCMS robust to security threats and present a discussion on the implementation and operability of the model.

The rest of this paper was organized as follows: Next Section provides the Literature Review while Sections 3 and 4 present the Methodology and Results respectively. Discussion is presented in Section 5 whereas Section 6 concluded this paper.

Literature Review

Majority of companies spend resources on securing their main systems and applications; they neglect to also review the security of the WCMS platform because they underrate them that nobody is interested in hacking the blog. They concentrate more often than not on the technology than the content itself that is interesting to hack. This is why WCMS security needs attention as well (Almroth, 2018). That is why Cyber Security Report (2016) defines mitigation of security threats facing WCMS as the ability, capability, or state where data and communication systems are protected against damage, unauthorized user or alteration, or manipulation. Due to lack of security awareness among most users, a critical challenge is to guarantee that users of WCMS applications are always safe against attacks such as SQL injections, XSS and, Cross-Site Forgery Requests (while there could exist other attacks on WCMS). This study focused on these three due to their prevalence – the user is referred to Cyber Security Report (2016) for an exhaustive discussion on general WCMS attacks. Hence in XSS attacking process, it takes place in this process: first, an attacker discovers a vulnerable website that has enabled a script injection. Secondly, an attacker launches a malicious script that steals each visitor's session cookies. And then on the third stage, for each website visit, a malicious script is activated (Figure 2). Then lastly in the fourth stage, the visitor's session is sent to the attacker.

As most websites (Kasli and Kaur, 2015) currently store information as data, they mostly rely on the underlying database and some basic functions such as create, read, update and delete records for data manipulation. For instance, structured databases use structured query language (SQL) to manipulate and perform these functions. An attack such as an SQL injection attack happens when the attacker manipulates the query data to modify the query logic to manipulate the back-end database (Kasli and Kaur, 2015). This causes a WCMS application to generate and send a query that functions differently from that intended by the programmer. For example, if a database contains user names and passwords, the application may contain code such as the following: query = "SELECT * FROM accounts WHEREname='"+ request.getParameter ("name") + "' AND password='"+ request.getParameter("pass").This code generates a query intended to be used to authenticate a user who tries to log in to a website (such as a WCMS application). However, if a malicious user enters "bad guy" into the name field and 'OR' a'='all into the password field, the query string becomes: SELECT * FROM accounts WHERE name='badguy' AND password='' OR 'a'='a' whose condition always evaluates

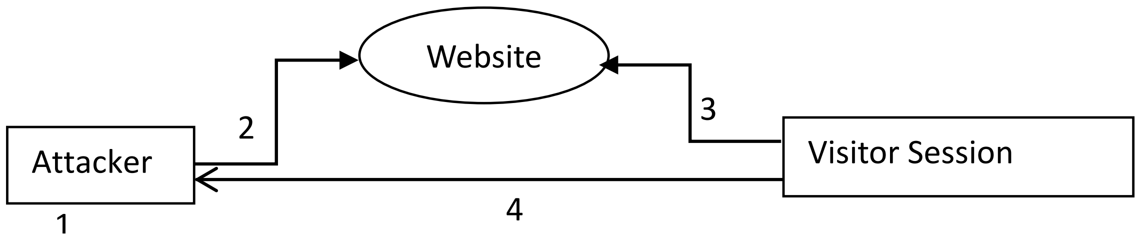

Figure 2. XSS attacking process.

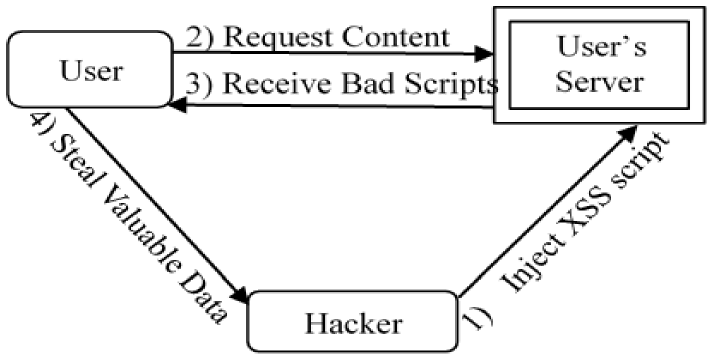

Figure 3. Cross-site scripting model.

to true, and the user will bypass the authentication logic (Figure 3). Notably, the problem goes beyond simply failing to check input that is incorporated into a query. Even web applications that perform some checks on every input may be vulnerable. For example, if the application forbids the use of the single-quote in the input (which may prevent legitimate inputs such as - O'Brian), the SQL-injection attack may still be possible because numeric literals are not delimited with quotes (Uwagbole et al., 2017). This is illustrated in Figure 2 above.

The bottleneck in the SQL injection attack is that web applications generally treat input strings as isolated lexical entities. That is, input strings and constant strings are combined to produce structured output (that is, SQL queries) without regard to the structure of the output SQL language. Several approaches to dealing with the Structured Query Language Command Injection Attacks (SQLCIA) problem have been proposed, but so far, no formal definition for SQLCIAs has been derived yet (Ali et al., 2015; Steiner, 2014). Hence, the effectiveness of these approaches can only be evaluated based on counter-examples, empirical results, and informal arguments as shown in (Ali et al., 2015). For instance, Steiner (2014) argues that WCMS may need to be able to sanitize the input queries being issued to the back-end database towards supporting the integrity of the database.

According to Alwan and Younis (2017), SQL injections are improperly filtered input that is sent to the SQL server. This input could be SQL queries that could access sensitive data. An adversary could use escape characters to include SQL queries in an input field. For instance, if a malicious user appends '1'='1' to an input field, this could lead to unwanted disclosure of data since the boolean expression OR '1'='1' is always true, and thus, the query in which the expressions are appended would also be allowed. These mechanisms examine input strings to prevent exploits of escape characters. For example, PHP uses the function mysql_escape_string to mask all kinds of special characters (Priyatna et al., 2014).

Another common attack, as mentioned earlier, is the XSS which occurs when hackers inject their codes into the output application of a web page that is displayed as part of the web page content in the browser. The unsuspecting web page viewers could then end up sharing their personal and sensitive information which gets stolen by this code as it executes automatically when the page displays (Gupta and Gupta, 2017). This code injection, which is similar to SQL injection in web application security can be used in three different ways, that is, stored XSS‖, reflected XSS, and -dom-based XSS. This is illustrated in Figure 3.

In the stored XSS (so-called persistent XSS), an attacker can inject malicious code into the page persistently such that the code gets stored in the server (Nithya et al., 2015). Ordinarily, the code is stored on the page which gets displayed to the visitors later on such that if a visitor goes to a page that is embedded with XSS attacking code, the code executes on the visitor’s computer and gets to infect the computer. Hackers usually post these codes in articles in forums or blogs that target and attack unsuspecting readers (Parsons, 2017). If stored XSS vulnerability is successfully exploited by hackers, it will persistently attack the users until the web administrator identifies and removes it (Svensson, 2016).

The security model (shown in Figure 2) proposed by Yousra (2013) is built on the premise of curbing XSS - being the common attack on various WCMS. XSS is generally believed to be one input sent to the server as part of the request as discussed earlier. The REFLECTED XSS (also called non-persistent XSS) is a temporary form of attack (Johns and Pfistner, 2017) since it does not inject code into the server but rather makes the server use the injected malicious code to immediately generate a page and then sends this temporary page’s URL to anyone that the attacker wants to attack. Thus, if the user clicks this URL, the malicious code in this temporary page executes. This attack is based on user trigging. This makes it more difficult to deploy unless the hacker can convince the user to trigger the dangerous URL. So the hacker has to find a method to make the URL look like a trusted website’s URL (Sarmah et al., 2018).

Notably, hackers can encode the URL into HEX value or other types of code for the URL to look as true and reliable (Nithya et al., 2015) as possible such that the user could get duped to believe that there is no virus command inside and clickable links or buttons. For example, in Figure 2, Google is a famous and reliable website. If the Google search engine has REFLECTED XSS, the hacker can inject malicious code into the URL and encode the URL. (Many tools on the Internet can provide the service of encoding the code from ASCII to decimal ASCII, hexadecimal, or other types. Interested reader on encoding is referred to (Sarmah et al., 2018). After finishing encoding the URL, the hacker could send this URL to trick the user into clicking and also using some tricks which can attract the user to click.

The Document Object Model (DOM)-based XSS attack is another type of XSS vulnerability that is commonly used by hackers as well (Gupta and Gupta, 2017). DOM is a platform and language-neutral interface which uses scripting or program to modify the content and update the structure and style of documents. It is widely used in HTML and XML in Web 2.0. DOM in HTML can generate a tree-structure of HTML documents. Therefore, each branch of the tree can be easily controlled and modified by DOM. However, DOM allows the scripting or program to change the HTML or XML document, the HTML or XML document can be modified by a hacker’s script or program (Jakobson, 2014). Therefore, DOM-based XSS uses DOM‟s vulnerability to implement the XSS. This type of XSS vulnerability is different from the REFLECTED or STORED XSS attack as it does not inject malicious code into a page. So, it is the problem of the insecure DOM object which can be controlled by the client-side on the web page or application. For this reason, hackers can let the attack payload execute in the DOM environment to attack the victim’s side. Unfortunately, the usual defenses for XSS vulnerability hardly work in this type of attack (Gupta and Gupta, 2017). This vulnerability occurs when an application takes untrusted input data and sends it to the web browser without proper validating. An adversary could exploit this vulnerability by including script code (e.g., Javascript) on a web page. Proper mechanisms for always treating output as text are necessary to prevent script code from being executed in a browser (Deshpande et al., 2017).

The security model (shown in Figure 2) proposed by Yousra (2013) is built on the premise of curbing XSS - being the common attack on various WCMS. XSS is generally believed to be one input sent to the server as part of the request as discussed earlier.

A Cross-site Forgery Request (CSRF) is an attack where a user performs unwanted actions on a vulnerable application in which s/he is currently authenticated in (Gupta and Gupta, 2017). An adversary could trick a user to load a page with a malicious request, and then inherit the victim’s identity and privileges to perform actions on the vulnerable application. The application would think that the requests made by the adversary are legitimate requests from the victim e.g., sending a link via email or chat, which could fool the victim to open the malicious website. Links and forms that involve state-changing functions are the main targets for CSRF attacks. If the victim visits one of the malicious websites, while still is authenticated at domain.com, the attacker could forge a request that includes the victim’s session information. As a result, the vulnerable application authorizes the malicious request because it appears to be the victim. One of the ways to prevent CSRF attacks is to include an unpredictable token in the body or URL of every HTTP request (Gupta and Gupta, 2017). These tokens should be unique to each user session, or unique to each request. A good practice is to include the token in a hidden field. Then the token is sent in the body of the HTTP request, thus, not exposed in the URL. The token could also be included in the URL. However, this could be compromised due to exposure to adversaries (Gupta and Gupta, 2017).

METHODOLOGY

A pilot study was conducted before the main study to ensure that the data collection tools developed for this study were suitable in content and length and that the respondents were interpreting the questions in the manner intended. For the pilot sample, Mugenda and Mugenda (2008) recommends 1% of the study population as being fit for a statistical test of instruments. Thus, the pilot study was carried out amongst the five sampled colleagues from the ICT

Department at Kisii University. The pilot survey was conducted to find out if the respondents could respond to the questions without difficulty. They were also asked to evaluate the questions for relevance, comprehension, meaning, and clarity. From the pilot study results, the ambiguous questions were refined and restructured accordingly. After analyzing the sample-filled questionnaires from the repeat pilot study, it was confirmed that the questions were well understood and that the respondents were providing the intended responses and, thus, implied that they could be used to collect the intended data during the study.

This study used a descriptive survey research design to collect data-rich in detailed description e.g., of events and phenomena. This design was appropriate since it allowed for the establishment of opinions and knowledge about content management security awareness among the sampled respondents within a short time using quantitative methods. Also, an experimental study design was used to derive the proposed model towards the mitigation of vulnerabilities and threats on WCMS. A group of ICT departmental members was recruited in a pre-experimental research design to assist in evaluating the security concerns of WCMS from cause to effect.

The population of interest comprised of public universities in Nairobi County (Nairobi County has four public Universities that is, Nairobi University, Multimedia University, Kenyatta University and the Technical University of Nairobi. C.f. (CUE HR Report, 2016). The unit of analysis in this study was the individual web administrators serving in the sampled public universities' main campuses (Johns and Pfistner, 2017).

The selected universities were visited and the questionnaires were administered to the respondents. The respondents were assured that strict confidentiality would be maintained in dealing with their data. The completed questionnaires were collected the same day they were administered. The study used census methods of collecting data. From the university administration, heads of ICT departments were identified and approached for their approval. Once their approval was obtained, their assistance was sort in identifying the web administrators. This was done to ensure that the sample used in this study was valid and measurable regarding the analysis of the result.

Before processing the responses, the completed questionnaires were confirmed in number for completeness and consistency. The data were then coded to enable the responses to be grouped into various categories. Data collected were both qualitative and quantitative as mentioned earlier. The open-ended questionnaire ensured the collection of both qualitative and quantitative data. Qualitative data were abstracted from the questionnaire and typed separately using a word processor. Qualitative data were analyzed through content analysis similar to Neuendorf (2016) while quantitative data were analyzed by descriptive analysis. Data were analyzed as per the objectives of the study. The descriptive statistical tools that is, SPSS was used. The findings were presented using tables and charts.

To enhance the content validity of the questionnaires, only appropriate and adequate items relevant to the research questions were included. According to Mugenda and Mugenda (2003), the procedure of assessing content validity is to seek expert or professional advice in that particular field. In this regard, this study leveraged the opinions of three content experts and project supervisors that validated the research instruments. Their comments and suggestions on restructuring and rephrasing questions that appeared vague and ambiguous were taken into account and effected by ensuring that the instruments collected valid data.

The researcher used Split-Half Reliability intending to determine how much error in the test score was due to poor test construction. This was to assist in infer the reliability of the test study. The reliability index was calculated using Cronbach's alpha coefficient similar to Mugenda and Mugenda (2008).

$$\frac{k}{(k-1)}\left[\frac{\sum_{i=j}^{k} cov(x_i x_j)}{Var(x_0)}\right] = \infty \quad (3.1)$$

$$\frac{k}{(k-1)}\left[1-\frac{\sum_{i=j}^{k} cov(x_j)}{Var(x_0)}\right] = \infty \quad (3.2)$$

$$\frac{k}{(k-1)}\left[\frac{\sum_{i=j}^{k} cov(x_i x_j)}{Var(x_0)}\right] = \frac{k}{(k-1)}\left[1-\frac{\sum_{i=j}^{k} cov(x_j)}{Var(x_0)}\right] = \infty \quad (3.3)$$

$$\sum var = 6.52083 \quad (3.4)$$

$$= \left[\frac{40}{40-1}\right] \times \left[6.52083 - \frac{2.14583}{6.5083}\right]$$

(1.025641026) (0.6709271672 = 0.69

Reliability Coefficient=0.69

According to Taber (2017), reliability coefficients of 0.6-0.7 are deemed acceptable since positive results from the calculation signify that the data collected are a true picture on the ground. The pilot study revealed a reliability coefficient of 0.69 which falls within the acceptable limits of reliability efficiency.

RESULTS AND FINDINGS

Commonly Used Open Source WCMS

The study also investigated the commonly used open-source WCMS. This was done on three levels. The first level required the respondents to indicate the open-source WCMS they commonly used while the second level required them to give an opinion on the open-source WCMS they perceived to be commonly used by other web administrators. The third level sought to understand from the respondents the form of open-source WCMS they would recommend for others including their peers. Respondents were allowed to indicate more than one open-source WCMS since some sampled universities' website sub-domains run in different types of WCMS from the main domain. These findings are shown in Table 1.

The majority of the respondents indicated that they commonly used Drupal (35%). About 27.5% of the respondents indicated that they commonly used both Joomla and Drupal while 12.5% indicated that they used Joomla in most instances. Also, about 15% indicated that Joomla, Drupal, and WordPress were commonly used by web administrators. On recommended open-source WCMS, 40% of the respondents recommended Drupal, 17.5% WordPress and 12.5% Joomla and Drupal, Joomla and WordPress for both. Only 10% recommended Joomla.

The findings of this study indicated that the most commonly used open-source WCMS is Drupal while the most perceived to be used open-source WCMS is WordPress. That notwithstanding, the most recommended

Table 1. Commonly Used Open Source WCMS.

	Open Source WCMS	F	%
Form of open WCMS commonly used by respondents	Joomla	4	12.5
	Drupal	14	35
	Wordpress	3	7.5
	Joomla and Drupal	11	27.5
	Joomla and Wordpress	3	7.5
	Joomla, Drupal and Wordpress	4	10
	Joomla	8	20
	Drupal	3	7.5
Opinion on commonly used open WCMS by web administrators	Wordpress	14	35
	Joomla and Drupal	7	17.5
	Joomla, Drupal and Wordpress	6	15
	Joomla	4	10
	Drupal	16	40
Open WCMS recommended to be used by other	Wordpress	7	17.5
	Joomla and Drupal	5	12.5
	Joomla and Wordpress	5	12.5
	Joomla, Drupal and Wordpress	3	7.5

Table 2. Level of Awareness of Security Concerns in Open Source WCMS.

Security Concerns	Fully Aware	Slightly Aware	Some what	Not At all	Mode
SQL injections and parameter manipulation	40	30	17.5	12.5	1
An adversary gaining unauthorized access to confidential data by utilizing SQL injections or XSS attacks and having access to confidential data	40	32.5	15	12.5	1

open-source WCMS was Drupal. This finding points to an understanding that while the respondents commonly used Drupal, they perceived the use of WordPress to be common among web administrators.

According to Augustyniak et al. (2005), WordPress is simple to install and use, and has been popular for the same reasons of easy use and also easy installation. The findings of Augustyniak et al. (2005) could, however, point to a both-sided interpretation. The first interpretation could be based on the superior position of university web administrators concerning other web administrators (that is, those serving in other organizations outside the academia or in lower-level institutions). As indicated from the opinion results Drupal is popular for security reasons.

The second direction of explanation could also be based on the expositions of the above scholars viewed along with a time frame. It is, thus, possible that WordPress has been popular in the past based on its simple user interface and that more advanced institutions are moving away from it to Drupal from the opinion of the respondents termed to be more secure but not easy to use. The use of Drupal as opposed to WordPress could also be linked to awareness of its vulnerability to security attacks.

Level of Security Awareness of Open Source WCMS Among Web Administrators.

All respondents indicated that they were aware of the security concerns of open source WCMS. Levels of awareness were investigated through a five-point Likert scale defined using the following levels: To a very small Extent (SE) = 1; To a Small Extent (NVI) = 2; To some Extentǁ. (SE) = 3; To a large extent (LE) = 4; and To a very large extent (VE) =5. About 50% of the respondents indicated that they were aware of the security concerns of WCMS to a very large extent. About 27% also indicated

Table 3. Important Security Concerns for Development of WCMS

	Quite important	Very important	Extremely important	Mean
Importance of confidentiality in ensuring security of WCMS	33.3	44.4	22.2	33.3
Importance of integrity in ensuring Security	22.2	55.6	22.2	33.3
Importance of backup in ensuring security of WCMS	11.1	44.4	44.4	33.3

their levels of awareness to a large extent. The study also investigated the specific security concerns of open source WCMS. A five-point Likert scale on levels of awareness was used. As indicated in Table 2, most respondents indicated that they were fully aware of all the security concerns that were being investigated (Mode=1). This finding indicates that security concerns for open access WCMS including SQL injections and parameter manipulation, and adversary gaining unauthorized access to confidential data by utilizing SQL injections or XSS attacks and having access to confidential data, an attacker gathers confidential data by sending emails to people, pretending to be a service they use and adversaries exploiting WCMS.

Security concerns for open-source WCMS

These results are shown in Table 3. 50% of the respondents indicated that the security of the system was extremely important for open WCMS. The results of the respondents in the same table indicated that the importance of confidentiality in ensuring security, WCMS was seen to be very important (that is, at 44.4%), integrity in ensuring security was shown to be 55%, backup in ensuring the security of WCMS at 44.4% as well as documentation at 50%. This implies that better documentation on security issues ranks a type of WCMS in a better percentage. On the other hand, time taken for installation was considered as not important that is, 37.5%. The popularity of the WCMS was indicated as not very important (47.5%). The respondents recorded higher scores for the security of the WCMS, usability of the interface, provisions for advanced personalization, the existence of developing communities, and support and consultancy. This indicates that the most considered security features desirable in an open WCMS. Further, the findings indicate that the time taken for the installation of the WCMS was not important because it does not contribute to any vulnerability as the security of WCMS is concerned.

Security Controls to Mitigate the WCMS Security Threats

To investigate security control measures necessary for the development of a security control model to proactively mitigate the WCMS security threats, the study focused on the occurrence of unauthorized entry into WCMS. Other factors investigated included the level of awareness of security concerns in open-source WCMS as well as important security concerns for open-source WCMS. 71% of the respondents reported having experienced unauthorized entry into WCMS. The respondents were further asked to indicate the extent to which they had ever experienced unauthorized entry into WCMS.

The results are captured in Figure 4; majority of web administrators (71%) indicated that they rarely experienced unauthorized entry into WCMS. Those who indicated that they experienced unauthorized entry into WCMS were 39%. This means that the attackers successfully managed to penetrate WCMS manned by web administrators. None of the respondents indicated that they had never experienced unauthorized entry into WCMS. This finding indicates that unauthorized entry into WCMS was an occurrence that web administrators experience.

Important Security Concerns for Developing WCMS

Table 3 shows findings on important security concerns for the development of open WCMS. The findings are analyzed to indicate their levels of agreement on their importance. This was done by the use of a three-point Likert scale. A measure of the frequency of occurrence of a response (mean) was used to indicate the direction of response. Further, the respondents indicated that all the security measures investigated were very important in the sense that this measured how keenly they took into consideration that WCMS is secure (Mean=33.3 for all variables). This points to an understanding that confidentiality is necessary for ensuring the security of WCMS, that integrity is important in ensuring the security of WCMS, that backup is important in ensuring the security of WCMS and that documentation is important in ensuring the security of WCMS.

DISCUSSION

The results from Table 1 imply that most website administrators in public universities in Kenya use open-

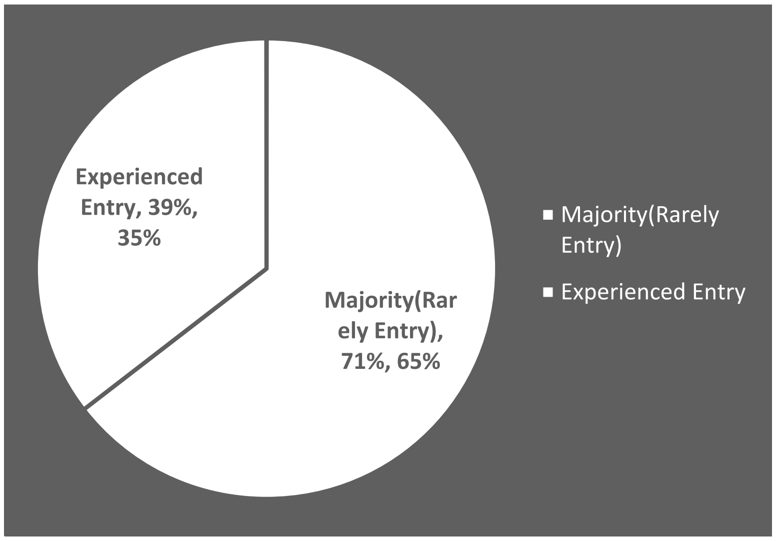

Figure 4. Experience of Unauthorized Entry into WCMS.

source WCMS. This means that there should more focus on the security aspect of WCMS to safe protect the users and applications supported by WCMS from security breaches and threats. Regarding usability, WordPress and Joomla were preferred by users while the users of Drupal preferred it to the others due to their view that it had superior WCMS security measures. This means that some web administrators are not aware of the security proposition of Drupal and as such deployed WCMS based on perceived ease of use and general usability. The study also revealed that all the web administrators sampled have heard or experienced (attempted) unauthorized access into their WCMS, which is a confirmation of security concerns in WCMS. The preferred security control measures by the users revolved around protection against SQL injections and XSS attacks, and as such, generally imply that they are protecting themselves against adversaries gaining unauthorized access to confidential data by utilizing SQL injections or XSS attacks and having access to confidential data.

It is evident that the security of the WCMS applications used by the public universities majorly depends on the choice of the WCMS. From the results, it shows that different WCMS have different levels of vulnerabilities, hence, the use and choice of secure WCMS by web administrators depend on the ease and level of their security awareness. In this regard, web administrators need to be brought to the attention of the underlined threat and vulnerability levels of each WCMS.

Security Control model

To this end, we develop a model and present it as a solution to choosing a more secure WCMS. By taking into consideration the developed security control model containing all integrated security indicators and components as indicated in (Figure 5). These considered components include system security, developing communities support, advanced personalization, confidentiality, integrity, backup provision, and vulnerabilities validation (e.g., CSRF, SQL injections, XSS).

From the study findings, security awareness of open-source WCMS includes security of the system and its usability. According to Black et al. (2018), the best WCMS must consider the security of the system and its usability- that is, simplicity, popularity, and support such as being simple to use and able to accommodate differences in user backgrounds as shown in Tretten and Karim (2014).

The scholars also advise that the choice of WCMS must put into place the levels of expertise that imply usability and at the same time ensure that its security is guaranteed. The usability of the WCMS implies simplicity, popularity, and support. Tretten and Karim (2014) provide that open-source WCMS must be simple to use to accommodate differences in training and expertise.

Suggesting on the general solution, Infrastructure (2016) advised website managers to endeavor to follow patching instructions from their software providers. Further, he advises on security practices and guidance which in agreement with our findings of this above.

Further, popular WCMS is advisable since users can exchange ideas on how to use them in case of complications. In this study, the respondents expressed usability of the interface, provisions for advanced personalization, the existence of developing communities and support and consultancy as important aspects of

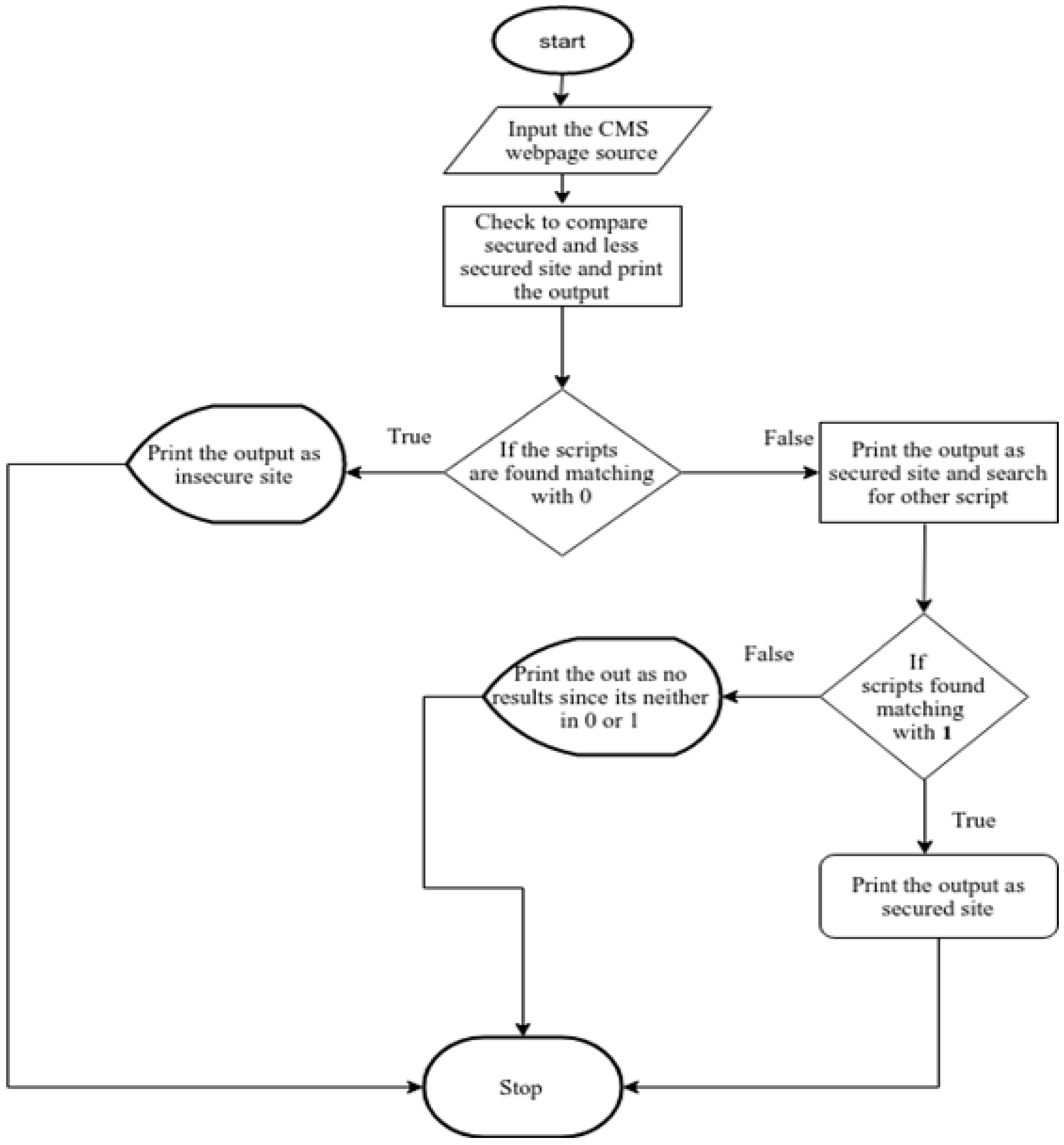
start
Input the CMS webpage source
Check to compare secured and less secured site and print the output
If the scripts are found matching with 0
True
Print the output as insecure site
False
Print the output as secured site and search for other script
If scripts found matching with 1
False
Print the out as no results since its neither in 0 or 1
True
Print the output as secured site
Stop

Figure 5. Flowchart explaining the model

WCMS. In generating code for WCMS E.G<!--security of WCMS-->WHILE considering the security awareness of the system then DO: IF users violating SQL Injection OR XSS Attack AND access to confidential information then they have unauthorized access into the WCMSset sending junk mails to users. The system should be shielded from SQL injections and parameter manipulation as shown in the pseudocode in Figure 6.

The pseudo-code model drawn follows the mentioned steps to provide the security controls of WCMS. Step one identifies all the WCMS used by website developers: Drupal, Joomla, WordPress because of all types of WCMS different levels of vulnerabilities. Step two, checks on the usability and functionality of each WCMS identified by technical evaluation score and the weighted by users. The last step leads to the identification of the highest threats performed attack by WCMS that is, SQL injection, XSS attacker and unauthorized access of information. This is because each of the mentioned attacks, attackers aim aims to execute malicious scripts in a web browser of the victim by including malicious code in a legitimate web page or web application. After this identification then the model issues the security awareness which leads security control solution criteria to exit. Finally, it provides also steps to be followed to choose a more secure WCMS by not only considering usability. Then lastly;

```
Pseudo-code presentation model
<!--Since most website designers uses Drupal, we assume that the number of
drupal users are greater than other CMS-->
<!--We declare and initialize all variables to 0 as shown-->
average, drupal, wordpress, joomla=0;
Userbility and function, security;
initialize userbility and function to zero;
initialize security to zero;
<!--check for the userbility of the CMS-->
While considering the userbility and functionality Of CMS
   then print Wordpress and Joomla are recommended
EndWhile
IF security Of CMS is considered
   then Drupal is preferred.
<!--For all the average rate of Website Developers using the CMS-->
IF drupal average users is greater than the other two CMS
   then print "Drupal is most used CMS"
ELSE IF joomla is greater than drupal and wordpress
   then print "Joomla is most used CMS
ELSE IF wordpress is greater than the joomla and drupal
   then print "Wordpress is most used CMS"
ELSE Drupal is widely used
   then print "Drupal is most used CMS":
ENDIF
<!--security of CMS-->
WHILE considering the security awareness of the system then DO:
   IF users violating SQL Injection OR XSS Attack AND access to confidential
information
      then they have unauthorized access into the WCMS
      set sending junk mails to users
<!--   printing security awareness-->
   ELSE a secure CMS should
      set confidentiality
      set integrity
      set back-up provision
   ENDIF
ENDWHILE
```

Figure 6. Pseudocode for cross-site scripting.

Provide a general discussion on the proposition of the proposed pseudo-code in terms of the objective of this paper.

Conclusion

Guaranteeing usability and security in WCMS is the top priority for web administrators as this informs the use and exhaustive utilization of WCMS applications and their integration with other applications running in an organization. While most WCMS provides easy access to web services to the users, they are vulnerable to security breaches and threats. This paper identifies the security vulnerabilities in WCMS as perceived by web administrators in public universities in Kenya, and their level of awareness of these vulnerabilities and control measure, and propose a security model towards mitigating these WCMS security vulnerabilities. Towards an exhaustive understanding of these WCMS vulnerabilities, we are currently incorporating the proposed model into the WCMS web applications to evaluate its performance. To curb the security the model follows the following steps:

Step 1: Identify all the WCMS used by website developers: Drupal, Joomla, WordPress
Step 2: Check on the usability and functionality of each WCMS identified in step 1
Step 3: Check on the security of the WCMS in step 1.
Step 4: Identify the most highly threats performed by the WCMS: that is, SQL injection, XSS attacker, unauthorized access of information.
Step 5: Issue security awareness to curb step 4.

Next, we look forwards to incorporating the proposed WCMS security model in the enterprise resource planning (ERP) at the university.

CONFLICT OF INTERESTS

The authors have not declared any conflict of interests.

REFERENCES

Ali NS, Shibghatullah AS, Al Attar MH (2015). Review of the defensive approaches for structured query language injection attacks and their countermeasures. Journal of Theoretical and Applied Information Technology 76(20).

Almroth FN (2018). A security overview of Content Management Systems. Retrieved 2(24), from Detectify Blog: https://blog.detectify.com/2018/12/04/security-overview-of-content-management-systems.

Alwan ZS, Younis MF (2017). Detection and Prevention of SQL InjectionAttack:A Survey. International Journal of Computer Science and Mobile Computing 6(8):5-17.

Augustyniak RH, Aguero D, Finley AM (2005). The IP's guide to the galaxy of portal planning: part I drafting a portal vision. Online Information Review 29(6):643- 655.

Black M, Chapman D, Clark A (2018). The Enhanced Virtual Laboratory: Extending Cyber Security Awareness through a Web-based Laboratory. Information Systems Education Journal, 16:(6)4

Cassetto O (2014). Why CMS Platforms Are Common Hacking Targets (and what to do about it). Retrieved 2 (24) 2020, from Imperva: https://www.imperva.com/blog/cms-security-tips/

CUE HR Report, EC (2016). Status Of Universities (Universities Authorized to Operate in Nairobi County). Nairobi: cue.

Cyber Security Report S (2016). Nairobi County Cyber Security Report 2016. Nairobi County: Communications Authority. Www.Dealsnow.Com, 2016, https://ccs.infospace.com/ClickHandler.ashx?encp. Accessed 9 Sept 2018

David K, Nora H (2007). New web site, new opportunities: Enforcing standards compliance within a content management system, Library Hi-Tech 25 (2): 276-287.

Deshpande VM, Nair DMK, Shah D (2017). Major Web Application Threats for Data Privacy & Security–Detection, Analysis and Mitigation Strategies.

Filotrani LJ (2018). WordPress for Journalists: From Plugins to Commercialisation. Routledge.

Gupta S, Gupta BB (2017). Cross-Site Scripting (XSS) attacks and defense mechanisms: classification and state-of-the-art. International Journal of System Assurance Engineering and Management 8(1):512-530.

Handova D (2019). How to Secure Your Content Management System (CMS). Retrieved 2 24, 2020, from SECURITYBOULEVARD:https://securityboulevard.com/2019/08/how-to-secure-your-content-management-system-cms/

Infrastructure CC (2016). Content Management Systems Security and Associated Risks. Retrieved 2 24, 2020, from CISA Cyber Infrastructure: https://www.us-cert.gov/ncas/alerts/TA13-024A

Jakobson G (2014). U.S. Patent No. 8,769,017. Washington, DC: U.S. Patent and Trademark.

Johns M, Pfistner S (2017). U.S. Patent Application No. 15/140154.

Kasli TS, Kaur N (2015). Detection and Prevention of SQL Injection Attacks using Novel Method in Web Applications. International Journal of Advances in Engineering and Technology 6(4):11-15.

Martinez-Caro JM, Aledo-Hernandez AJ, Guillen-Perez A, Sanchez-Iborra R, Cano MD (2018). A Comparative Study of Web Content Management Systems. Information 9(2):27.

Mesa O, Vieira R, Viana M, Durelli VH, Cirilo E, Kalinowski M, Lucena C (2018). Understanding vulnerabilities in plugin-based web systems: an exploratory study of wordpress. In Proceedings of the 22nd International Systems and Software Product Line Conference 1:149-159.

Mugenda AG, Mugenda A (2008). Social Science Research: Theory and Principles. Nairobi: Applied.

Mugenda DM, Mugenda D (2003). Research methods: Quantitative and Qualitative methods. Revised in Nairobi 56(12):23-34.

Nardi PM (2018). Doing survey research: A guide to quantitative methods. Routledge.

Nithya V, Pandian SL, Malarvizhi C (2015). A survey on detection and prevention of cross-site scripting attack. International Journal of Security and Its Applications 9(3):139-152.

Parsons MJ (2017). A Secure Software Design Pattern in the Prevention for Reflected Cross-Site Scripting (Doctoral dissertation, Colorado Technical University).

Peltier TR (2016). Information Security Policies, Procedures, and Standards: guidelines for effective information security management. Auerbach Publications.

Piper B, Jepkemei E, Kwayumba D, Kibukho K (2015). Kenya's ICT Policy in Practice: The Effectiveness of Tablets and E-Readers in Improving Student Outcomes. In FIRE: Forum for International Research in Education 2(1):3-18. Lehigh University Library and Technology Services. 8A East Packer Avenue, Fairchild Martindale Library Room 514, Bethlehem, PA 18015.

Priyatna F, Corcho O, Sequeda J (2014). Formalization and experiences of R2RML-based SPARQL to SQL query translation using morph. In Proceedings of the 23rd international conference on World wide web pp. 479-490.

Neuendorf KA (2016). The content analysis guidebook. Sage.

Sarmah U, Bhattacharyya DK, Kalita JK (2018). A survey of detection methods for XSS attacks. Journal of Network and Computer Applications 118:113-143.

Steiner S (2014). A Hybrid Runtime Approach to Combating High-Level Semantic Attacks (Doctoral dissertation, The University of Idaho).

Svensson R (2016). Exploiting Vulnerabilities. In From Hacking to Report Writing. Apress, Berkeley, CA. pp. 89-152.

Taber KS (2018). The use of Cronbach’s alpha when developing and reporting research instruments in science education. Research in science education 48(6):1273-1296.

Tretten P, Karim R (2014). Enhancing the usability of maintenance of data management systems. Journal of Quality in Maintenance Engineering 20(3):290-303.

Uwagbole SO, Buchanan WJ, Fan L (2017). An applied pattern-driven corpus to predictive analytics in mitigating SQL injection attack. In Emerging Security Technologies (EST), 2017 Seventh International Conference on IEEE pp. 12-17.

Williams B, Damstra D, Stern H (2015). Professional WordPress: design and development. John Wiley & Sons.

Elhakeem YFGM, Barry BI (2013). Developing a security model to protect websites from cross-site scripting attacks using ZEND framework application. In 2013 International Conference on Computing, Electrical and Electronic Engineering (Icceee) (pp. 624-629). IEEE.