INTERNATIONAL JOURNAL OF PURE AND APPLIED RESEARCH IN ENGINEERING AND TECHNOLOGY

A PATH FOR HORIZING YOUR INNOVATIVE WORK

# WEB SECURED VULNERABILITIES OF WEAK PROGRAMMING LANGUAGE

**MAYURA SHARAD PATIL, NILIMA DONGRE**

1. ME Student, RAIT, Nerul, Navi Mumbai, India.
2. Assistant Professor, RAIT, Nerul , Navi Mumbai, India

**Accepted Date: 05/03/2015; Published Date: 01/05/2015**

**Abstract:** Dynamic web page technology has become part of the development of the Internet. Most web applications have critical bugs (faults) affecting their security, which makes them vulnerable to attacks by hackers and organized crime. To prevent these security problems from occurring it is of utmost importance to understand the typical software faults. The most widely spread and critical web application vulnerabilities are SQL Injection and XSS. The source code of security patches of widely used web applications written in weak and strong typed languages. Many programming languages are currently used to develop web applications. Ranging from proprietary languages (e.g., C#, VB) to open source languages (e.g., PHP, CGI, Perl, Java), the spectrum of languages available for web development is immense.

**Keywords:** Web Secure, PHP, CGI, Java

**Corresponding Author: MS. MAYURA SHARAD PATIL**

**Access Online On:**

www.ijpret.com

**How to Cite This Article:**

Mayura Sharad Patil, IJPRET, 2015; Volume 3 (9): 674-680

***PAPER-QR CODE***

## INTRODUCTION

This Special Topic contributes to fill the gap of source code defects generating major web application. The goal is to understand the typical software faults that are behind the majority of web application vulnerabilities, taking into account different programming languages. To understand the relevance of these kinds of vulnerabilities for the attackers.

Regarding the programming language perspective, we considered some of the most relevant in the context of web applications. First, we focused on the most widely used weak typed language, PHP. Then, we analyzed strong typed languages, namely Java, C#, and VB.As shown in figure [1] for web application architecture.

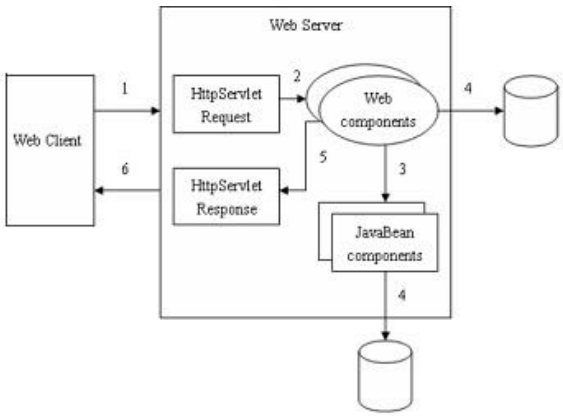


**Fig 1: Web Application Architecture**

1.1 Vulnerability and Programming Language:

Many programming languages are currently used to develop web applications. Ranging from proprietary languages (e.g., C#, VB) to open source languages (e.g., PHP, CGI, Perl, Java), the spectrum of languages available for web development is immense. Programming languages can be classified using taxonomies, such as the programming paradigm, the type system, the execution mode, and so on.

Related Works

One of the aspects that contribute to security problems seems to be related to how bad different programming languages are in terms of propensity for mistakes. Jose´ Fonseca [1] common security problems related to the easiness in programming with PHP and its features, but this affects many other programming languages. One of the best practices to find software faults is to perform a static analysis to the code [2]. This is a labor intensive job, usually done

with automated tools, although they lack the precision of the manual counterpart. To improve them and to help predict software failures, a new defect classification scheme was proposed [2].

To improve software quality, developers need a deeper knowledge about the software faults that must be mitigated. The underlying idea is that knowing the root cause of software defects helps removing their source, therefore contributing to the quality improvement [3]. Researchers at IBM developed a classification scheme of software faults, intended to improve the software design process and, consequently, reduce the number of faults [4], [5].

Orthogonal Defect Classification - A technology for software process measurement and analysis, providing a standard of unique, non-redundant attribute definitions for defect classification as well as analysis metrics and procedures for process evaluations.

Web application attacks

Web application-level attacks refer to the vulnerabilities inherent in the code of a Web

Application itself, regardless of the technology in which it is implemented or the security of the Web server/back end database on which it is built. Attacks against Web-based mail are also included in this category. The origin of these vulnerabilities may be errors in HTML forms, client-side scripts, server-side scripts (.asp, .jsp, .php, .pl, etc.), business logic objects (COM, COM+, CORBA, etc.), SQL sentences processing, etc.

3.1 Vulnerability

In order to reach the desired result, an attacker must take advantage of a computer or network vulnerability. A vulnerability is a weakness in a system allowing unauthorized action. We define the following vulnerabilities in Web applications.

Code injection

Code injection vulnerabilities allow for injecting arbitrary user-chosen code into a page. These vulnerabilities arise from none existent or poorly designed input validation routines on the server-side. The main categories of code injection are:

• Script injection:

The attack involves Web servers that dynamically generate HTML pages. If these servers embed browser input in the dynamic pages that they send back to the browser, these servers can be

manipulated to include content in the dynamic pages that will allow malicious scripts to be executed.

- SQL injection:

An attacker creates or alters existing SQL commands to gain access to unintended data or even the ability to execute system level commands on the host. See for example.

• XPath injection: XPath is a language for addressing parts of an XML document. An attacker can modify search strings to access unauthorized data in XML documents.

Canonicalization

Canonicalization vulnerabilities occurs when an application makes a security decision based on a name (a filename, a folder name, a Web address), without having in mind the fact that the name may be expressed in more than one way. The most common way of exploiting canonicalization issues is in the form of path traversal attacks, which allow a malicious user to execute commands or view data outside of the intended target path.

OpenWeb Application Security and Detection Methods

According to OWASD (Open Web Application Security and Detection)[8], the most efficient way of finding security vulnerabilities in web applications is manual code review. This technique is very time-consuming, requires expert skills, and is prone to overlooked errors. Therefore, security society actively develops automated approaches to finding security vulnerabilities. These approaches can be divided into two wide categories:

Black-box testing

White-box testing

The second approach is based on web application analysis from the server side, with assumption that source code of the application is available. In this case dynamic or static analysis techniques can be applied. A comprehensive survey of these techniques was made by Vigna et al. According to this survey several statements could be made:

• The most common model of input validation vulnerabilities is the Tainted Mode model [3]. This model was implemented both by means of static [7, 9, 15] or dynamic analysis [3-6].

• Another approach to model input validation vulnerabilities is to model syntactic structure for sensitive operations arguments. The idea behind this is that the web application is susceptible

to an injection attack, if syntactic structure for sensitive operation arguments depends on the user input. This approach was implemented by means of string analysis in static [16, 17] and it was applied to detect SQLI [19] and XSS [18] vulnerabilities in PHP. After all, this approach was implemented to detect injection attacks at runtime.

• One of the main drawbacks of static analysis in general is its susceptibility to false positives caused by inevitable analysis imprecisions. This is made worse by dynamic nature of scripting languages. However, static analysis techniques normally perform conservative analysis that considers every possible control path.

• One of the contrary, one of the main drawbacks of dynamic analysis is that it is performed on executed paths and does not give any guarantee about paths not covered during a given execution. However, dynamic analysis having access to internals of web application execution process has the potential of being more precise.

The contributions of this paper are the following:

• We improve classical Tainted Mode model so that inter-module data flows could be checked.

• We introduce a new approach to automatic penetration testing by leveraging it with information obtained from dynamic analysis. Thus more accuracy and precision is achieved due to widened scope of web application view (database scheme and contents, intra-module data flows).Input validation routines can be tested for correctness

**Components Based Comparison of programming language practices.**

| Components | PHP | C# | JAVA |
|---|---|---|---|
| Memory management | Default using GC(by PHP 5.3) or Manual(by extension API) | Yes, provided by the CLR | Garbage collection, provided by the JVM |
| Error Detection Method-Black Box Testing | Used in Client Side Scripting Language. | Static/Manual Error Handling | Yes |
| White Box Testing | Used in Server side scripting language | Dynamic Error Handling | Yes/More Accurate. |
| Algorithms | FGK algorithm(Faller Gallager Knuth) or the Vitter algorithm | Clustering Algorithms | Clustering Algorithms |

| | | | | |
|---|---|---|---|---|
| Algorithm Technique | Huffman Coding, Data Compression | Pattern Recognition | Pattern Recognition | |

**CONCLUSION**

According to our findings, weak typed are the preferred targets for the development of exploits. We also observed that a single fault type was responsible for most of the security problems analyzed. We saw that the fault types responsible for XSS and SQLi belong to a narrow list, which points a path to the improvement of web applications, namely in the context of code inspections and analysis. This study showed that the way programmers fix vulnerabilities seems to have a degree of dependence with the type of language used. However, the number of vulnerabilities analyzed in our and other studies show that the use of a specific language is not guarantee of success in preventing vulnerabilities. This is likely to continue and lead to the growth in managed security services as companies look to bring in specialists to augment their overextended teams. Ultimately this will lead to better protection for web, cloud and mobile applications.

**REFERENCE**

1. Analysis of Field Data on Web Security Vulnerabilities, Jose Fonseca, Nuno Seixas, Marco Vieira, and Henrique Madeira Components Based Comparison of programming language practices.

2. Unforgivable Vulnerabilities, Steve Christey, The MITRE Corporation, coley@mitre.org, August 2, 2007.

3. Detecting Security Vulnerabilities in Web Applications, Using Dynamic Analysis with Penetration Testing, Andrey Petukhov, Dmitry Kozlov, Application Security conference-Belgium 2008.

4. A new taxonomy of Web attacks suitable for efficient Encoding, Computers Security Vol 22, No 5, pp 435-449, 2003.

5. Acunetix Ltd., “Is Your Website Hackable? Do a Web Security Audit with Acunetix Web Vulnerability Scanner,” http://www.acunetix.com/security-audit/index/, May 2013.

6. G. Alvarez and S. Petrovic, “A New Taxonomy of Web Attacks Suitable for Efficient Encoding,” Computers and Security, vol. 22, no. 5, pp. 435-449, July 2003.

7. P. Anbalagan and M. Vouk, “Towards a Unifying Approach in Understanding Security Problems,” Proc. Int’l Symp. Software Reliability Eng., pp. 136-145, 2009.

8. A. Avizienis, J.C. Laprie, B. Randell, and C. Landwehr, “Basic Concepts and Taxonomy of Dependable and Secure Computing,” IEEE Trans. Dependable and Secure Computing, vol. 1, no. 1, pp. 11- 33, Jan.-Mar. 2004.

9. US-CERT Vulnerability Notes Database, “Homepage,” http://www.kb.cert.org/vuls/, May 2013.

10. R. Chillarege, I.S. Bhandari, J.K. Chaar, M.J. Halliday, D. Moebus,B. Ray, and M. Wong, “Orthogonal Defect Classification—A Concept for In-Process Measurement,” IEEE Trans. Software Eng., vol. 18, no. 11, pp. 943-956, Nov. 1992.