Exploring Guidance for Prevent Against XSS Attacks in Open CMSs

Manal I.M. Hijazi[1]
Hij_man@hotmail.com[1]
Islamic University in Gaza

Tawfiq S.M. Barhoom[2]
Tbarhoom@iugasa.edu.ps[2]
Islamic University in Gaza

Abstract: Personal information, as well as web pages security are important for everyone because attackers used to steel our sensitive information or damaged that websites. Cross Site Scripting XSS is one type of the methods that is used by attackers. Since web browser supports the execution of scripting commands embedded in the retrieved content, attacker can exploit this feature maliciously to violate the client security. Content Management Systems CMSs give web developer an easy way to have personal websites, for those people without security prior experience, and who would be under great hunting of attackers. They believe that Content Management System just a plug-in, but it is really a website.
In this paper, we concentrate on crossing site scripting attacks problem, as one of the most common attacks in the recent World Wide Web. In this research, experiments are limited to Joomla and WordPress websites. At the end, we extracted some security guidance and rules in general for all Content Management Systems designers. Some of these rules are beneficial; especially for Joomla and WordPress developers. In this work, we trained a group of amateurs to develop their websites using Joomla and WordPress through our extracted security guidance. We believe that this work was not done before.

***Keywords*:** Cross Site Scripting, Malicious code, content Management System, Joomla, WordPress, Security Guidance.

I. INTRODUCTION

The field of information security has grown and evolved significantly in recent years. Internet has become the best friend for different kinds of users, experienced or beginners, or even teenagers. The world is still looking for security, privacy and confidentiality.
The main goals of information security are Confidentiality, Integrity and Availability. Confidentiality means the information available on a system should be safe from unauthorized people; better examples would be customer credit card information, patient medical information in hospitals or personal information of employees in an organization. If that information is not secured, the company or the organization involved in that will eventually lose its reputation and business[10]. Information security means more than confidential information about a business, customers, finances or data that should denial of a competitors, or a black hat hackers. Protecting confidential information is a business requirement, and in many cases it is an ethical and legal requirement. For any person, information security has a significant effect on privacy, which is viewed very significant in different cultures. Some individuals like to break that privacy possibly for political purposes, or for fun, and entertainment. These peoples called attackers.
Cross-Site Scripting (XSS) attack is one of the most dangerous attacks that may be launched by the attackers; it's a type of injection, in which malicious scripts are injected into otherwise benign and trusted web sites [11]. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user. XSS is still a common attack. We must be able to recognize these attacks and seek to be in a secure mode, so it is necessary to help those with limited experience in reaching safety required levels when designing their own websites

A content management system (CMSs) is a computer software that allows publishing, editing and modifying content as well as maintenance from a central interface.[1][14]. Such systems of content management provide procedures to manage workflow in a collaborative environment. These procedures can be in the form of manual steps or an automated cascade. CMSs help those with limited web designing experience or what we call

تاريخ الاستلام 2014/10/19م تاريخ الاعتماد 2015/2/9م

amateurs to create their own web pages easily and without onerous costs. Drupal, WordPress, and Joomla are some examples of open CMSs.

In this paper, the work was divided into two stages. In the scanning and analyzing phase, the levels of security for 20 number of websites designed using Joomla and WordPress were analyzed; 2) dynamic tools to search for XSS vulnerabilities in that websites were used; 3) related attacks and the size of the damaged gap were analyzed and determined; 4) discovered vulnerability and defense XSS attacks were re-corrected. In the end of this stage, we were able to extract the security guidance for developing secure websites. In the Training Phase, we trained a group of amateurs to use the extracted guidance to develop secure websites using Joomla and WordPress. We analyzed their work before and after using the extracted guidance to see how it was beneficial to them, and to which level they applied those guidelines. The rest of the paper is organized in five sections. Section2 discusses related work in this area. Section 3 describes the our methodology including that two phases and results. In section 4, evaluate the results, and suggests possible future work.

II. RELATED WORK

Generally, XSS is found in input fields of forms, guest books, shout boxes, search boxes, etc. XSS allows Html/JS/VBS code to execute within the victim's browser. There are largely two distinct countermeasures for XSS prevention at the server side: input filtering and output sanitation. Input filtering describes the process of validating all incoming data. The protection approach implemented by these filters relies on removing predefined keyword, such as JavaScript or document. Output sanitation is employed, certain characters such as < , ", or ', are HTML encoded before user-supplied data is inserted into the outgoing HTML as long as all un-trusted data is "disarmed." This way, XSS can be prevented. Both of the above protections are known frequently fail. [18]

From the client side perspective, two options exist to reduce the risk of being attacked through this vulnerability. The first disabling scripting language in the web browser as well as the HTML-enabled e-mail client provide the most protection but have the side effect of disabling functionality. The second only following links from the main web site for viewing will significantly reduce a user's exposure while still maintaining functionality. Client side solution acts as a web proxy to mitigate XSS attack which is manually generated rules to mitigate XSS attempts. Client side solution effectively protects against information leakage from the users' environment. However, none of the solutions satisfies the need of the client side. There are several client side solutions.

Several existing systems have been adapted to detect XSS attack. Application level firewalls [7] and reversal proxies [4] have been adapted to try to mitigate the XSS problem. Firewalls focus on tracking sensitive information and controlling whenever data is to be sent to un-trusted domains. Reversal proxies receive all responses from the web application and check whether there are any unauthorized scripts on them. Vogt et al [17] presents a client side approach that aims to identify information leakage using tainting of input data in the browser. The presented approach stops XSS attacks on the client side by tracking the flow of sensitive information inside the web browser. If the sensitive information is about to be transferred to a third party, the user can decide if this should be permitted or not, as an additional protection side.

Selvamani et al [12] present another Client Side Solution. CSS to mitigate XSS attacks. The main contribution of the (CSS) is that it effectively reduces XSS attacks. It provides protection without relying on web application providers. CSS supports XSS mitigation mode that significantly reduces the number of connection alert prompts while, at the same time, it provides protection against XSS attacks where the attackers may target sensitive information such as cookies and sessions IDs. It acts as a web proxy to protect XSS attacks in the browser side. The author used a technique to determine if a request for recourse is a local link. It is achieved by checking the refer HTTP header and comparing the domain in the header and the domain of the requested page. All the domain values are determined by splitting and parsing URLs.

Some authors [8] have proposed the use of static analysis techniques to discover input validation of flaws in a web application; however, this approach requires access to the source code of the application. Moreover, those static analysis schemas are usually complemented by the use of dynamic analysis technique. Some authors [15] proposed using of the static analysis techniques to discover malicious input code in web pages, but this approach requires access to the source code of the application [6][10]. Moreover, those static analysis schemas are usually complemented by the use of dynamic analysis technique. Balzarotti et al [6] use this technique to confirm potential vulnerabilities detecting during the static analysis by watching the behavior of the application at run time.

On the other hand, there are a lot of XSS detecting tools used in an open source systems such as XSS-Me. Open-source software (OSS) is a computer software that is available in source code form. The source code and certain other rights normally reserved for copyright holders are provided under a software license that permits users to study, change, and improve and also to distribute the software [21]. XSS-Me one of the best open source tools was the Exploit-Me series presented by securitycompass.com [9]. Security compass created these tools to help developers easily identify XSS and SQL injection vulnerabilities. XSS-Me is a Firefox add-on that loads in the sidebar. It identifies all input fields on a page and iterates through a user provided list of XSS strings: opening new tabs and checking the results. So users will get a report about attacks got through, what did not, and what might have.

Michael Meike et al [13] test whether users can trust security of Joomla and Drupal, sense of the systems' security. They evaluate how different configuration settings might influence security issues. Second, sending various malicious input as simple requests that could lead to XSS or SQL injection. They were aided by several simple tools - including Web Scarab and Tamper Data — to perform simple security tests by manipulating parameters sent to Web servers, such as modifying data in HTTP request aiders. They inspected the source code files of both Joomla and Drupal for additional problem areas. We simply reviewed the code to see whether the developers had taken appropriate measures before they used the variables' content. They find that Joomla and Drupal provide extensive security mechanisms. There is ample opportunity for inexperienced and experienced users to open the doors form malicious code. They recommend users should carefully set configuration settings with security in mind, and non-technical users should follow the community's recommendations.

Savan K Patel et al [20] tried to analyze the performance of Joomla, Drupal and WordPress in the same condition. They tried to prove statistically by comparing their page performance criteria which CMS is to be preferred. The same pages were created in three CMSs then hosted on local as well as live server. By requesting this page from client side, different values of page performance criteria were recorded like page load time (PLT), page size (PS), number of request, number of CSS. They find by comparing all these parameters and results that for informative and intranet site Drupal is better, for intranet with multiple functionality site Joomla is better and for live site none than the others WordPress is the best.

Savan K. Patel et al [19] analyze the performance of security in Joomla, Drupal and WordPress in the same condition. They focus on hacking and its relevant information by showing the number of web attacks statistics taken in 2011. By comparison to see out of these CMSs which provide better web security, CMSs provide such a nice basic security that you cannot directly hack the site using different web hacking techniques. It seems generally that these CMSs site were hacked due to a fault. Some controversial got cookie information of some sensitive files and directories apart from that and also found some broken links in all three CMSs using testing tool.

As shown before in the related works, CMS are widely spread used in publishing information through WWW. We found that Joomla and WordPress made the majority of CMS different platforms. This work will be restricted to these two systems and more importantly how to solve XSS attacks problems in the different versions of these systems.

IIV. METHODOLOGY AND RESULTS

In this work, we devoted our study on XSS attack in open CMS. We extracted security guidance against XSS attacks from analyzing systems that had been hacked before, designed by Joomla or WordPress. We detect the efficiency of that security guidance on the different versions of both Joomla and WordPress. Then we trained a group of amateurs to use these security guidance in developing secure Joomla or WordPress websites. The structure of our approach in more details is described in Figure 1 and Figure 2.

Phase 1: Scanning and Analyzing Phase

The purpose of this phase is to analyze different attacked websites based on Joomla and WordPress done in different versions to discover and analyze vulnerability points in them. At the end of this phase we could extract valuable security guidance that helps web designers to save their websites. There are three types of that security guidance: security guidance for Joomla developers, security guidance for WordPress developers, and the third type is for general guidance for web developer. As shown in Figure1, this phase includes the following steps:

1. *Choosing ten random websites based on Joomla and others ten based on_WordPress*

With nonalignment to well-known websites that support security issues, we selected randomly 10 websites based on Joomla, and other 10 based on WordPress of different versions. Searching of those websites were done from Google engine, to find attacked websites from attacked lists and then, scanned them to insure that they are actually attacked. Of those chosen websites there were old versions, and we use them to extract security guidance because recent studies done in 2012 by well-known professional companies selected websites developed by old versions as one of the best websites in 2012. [3][5] . We tried to find those websites based on what CMS tools, and what is the version, an online dynamic scanning tools called What CMS tools. Those tools helped us to know which website was based on Joomla or WordPress, through analysis that pages. We used more than one tool to be sure of the results. Tools of what CMS that are used: What is CMS?[1] What does CMS use? [2] CMS Detector [3]

1 http://whatcmsisthis.com/

2 http://whatcms.org/

3 http://onlinewebtool.com/cmsdetector.php

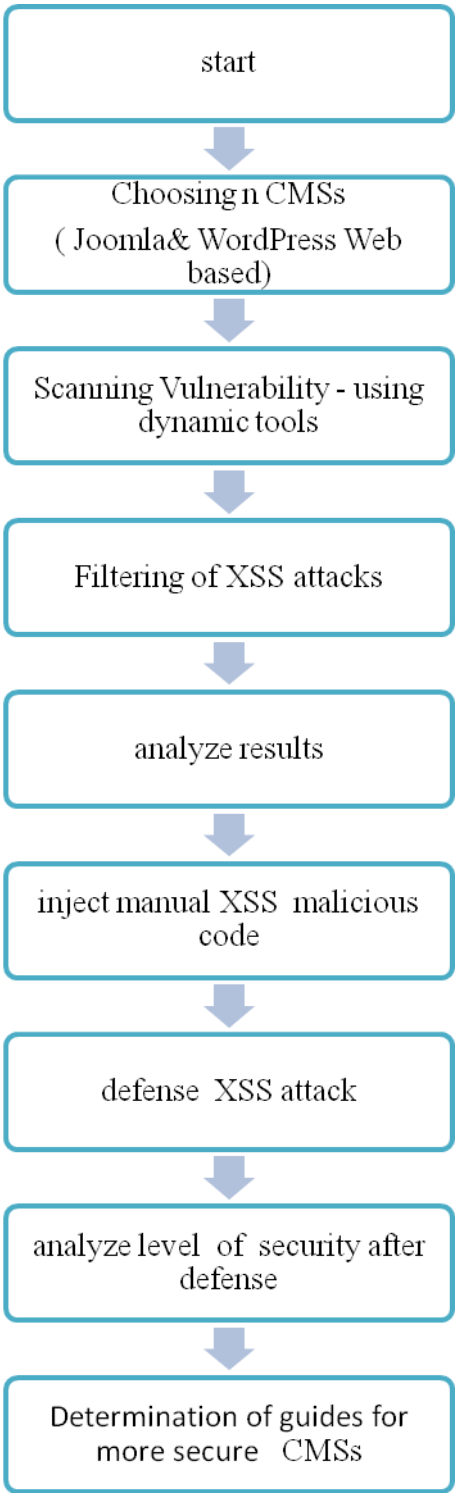

Figure1: Phase 1 of our Methodology

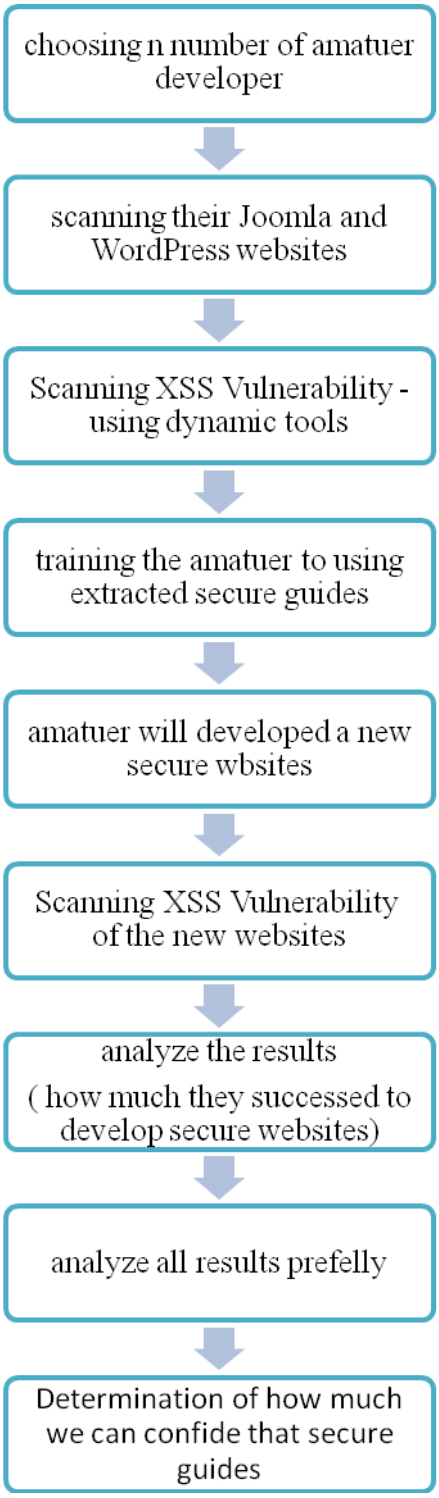

Figure 2: Phase 2 of our Methodology

2. Scanning websites using dynamic XSS scanning tools:

The results of Different scanning tools usually are not the same. In this step, we made scanning of those 20 websites using scanning tools to find out XSS attacks vulnerability using WebCruiser Web Vulnerability Scanner v 2.8.0 and Netsparker Community Edition 3.1.6.0.

3. Analyzed discovered weak points, then choosing different 10 XSS attacks of that discovered week points.

In this step, we analyzed the results of scanning tools. We took only XSS vulnerabilities in account. We searched for different 10 XSS attacks in both Joomla and WordPress scanned websites. Those different attacks covered the three types of XSS attack. So as we can control most of XSS vulnerabilities points that can face the developers. Because there are no visible attacks details results with using dynamic XSS scanning tools from the analysis of attacked websites in the last step, we could write the following different attacks and we worked with them in different versions of Joomla and WordPress. EditPlus version 2 has been chosen to review the code of designed websites to help amateurs to understand how the malicious code is executed. This program was used to open the code file contained in Joomla and WordPress package and also to inject security guidance algorithms which prevent executing the attacks in the client side.

4. Technical Details of attacks

From the previous step some attacks of those types were analyzed using scanning tools to analyze the security levels of some websites based on Joomla and WordPress, but we sow that scanning tools did not represent the details of attacks, so we had to write the attacks depending on the details results from the previous step. For example, the results of scanning http://ilovemountains.org/, website based on WordPress with WebCruiser WVS tool show that there are four get vulnerability and one post vulnerability. We cannot anticipate what XSS code vulnerability, so we tried to write ten different XSS attacks code which as much as possible cover different types of XSS vulnerability.

5. Searching for the best methods to defend against these 10 points of XSS attacks.

We studied that 10 different XSS attacks result from the previous steps in depth and analyzed related scripted code until we reached to the primary and definite cause of the problem. We found that we can protect or prevent most of XSS attacks from taking place by breaking down the attack in the server side. As mentioned before, and after continuous studies and work, we reached to breakdown each of the chosen 10 XSS vulnerability. We used some programming algorithms based on WordPress algorithms and PHP programming language and rule to defend against XSS attacks.

6. Extract security guidance to be applied against weak points that discovered to develop secure websites using Joomla or WordPress.

According to the results we obtained and how to deal with the XSS vulnerabilities, we classify these guidance into two groups, the first group according to the type of CMSs and the other group was made according to defense side which can be programming algorithms or practical rules which were used in the second phase. In brief, guidance classifications are:

- Guidance according to CMSs type:
  - General Guides
  - WordPress Guides
  - Joomla Guides

- Guidance according to defense side:
  - Programming(server side)
  - Practical rules(client side)

Results of Phase 1

Internet applications today are not static HTML pages. They are dynamic and filled with ever changing content or data. This data can contain simple text, or images, and can also contain HTML tags such as <p> for paragraph, <img> for image and <script> for scripts. XSS attacks infect the website via a form of User Input. So you should be aware of a sort of data that can land on your web page from an external source. A malicious script code could come as different ways, more details are in table 1 [2] [16]. The results of scanning websites may include one or more of the previous malicious script, but we wanted to see the code of that websites to know what malicious script was injected. Scanning tools did not show any visible attacks details so that we had to write ten different XSS with different degrees of danger and covered the three types of XSS attacks. XSS vulnerabilities only were taken into account. We worked with those attacks in the different versions of Joomla and WordPress.

Table 1: different ways of malicious script

Tag	Effective as malicious code
<SCRIPT>	the most popular way and sometimes easiest to detect
<BODY>	can contain an embedded script by using the ONLOAD event or BACKGROUND attribute
<IMG>	Some browsers will execute a script when found in the <IMG> tag as shown here: <IMG SRC="javascript:alert('XSS');">
<IFRAME>	allows to import HTML into a page , which may contain a script
<INPUT>	If the TYPE attribute of the <INPUT> tag is set to “IMAGE”, it can be manipulated to embed a script
<LINK>	often used to link to external style sheets, and could contain a script:
<TABLE>, <TD>	BACKGROUND attribute of the TABLE tag can be exploited to refer to a script
<DIV>	similar to the <TABLE> and <TD>
<OBJECT>	can be used to pull in a script from an external site
<EMBED>	Help to inject a malicious script inside a flash file

We found that we can protect or prevent most of XSS attacks by taking some guidance and rules in mind in the server side and client side. At the server side, and by using EditPlus program, we analyze the code of different pages contained within the site. We searched for the places in which information is stored, we found that different forms in pages send data to many PHP pages. In the following details, we will describe in which place each data which may contain malicious code is stored.

A- Guidance according to CMSs type:

1- WordPress Guides

Through our research we could settle the following safety guidance rules concerned with different versions of WordPress specifically. This is the guidance which we have used in training the amateurs.

Analyzing WordPress website:

1- Post Data

Data contained in the Post will be sent to Post.php found in the path *C:\AppServ\www\wordpress\wp-admin\ Post.php*. So to prevent executing malicious code, follow the following instructions:
At the beginning of the following file "C:\AppServ\www\wordpress\wp-admin\ Post.php", insert the code in Figure 3.

```
include_once "../wp-includes/kses.php";
//wp_filter_kses( $data );
if(isset($_POST['content'])){        $content = $_POST['content'];
                                     $_POST['content'] = wp_filter_kses($content);
                                }
```

Figure 3: defense code of Post Data

2- Comment Data

Data contained in the comment will be sent to wp-comments-post.php found in the path *C:\AppServ\www \wordpress\wp-comments-post.php.* So to prevent executing malicious code, follow the following instructions:

At the beginning of the file of "wp-comments-post.php", insert the code in Figure 4.

```
include_once "wp-includes/kses.php";
if(isset($_POST['comment'])){
        $data = $_POST['comment'];
        $_POST['comment'] = wp_filter_kses($data);
}
```

Figure 4: defense code of Comment Data

3- In search

Data contained in the Post will be sent to index.php found in the path *C:\AppServ\www \WordPress\index.php.* So to prevent executing malicious code, follow the following instructions:

At the beginning of the file "index.php", insert the code in Figure 5.

```
$_GET[s] = htmlspecialchars($_GET[s]);
```

Figure 5: defense code of search Data

4- New user Data

Data contained in the New user form will be sent to user-new.php found in the path In *C:\AppServ\www\wordpress\wp-admin\user-new.php.* So to prevent executing malicious code, follow the following instructions:

At the beginning of the file "user-new.php", insert the code in Figure 6.

```
include_once "../wp-includes/kses.php";
if(isset($_POST['user_login'])){
        $data = $_POST['user_login'];
        $_POST['user_login'] = wp_filter_kses($data);
        $data = $_POST['email'];
        $_POST['email'] = wp_filter_kses($data);

        $data = $_POST['first_name'];
        $_POST['first_name'] = wp_filter_kses($data);

        $data = $_POST['last_name'];
        $_POST['last_name'] = wp_filter_kses($data);

        $data = $_POST['url'];
        $_POST['url'] = wp_filter_kses($data);

        $data = $_POST['pass1'];
        $_POST['pass1'] = wp_filter_kses($data);

        $data = $_POST['pass2'];
        $_POST['pass2'] = wp_filter_kses($data);

        $data = $_POST['role'];
        $_POST['role'] = wp_filter_kses($data);

        $data = $_POST['createuser'];
        $_POST['createuser'] = wp_filter_kses($data);          }
```

Figure 6: defense code of New user Data

5- Saving against XSS from loading through Plugin, Themes, Template or media

- Install protection plugins to protect the site from their company original site.
- Install an authentication plugin from the original site company of CMS

2- Joomla Guides

As we did with WordPress, we could find the following safety guidance rules that deal with different versions of Joomla:

Analyzing Joomla website:

1- Post Data called here (Article)

Data contained in the Post will be sent to Post.php found in the path *C:\AppServ\www\Joomla15\indix.php*. So to prevent executing malicious code, follow the following instructions:

At the beginning of the file "indix.php", insert the code in Figure 7. Note that the following code is especially for escaping the title of the post, and we must do the same thing with all data sent within the article form with replacement of title word with elements names.

```
$_POST[title] = htmlspecialchars($_POST[title];
```

Figure 7: defense code of post (Article) data

2- Users Data

Data contained in the Post will be sent to Post.php found in the path *C:\AppServ\www\Joomla15\indix.php*, So to prevent executing the following instructions:

Note that the following code in figure 8, is especially for escaping the name of user, and we must do the same thing with all data sent within the User form with replacement of name word with elements names.

```
$_POST[name] = htmlspecialchars($_POST[name];
```

Figure 8: defense code of Users data

We note that different forms send data to index.php page, so that escaping will be done in this page.

3- Comment Data

You can see that there is no previous designed form to insert comments, so that you must program this form to get comment features in your Joomla websites. This does not decrease the importance of Joomla.

4- Saving against XSS from loading through Plugin, Themes, Template or media

- Install protection plugins to protect the site from their company original site.
- Install an authentication plugin from the original site company of CMS
-

3- General Guides

These guides are useful for all CMS which are related to PHP programming language. This guidance is presented in section of Programming rules.

B- Guidance according to Defense side

1- Programming rules: server side

1- Web coding

Code your web applications carefully and use the proper escaping mechanisms in the right places. The most important thing is to code the web application with safe rules at the beginning steps of development process. This will help to avoid damage of websites in early stages.

2- Filtering mechanisms; Filtering for XSS

The simplest and arguably the easiest form of XSS protection would be to pass all external data through a filter which will remove dangerous keywords, such as the infamous <SCRIPT> tag, JavaScript commands, CSS styles and other dangerous HTML markup (such as those that contain event handlers.). Usually server-side code is written in PHP, ASP, or some other web-enabled development languages by searching for keywords and then replacing them with empty strings which we could call filters. Filtering function is a good way to extract

unwanted characters so that malicious code will be filtered before reaching to the server side. Different types of filtering algorithms in more details are in table 2.

Table 2 : Filtering algorithms

Filer type	Filter algorithm	Details
PHP Filter List	FILTER_VALIDATE_BOOLEAN	Return TRUE for "1", "true", "on" and "yes", FALSE for "0", "false", "off", "no", and "", NULL otherwise
	FILTER_VALIDATE_EMAIL	Validate value as e-mail
	FILTER_VALIDATE_FLOAT	Validate value as float
	FILTER_VALIDATE_INT	Validate value as integer
	FILTER_VALIDATE_IP	Validate value as IP address, optionally only IPv4 or IPv6 or not from private or reserved ranges
	FILTER_VALIDATE_URL	Validate value as URL
Sanitize Filters:	FILTER_SANITIZE_EMAIL	Remove all characters, except letters, digits and !#$%&'*+-/=?^_`{|}~@.[]
	FILTER_SANITIZE_ENCODED	URL-encode string, optionally strip or encode special characters
	FILTER_SANITIZE_NUMBER_FLOAT	Remove all characters, except digits, +- and optionally .,eE
	FILTER_SANITIZE_NUMBER_INT	Remove all characters, except digits and +-
	FILTER_SANITIZE_SPECIAL_CHARS	HTML-escape '"<>& and characters with ASCII value less than 32
	FILTER_SANITIZE_STRING	Strip tags, optionally strip or encode special characters
	FILTER_SANITIZE_URL	Remove all characters, except letters, digits and $-_.+!*'(),{}|\\^~[]`<>#%";/?:@&=
PHP 5 Filter Functions	FILTER_INPUT_ARRAY()	Get multiple inputs from outside the script and filters them, as they come in from user side.
	FILTER_VAR_ARRAY()	Get multiple variables and filter them
	FILTER_INPUT()	Get input from outside the script and filter it
	FILTER_ID()	Returns the ID number of a specified filter
	FILTER_LIST()	Returns an array of all supported filters
	FILTER_VAR()	Get a variable and filter it

Using filter_input of id to prevent XSS attacks (PHP approach) is benefitial with WordPress and Joomla. Filtering input helps us to filter non-useful additional character so that we receive true data. For example, safe HTML data will result in using filter algorithm called filter_input as illustrated in Figure 9, true URL address formatting by the same algorithm detailed in Figure 10. In the same manner we, can use it with numeric data, valid Email, etc.

```
<?php
        $search_html = filter_input(INPUT_GET, 'search', FILTER_SANITIZE_SPECIAL_CHARS);
?>
```

Figure 9: defense code – filtering HTML data

```
<?php
        $search_url = filter_input(INPUT_GET, 'search', FILTER_SANITIZE_ENCODED);
?>
```

Figure 10: defense code – filtering URL data

3- Character Escaping from XSS code

This is the primary means to disable an XSS attack. When performing Escaping effectively the browser manipulates the received malicious code as a data. If an attacker manages to put a script on your page, the victim will not be affected because the browser will not execute. Escaping JavaScript, Cascading Style Sheets, and sometimes XML data help in protecting the website from XSS attacks. Escaping everything will make your own scripts and HTML markup not working, so we must know when to escape.

- Use HTML Escaping when Un-trusted received data is inserted in between HTML opening and closing tags.
- Use JavaScript Escaping when Un-trusted received data is inserted inside certain attributes such as STYLE and all event handlers such as ONMOUSEOVER and ONLOAD.
- Use CSS Escaping when Un-trusted data is inserted inside CSS styles.

How to protect WordPress and Joomla by using escaping (PHP approach)

Escaping character from data received from URL address with Get method as Figure 11.

```
$_GET['name'] = htmlspecialchars($_GET['name'], ENT_QUOTES, 'UTF-8');
```

Figure 11: defense code – Escaping character received from URL address (Get method)

Escaping character from data received from the page (forms) with Post method as Figure 12.

```
$_POST['content'] = htmlspecialchars($_POST['content'], ENT_QUOTES, 'UTF-8');
```

Figure 12: defense code – Escaping character received from the page (Post method)

Content refers to the data which we want to escape XSS character from. Replace content with name of data received from the form within the page.

How to protect WordPress websites by escaping algorithms

WordPress provides some additional algorithms to prevent attacks by escaping characters. Details of how to use WordPress algorithms are shown below. Note that Content refers to the element that received the value. Not that Content refers to the data which we want to escape, so you have to replace content with the name of data received from the form within the page. These algorithms are:

- wp_filter_kses() – allows you to use safe HTML, html sanitization library
- esc_html() – Be mindful when using this one, it kills html, so if you require html like features
- esc_url() – This is used to print url's to the page.
- esc_textarea() – This is designed specifically for use with textareas, it double encodes entities //making it more effective for textarea's.
- esc_attr() – As implied by the name, this is designed to escape attributes.
- esc_url_raw() – This is used to save url's to the database or to redirect.
- esc_js() – Supersedes js_escape() and used when you are printing JS to the page.

Example for applying these algorithms shown in figure 13

```
require_once('./admin.php');
if(isset($_POST['content'])){
        $content = $_POST['content'];
        $_POST['content'] = wp_filter_kses($content);        }
```

Figure 13 : defense code - wp_filter_kses()

A- Practical (client side)

There is no CMS on earth which is 100% secure and attacker proof. However, the following simple steps will increase the level of web security.

1- Avoid default 'admin' username with using something unique and safe, with a strong password containing of minimum 8 letters with special characters, numbers and alphabets.
2- Use the latest version of CMS from their company original site.
3- Avoid using vulnerable third party extensions which may include some type of attack.
4- Delete unused templates and unwanted files/folders from root directory.
5- Use correct hosting settings.

   - Safe_mode should be ON,
   - Use PHP5 rather than PHP4.

6- Write-protect configuration file.

The following changes in PHP.INI file will help to increase the security level of website:

   - disable_functions ="show_source, system, shell_exec, passthru, exec, popen, proc_open, allow_url_fopen"
   - file_uploads = Off, If you don't want file upload

7- Change the default database prefix for tables exactly for Joomla from jos_ to some other string to make guessing impossible.
8- Scan local machine through which changes are made.
9- Edit the configuration files to delete the version number of installed package
10- Install an authentication plugin from the original site of company of CMS
11- Add a suffix to your admin URL. This can prevent injection of attacks in URL address and prevent page redirect to a 404 (not found) page.

12- Install protection plugins to protect the site from their company original site.
13- Change configuration of browsers to prevent the execution of script languages.

Use an automated XSS scanning at regular times to make sure that the website is still secure, especially when updates are made against new vulnerabilities.

Phase 2 : Training Process

For operation with Joomla or WordPress it is necessary to have WEB–server with support PHP, MySQL (Apache) and WEB a browser for the user (Internet Explorer, Mozilla Firefox, Opera). Implementing the major server and database requirement before installing and setting up the CMS Joomla or WordPress. After we had extracted security guidance to be applied against weak points that are discovered to develop secure websites using Joomla or WordPress, we started the second phase that includes the following steps

1. ***Choosing a group of web amateur developers who can continue with us to the end of the research.***

After some efforts and discussion with some students to select a group of 15 of them on how they can understand my work and go further with me until we finish the research. These people now are called amateurs and like developing websites. I had taught this group of amateurs how to develop websites using Joomla version 2.5, and 1.5, and WordPress of version 3.9.1. The training period took about four weeks. We trained them to install some requirements that are needed to have CMSs such as WEB–server with support PHP and MySQL (Apache), and that took about one week. To teach them Joomla took about 18 hours. Training them on WordPress needed about 6 hours. After they had good experience with Joomla and WordPress we asked them to develop their own websites; one based on Joomla and the other based on WordPress.

2. ***The same automated scanning tools in phase 1 (***WebCruiser Web Vulnerability Scanner v. 2.8.0 and Netsparker Community Edition 3.1.6.0) ***were applied on their developed websites to chick up for XSS vulnerabilities in their websites.***
3.
4. ***To understand XSS attacks results from scanning tools from phase one, we had to teach them principles of some programming languageslike*** Jcreator, HTML and PHP. This group of amateurs did not have the same level of understanding and knowledge, so more efforts were needed to make them reach nearly an acceptable level of knowledge. This took about 8 hours.

5. ***After that, we had begun to teach them how to inject XSS attacks malicious code step by step and to show them how the results of injected websites would be***.

We represented each attack of the ten different XSS attacks, analyzed their malicious codes, and how to execute them by inserting as a comment. We measured their different opinions and responses to 10 different XSS attacks.

We asked them to inject those 10 attacks by themselves. We gave them about seven days period without any information about the security or defense to let them feel the value and cost of losing a website. Some of them lost their websites with one XSS attack and could not be retrieved.

6. ***We started to teach them to use the extracted security guidance from phase 1.***

They started to defend and protect their websites against XSS attacks. They enjoyed this step of work, and they were happy with that success. We trained them how to defend each attack in Joomla, WordPress, and PHP language. We also gave them general rules and information to protect their websites. This step took about 16 hour.

7. ***In another step, we divided them in to two groups:***

One of them acts as an attacker and the other group will defend against their attacks. We aimed to analyze their responses to the efficiency of the rules and guidance, and how they could deal with the problems. This effort took about 4 hours.

8. ***After the training period had completed successfully, we compared the results*** of scanned websites before and after training process manually by injecting different malicious XSS codes.

9. ***Testing Security Guidance and rules by true attackers.***

At the end of our work, we asked two true attackers to test our extracted guidance by trying to attack our designed websites that include the security rules.

Results of Phase 2

At the end of the work, and after training amateurs to use the extracted programming and practical security guidance, the results of this phase gave confidence; that any person who likes web developing with some training can develop a secure CMS web site. There were some defaults because of no previous knowledge about network architecture needed to install CMSs Programs. They developed other websites using WordPress, and there is no need to deal with different versions because they are similar. The most difficult part of training was helping them to understand web programming code because of little related prior experience. After they had understood the security guidance, they showed more interaction with attacks analysis. In the following tables, our impression is presented with details about training process which took about 7–8 weeks (about 60 hours). The training process completed with 11 amateurs after losing four of them. They had trained on using Joomla and WordPress to develop more secure websites. training group of amateurs includes different types of scientific spatiality with different levels of knowledge. No one of them had used Joomla or WordPress before, so this work is a good chance to understanding those two CMS platforms, with supporting security characteristics.

At the end of research we had compared the results of amateurs' activities, knowledge, experience of security and application of security guidance and roles before and after the training period ended. Comparison details shown in table 3. With deep studying and analyzing data of table 3, it is clear that the results of our program were encouraging:

- We found that the knowledge necessary to web programming language knowledge increased from 31.8 up to 70 %.
- Web development using Joomla and jumped from nothing to 66 %, and increased to 85.5 % at the end of work. The same thing happened with WordPress which reached 77 % increased up to 90 %.
- Understanding attack codes analysis became easier to be understood (reached 85 %).
- Understanding and application of security guidance and rules reached the same percentage 95 %, which is an acceptable degree of success.

We found that the results of scanning websites developed at the end of training program were excellent in respect of security issues. That was the same as the comment we received from the true two hackers

Table 3: Amateurs data results

Activity	Results degree		Notes
	Before training	After training	
Number of amateurs	15	11	4 were missed
Acceptance degree of the idea	100%		All enjoyed the idea
Understanding level of web programming (HTML, Jscript, PHP)	31.8 %	Up to 70%	Understanding level increased when they applied the practical work
Understanding web development using Joomla	0 %	66% up to 85.5 %	They took long time about 18 hour
Understanding web development using WordPress	0 %	77 % up to 90 %	Better than work with Joomla, took about 6 hour
Understanding attack code analysis	0 %	70 % up to 85 %	We assisted them using EditPlus v. 2 program
Injection of malicious code	----	85 %	They showed more ability to inject attacks in different forms
Believe in the application of security guidance	10%	Up to 95%	All of them believe of the importance of using security rules to save their works
Understanding of security guidance and rules	--	90 %	At the end of work they understood the guidance very well , because of repeated use
Degree of application of security guidance and rules	---	95 %	Some of them were smart in the work
Acceptance of security guidance	---	95%	All of the amateurs accept and thrust the importance of these guidance
Applications and success of security guidance at the end of work	---	95 %	After hard work , amateurs applied all of these guidance
Security testing of their websites by scanning tools	---	Free XSS attacks	Most of the developed websites were more secure after training
Amateurs training interaction	---	85 %	Dealing as a friends

Testing Security Guidance by true attackers

To secure a website or a web application, we had to understand the target application, how it works and the scope behind it. Ideally, the penetration tester should have some basic knowledge of programming and scripting languages, and also web security. As mentioned before in this chapter, we asked two true attackers to test our extracted security guidance and rules. The results we received informed us, after one week of testing, that there is no complete perfect work. Attackers spend every effort to inject their malicious code, but our work increased the level of security of websites. Attackers have to search deeply to find any gap. Those hackers told us that their scanning depends on trying to inject different malicious codes in different forms through our websites.

To insure success of the final results of extracting security guidance and training processes, we used the same scanning tools to scan the XSS vulnerability with secured developed websites, and the results were perfect. The results show that there were no XSS Vulnerabilities in those websites

V. CONCLUTION

Safety of websites is important and is needed for everybody. Too much effort was done and still being done to extract rules and guidance to save information and to achieve confidentiality. In conclusion, Dealing with WordPress was easier than Joomla, less effort needed to understand how to develop complete website based on WordPress. However, we found that different versions of Joomla support more security levels rather than WordPress. The capability of using Joomla to develop their own websites jumped from nothing to 85.5% at the end of work. The same thing happened with WordPress which reached up to 90%. We found that amateurs understood the importance of our extracted security guidance and were able perfectly to apply that guidance in their developed websites. We could say that all of them became more believers in the importance of using security issues to stop any unexpected security gap in their websites.

They were able to practically use the safe rules to secure their web pages which are developed based on Joomla and WordPress to high degree up to 90 %, which is an acceptable degree of success. They enjoyed this kind of work, what we could call an Ethical Hacker. Scanned websites developed at the end of training program by amateurs were excellent, where the security levels reached 95 %. The results obtained by scanning tools were XSS free, but we cannot say that the percentage is 100% because there is no complete security work. That was similar to the results we received in the comment from the true hackers, whom we asked to examine our developed websites.

We advise that the study should be done with more amateurs to collect more specific results with other CMSs like Drupal which competes with Joomla and WordPress nowadays. There are many attacks other than XSS attack, like Brut force, denial of service, content spoofing, and DNS Hijacking, etc. It is also suggested that new studies should be directed to defend against these attacks. It is a good way to build forum websites to publish this extracted guidance to help amateurs everywhere to secure their own websites, especially if they used Joomla or WordPress designing Platforms. There is no an Arabic language specialty in CMS that is found in this days. We hope that developers can build an Arabic CMS, with security features that ones can confide with to develop Arabic websites.

REFERENCE

[1] A. Rockley, P. Kostur, and S. Manning, *Managing enterprise content: A unified content strategy*: New Riders, 2003.

[2] Acunetix, https://www.acunetix.com/websitesecurity/cross-site-scripting, 13/July, 2014

[3] B. Aziz, A. Arenas, G. Cortese, B. Crispo, and S. Causetti, "A Secure and Scalable Grid-Based Content Management System," in *Availability, Reliability, and Security, 2010. ARES '10 International Conference on*, 2010, pp. 404-409.

[4] C. Ding, "Cross-Site Request Forgery Attack and Defence: Literature Search." V1. 0, vol, visited in 2014.

[5] Creativebloq, http://www.creativebloq.com/web-design/examples-wordpress-11121165, 20/May, 2014

[6] D. Balzarotti, M. Cova, V. Felmetsger, N. Jovanovic, E. Kirda, C. Kruegel, *et al.*, "Saner: Composing static and dynamic analysis to validate sanitization in web applications," in *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, 2008, pp. 387-401.

[7] E. Kirda, C. Kruegel, G. Vigna, and N. Jovanovic, "Noxes: a client-side solution for mitigating cross-site scripting attacks," in *Proceedings of the 2006 ACM symposium on Applied computing*, 2006, pp. 330-337.

[8] G. Wassermann and S. Zhendong, "Static detection of cross-site scripting vulnerabilities," in *Software Engineering, 2008. ICSE '08. ACM/IEEE 30th International Conference on*, 2008, pp. 171-180.

[9] http://Labs.securitycompass.com/index.php/explot-me/, February, 2014.

[10] J. John," *Information security*", processed by ACM Puplication, January 2006.

[11] J. Grossman, *XSS Attacks: Cross-site scripting exploits and defense*: Syngress, processed by Elsevier Limited, Oxford, , PP 448, 2007.

[12] K. Selvamani, A. Duraisamy, and A. Kannan, "Protection of Web Applications from Cross-Site Scripting Attacks in Browser Side," (IJCSIS) International Journal of computer Science and information Security, Vol.7, No, 3, *arXiv preprint arXiv:1004.1769*, 2010.

[13] M. Meike, J. Sametinger, and A. Wiesauer, "Security in Open Source Web Content Management Systems," *Security & Privacy, IEEE*, vol. 7, pp. 44-51, 2009.

[14] M. White, *The Content management handbook*: Library Assn Pub Limited, 2005.

[15] N. Jovanovic, C. Kruegel, and E. Kirda, "Pixy: A static analysis tool for detecting web application vulnerabilities," in *Security and Privacy, 2006 IEEE Symposium on*, 2006, pp. 6 pp.-263.

[16] OWASP, https://www.owasp.org/index.php/Cross-site_Scripting_(XSS), 2/May, 2014.

[17] P. Vogt, F. Nentwich, N. Jovanovic, E. Kirda, C. Kruegel, and G. Vigna, "Cross Site Scripting Prevention with Dynamic Data Tainting and Static Analysis," in *NDSS*, 2007.

[18] R. A. Martin, "Integrating your information security vulnerability management capabilities through industry standards (CVE&OVAL)," in *Systems, Man and Cybernetics, 2003. IEEE*

International Conference on, 2003, pp. 1528-1533 vol.2.

[19] S. K. Patel, V. R. Rathod, and J. B. Prajapati, "Comparative analysis of web security in open source content management system," in *Intelligent Systems and Signal Processing (ISSP), 2013 International Conference on*, 2013, pp. 344-349.

[20] S. K. Patel, V. R. Rathod, and S. Parikh, "Joomla, Drupal and WordPress - a statistical comparison of open source CMS," in *Trendz in Information Sciences and Computing (TISC), 2011 3rd International Conference on*, 2011, pp. 182-187.

[21] W. T. Verts, "Open source software," *World Book Online Reference Center,* 2008. Available on http://www. WordBookonline.com/wb/article?id=ar751706, February, 2014.