PHP Vulnerabilities in Web Servers

Written By:
David K. Liefer
Steven K. Ziegler

Abstract:

The Internet has grown to be hugely popular and used by people of all different backgrounds and professions. Individual web pages are created by just about everyone whether or not they have any development experience or not. PHP (PHP: Hypertext Preprocessor) is one of the more popular scripting languages used by beginners and advanced users. It is an attractive alternative to Java for the novice user but little do they know there are some frightening vulnerabilities that can be exploited by clients looking to cause problems or gain access to private information or resources that cannot be tied to them. A few of these exploits include remote and local file inclusion or execution. Through these basics types of vulnerabilities a malicious client could gain complete access to a web server. To avoid these attacks a web developer needs to take care when writing PHP scripts. The most common mistake made by developers is to unknowingly expose internal variables to clients or when access is needed not properly sanitizing them to ensure the values make sense for the context in which they are being used. If un-sanitized variables are used in conjunction with certain PHP function calls private files can be accessed or remote files can be uploaded and executed. A few of the more dangerous PHP functions calls are _GET[] or passthru(). These functions need to be used with care or disabled by the system administrator to avoid problems with badly written PHP scripts. To aid in the detection of potential vulnerabilities the authors of this paper implemented a Vulnerability Detection Tool (VDT). The tool is a java based program which takes a set of user defined rules then uses these rules to parse through the contents of every file in a web development directory. If a defined rule is violated a report giving the filename, line number, and severity is displayed in the progress window of the GUI. Readers are encouraged to evaluate the tool, provide feedback to the authors, and when so motivated submit improvements to the tool.

Table of Contents:

---

1. Introduction

In the early days of the Internet most web development was done by professionals. Since then an array of web development tools, several scripting languages, and easily configurable web server software has made it easier for the novice to create and host their own website's. One of the more popular scripting languages is PHP. It has a significant user base and thousands of free scripts can be found all over the internet to perform all kinds of useful functions. It is an attractive alternative to Java for the novice user but little do they know there are some frightening vulnerabilities that can exploited by clients looking to cause problems or gain access to private information or resources that cannot be tied to them. In this paper, the basics behind the PHP scripting language and Apache web server architecture will be outlined. The latter mainly to understand how requests and data get forwarded from the web server to the underlining PHP module for interpretation and then passed back to the web server core where it is sent to the client requesting the information. Any web server could have been used for this paper, but the Apache web server was chosen dues to its popularity and availability as an open source product. Next, an investigation of the various PHP vulnerabilities will be conducted to provide readers with information that will help them write PHP scripts that are not easily exploited. To help find these vulnerabilities in website source code the writers of this paper have created a simple yet flexible tool that will recursively check selected directories for files with certain extensions to see if they have violated any of the predefined or user defined rules. When a rule is violated the severity is reported along with the filename and line number. The tool is aptly named the Vulnerability Detection Tool and its design and use is also outlined in this paper.

---

2. Background

The subject of this paper is to find PHP vulnerabilities in web servers. The web server we chose to use for this project is Apache, which is an open source product produced by the Apache Software Foundation. A little background information on PHP and the Apache Web server is probably warranted.

2.1 PHP

The roots of PHP are quite simple and originate with one man. His name was Rasmus Lerdorf and in 1995 he wrote a simple set of Perl scripts to track accesses of his online resume. He named these scripts "Personal Home Page Tools". Over time the size and number of Perl scripts got rather large and it was clear that an implementation in a standard programming language would be required to make it more scalable and easier to maintain. He chose the C programming language and made the application and source available to everyone.

He called the application "Personal Home Page / Forms Interpreter" or PHP/FI. The new implementation gave users the ability to communicate with databases and make simple dynamic web applications. Over the years, PHP/FI became quite popular and in 1997 the second version of PHP/FI was released. This version incorporated fixes and enhancements from the user community. The official release date for PHP/FI 2.0 was November 1997 but its life would be short lived because PHP/FI would receive a major overhaul and name change from some new developers.

A couple of students attempted to use PHP/FI for a university project but realized PHP/FI was not powerful enough to support the eCommerce application they developed for their project. Their names were Andi Gutmans and Zeev Suraski. Instead of abandoning PHP/FI they decided to redesign it so it fulfilled the needs for their project. The new version 3.0 was released, with the cooperation of Rasmus Lerdorf, as a successor to PHP/FI and was renamed to simply PHP. The new acronym is meant to be recursive for PHP: Hypertext Preprocessor. The new version contained many enhancements but one of its strongest was extensibility. This feature alone attracted many developers to create extension modules that added to the functionality of PHP and also its popularity. The new release also provided a solid infrastructure for different databases, protocols, and APIs. In addition, its object oriented support and much more consistent and verbose language syntax made it a powerful web development application.

Once renamed and released Andi Gutmans and Zeev Suraski took over development of PHP. The new version 3.0 added a lot of new functionality but did not do it very efficiently nor was the architecture as modular as the authors would have liked. To solve this problem a re-write of the PHP core was required.

The re-write of the PHP core was completed and ready to release as version 4 in May of 2000. This version contained what the authors dubbed as the "Zend Engine" and was named using parts of both their first names. The "Zend Engine" met all the design requirements for performance and modularity. In addition, it added many new features such as support of more Web servers, HyperText Transfer Protocol (HTTP) sessions, output buffering, and fixes for security vulnerabilities dealing with handling user input. The next major version was released in July 2004 and contains the second version of the Zend Engine, more fixes, and additional functionality. The latest released version, as of this writing is 5.2.5. Now that we know about the history of PHP lets look at the language itself to better understand its popularity. In section 3 of this paper some of the vulnerabilities of PHP will be investigated.

The main idea behind PHP is to be able to write PHP scripts embedded within Hyper Text Markup Language (HTML). An example of a simple script to echo something onto the web browser is shown in Figure 1 below.

```
<html>
    <head>
        <title>Example</title>
    </head>
    <body>
        <?php
            Echo “Hello World”;
        ?>
    </body>
</html>
```

Figure 1: Simple PHP Example

In this simple example the PHP tags allow the web server to know when to send the code to the PHP module running in the background. When the PHP start tag is encountered the web server application that is