.. \_setup:

Installation

============

phpMyAdmin does not apply any special security methods to the MySQL

database server. It is still the system administrator's job to grant

permissions on the MySQL databases properly. phpMyAdmin's :guilabel:`Users`

page can be used for this.

.. warning::

:term:`Mac` users should note that if you are on a version before

:term:`Mac OS X`, StuffIt unstuffs with :term:`Mac` formats. So you'll have

to resave as in BBEdit to Unix style ALL phpMyAdmin scripts before

uploading them to your server, as PHP seems not to like :term:`Mac`-style

end of lines character ("``\r``").

Linux distributions

+++++++++++++++++++

phpMyAdmin is included in most Linux distributions. It is recommended to use

distribution packages when possible - they usually provide integration to your

distribution and you will automatically get security updates from your distribution.

Debian

------

Debian's package repositories include a phpMyAdmin package, but be aware that

the configuration file is maintained in ``/etc/phpmyadmin`` and may differ in

some ways from the official phpMyAdmin documentation.

OpenSUSE

--------

OpenSUSE already comes with phpMyAdmin package, just install packages from

the `openSUSE Build Service <http://software.opensuse.org/package/phpMyAdmin>`\_.

Ubuntu

------

Ubuntu ships phpMyAdmin package, however if you want to use recent version, you

can use packages from

`PPA for Michal Čihař <https://launchpad.net/~nijel/+archive/phpmyadmin>`\_.

Gentoo

------

Gentoo ships the phpMyAdmin package, both in a near stock configuration as well

as in a ``webapp-config`` configuration. Use ``emerge dev-db/phpmyadmin`` to

install.

Mandriva

--------

Mandriva ships the phpMyAdmin package in their ``contrib`` branch and can be

installed via the usual Control Center.

Fedora

------

Fedora ships the phpMyAdmin package, but be aware that the configuration file

is maintained in ``/etc/phpMyAdmin/`` and may differ in some ways from the

official phpMyAdmin documentation.

Red Hat Enterprise Linux

------------------------

Red Hat Enterprise Linux itself and thus derivatives like CentOS don't

ship phpMyAdmin, but the Fedora-driven repository

`Extra Packages for Enterprise Linux (EPEL) <http://fedoraproject.org/wiki/EPEL>`\_

is doing so, if it's

`enabled <http://fedoraproject.org/wiki/EPEL/FAQ#howtouse>`\_.

But be aware that the configuration file is maintained in

``/etc/phpMyAdmin/`` and may differ in some ways from the

official phpMyAdmin documentation.

Installing on Windows

+++++++++++++++++++++

The easiest way to get phpMyAdmin on Windows is using third party products

which include phpMyAdmin together with a database and web server such as

`XAMPP <http://www.apachefriends.org/en/xampp.html>`\_.

You can find more of such options at `Wikipedia <https://en.wikipedia.org/wiki/List\_of\_AMP\_packages>`\_.

.. \_quick\_install:

Quick Install

+++++++++++++

#. Choose an appropriate distribution kit from the phpmyadmin.net

Downloads page. Some kits contain only the English messages, others

contain all languages. We'll assume you chose a kit whose name

looks like ``phpMyAdmin-x.x.x -all-languages.tar.gz``.

#. Ensure you have downloaded a genuine archive, see :ref:`verify`.

#. Untar or unzip the distribution (be sure to unzip the subdirectories):

``tar -xzvf phpMyAdmin\_x.x.x-all-languages.tar.gz`` in your

webserver's document root. If you don't have direct access to your

document root, put the files in a directory on your local machine,

and, after step 4, transfer the directory on your web server using,

for example, ftp.

#. Ensure that all the scripts have the appropriate owner (if PHP is

running in safe mode, having some scripts with an owner different from

the owner of other scripts will be a problem). See :ref:`faq4\_2` and

:ref:`faq1\_26` for suggestions.

#. Now you must configure your installation. There are two methods that

can be used. Traditionally, users have hand-edited a copy of

:file:`config.inc.php`, but now a wizard-style setup script is provided

for those who prefer a graphical installation. Creating a

:file:`config.inc.php` is still a quick way to get started and needed for

some advanced features.

Manually creating the file

--------------------------

To manually create the file, simply use your text editor to create the

file :file:`config.inc.php` (you can copy :file:`config.sample.inc.php` to get

a minimal configuration file) in the main (top-level) phpMyAdmin

directory (the one that contains :file:`index.php`). phpMyAdmin first

loads :file:`libraries/config.default.php` and then overrides those values

with anything found in :file:`config.inc.php`. If the default value is

okay for a particular setting, there is no need to include it in

:file:`config.inc.php`. You'll probably need only a few directives to get going; a

simple configuration may look like this:

.. code-block:: xml+php

<?php

$cfg['blowfish\_secret'] = 'ba17c1ec07d65003'; // use here a value of your choice

$i=0;

$i++;

$cfg['Servers'][$i]['auth\_type'] = 'cookie';

?>

Or, if you prefer to not be prompted every time you log in:

.. code-block:: xml+php

<?php

$i=0;

$i++;

$cfg['Servers'][$i]['user'] = 'root';

$cfg['Servers'][$i]['password'] = 'cbb74bc'; // use here your password

$cfg['Servers'][$i]['auth\_type'] = 'config';

?>

For a full explanation of possible configuration values, see the

:ref:`config` of this document.

.. index:: Setup script

.. \_setup\_script:

Using Setup script

------------------

Instead of manually editing :file:`config.inc.php`, you can use phpMyAdmin's

setup feature. First you must manually create a folder ``config``

in the phpMyAdmin directory. This is a security measure. On a

Debian system you can use the following commands:

.. code-block:: sh

/usr/sbin/pma-configure

.. note::

Debian and Ubuntu have simplified this setup and all you need to do is to

execute :program:`/usr/sbin/pma-configure`.

On other platforms, simply create the folder and ensure that your web

server has read and write access to it. :ref:`faq1\_26` can help with

this.

Next, open your browser and visit the location where you installed phpMyAdmin, with the ``/setup`` suffix. If you have an existing configuration,

use the ``Load`` button to bring its content inside the setup panel.

Note that \*\*changes are not saved to disk until you explicitly choose ``Save``\*\*

from the \*Configuration\* area of the screen. Normally the script saves the new

:file:`config.inc.php` to the ``config/`` directory, but if the webserver does

not have the proper permissions you may see the error "Cannot load or

save configuration." Ensure that the ``config/`` directory exists and

has the proper permissions - or use the ``Download`` link to save the

config file locally and upload it (via FTP or some similar means) to the

proper location.

Once the file has been saved,

the permissions must be reset, again as a security

measure:

.. code-block:: sh

/usr/sbin/pma-secure

.. note::

Debian and Ubuntu have simplified this setup and all you need to do is to

execute :program:`/usr/sbin/pma-secure`.

Now the file is ready to be used. You can choose to review or edit the

file with your favorite editor, if you prefer to set some advanced

options which the setup script does not provide.

#. If you are using the ``auth\_type`` "config", it is suggested that you

protect the phpMyAdmin installation directory because using config

does not require a user to enter a password to access the phpMyAdmin

installation. Use of an alternate authentication method is

recommended, for example with HTTP–AUTH in a :term:`.htaccess` file or switch to using

``auth\_type`` cookie or http. See the :ref:`faqmultiuser`

for additional information, especially :ref:`faq4\_4`.

#. Open the `main phpMyAdmin directory <index.php>`\_ in your browser.

phpMyAdmin should now display a welcome screen and your databases, or

a login dialog if using :term:`HTTP` or

cookie authentication mode.

#. You should deny access to the ``./libraries`` and ``./setup/lib``

subfolders in your webserver configuration.

Such configuration prevents from possible

path exposure and cross side scripting vulnerabilities that might

happen to be found in that code. For the Apache webserver, this is

often accomplished with a :term:`.htaccess` file in those directories.

#. It is generally a good idea to protect a public phpMyAdmin installation

against access by robots as they usually can not do anything good

there. You can do this using ``robots.txt`` file in root of your

webserver or limit access by web server configuration, see

:ref:`faq1\_42`.

.. \_verify:

Verifying phpMyAdmin releases

+++++++++++++++++++++++++++++

Since July 2015 all phpMyAdmin releases are cryptographically signed by the

releasing developer, who is currently Marc Delisle. His key id is

0x81AF644A, his PGP fingerprint is:

.. code-block:: console

436F F188 4B1A 0C3F DCBF 0D79 FEFC 65D1 81AF 644A

and you can get more identification information from `https://keybase.io/lem9 <https://keybase.io/lem9>`\_. You should verify that the signature matches

the archive you have downloaded. This way you can be sure that you are using

the same code that was released.

Each archive is accompanied with ``.asc`` files which contains the PGP signature

for it. Once you have both of them in the same folder, you can verify the signature:

.. code-block:: console

$ gpg --verify phpMyAdmin-4.4.9-all-languages.zip.asc

gpg: Signature made Fri Jun 12 13:09:58 2015 CEST using RSA key ID 81AF644A

gpg: Can't check signature: No public key

As you can see gpg complains that it does not know the public key. At this

point you should do one of the following steps:

\* Download the keyring from `our download server <https://files.phpmyadmin.net/phpmyadmin.keyring>`\_, then import it with:

.. code-block:: console

$ gpg --import phpmyadmin.keyring

\* Download and import the key from one of the key servers:

.. code-block:: console

$ gpg --keyserver hkp://pgp.mit.edu --recv-keys 81AF644A

gpg: requesting key 81AF644A from hkp server pgp.mit.edu

gpg: key 81AF644A: public key "Marc Delisle <marc@infomarc.info>" imported

gpg: no ultimately trusted keys found

gpg: Total number processed: 1

gpg: imported: 1 (RSA: 1)

This will improve the situation a bit - at this point you can verify that the

signature from the given key is correct but you still can not trust the name used

in the key:

.. code-block:: console

$ gpg --verify phpMyAdmin-4.4.9-all-languages.zip.asc

gpg: Signature made Fri Jun 12 13:09:58 2015 CEST using RSA key ID 81AF644A

gpg: Good signature from "Marc Delisle <marc@infomarc.info>" [unknown]

gpg: WARNING: This key is not certified with a trusted signature!

gpg: There is no indication that the signature belongs to the owner.

Primary key fingerprint: 436F F188 4B1A 0C3F DCBF 0D79 FEFC 65D1 81AF 644A

The problem here is that anybody could issue the key with this name. You need to

ensure that the key is actually owned by the mentioned person. The GNU Privacy

Handbook covers this topic in the chapter `Validating other keys on your public

keyring`\_. The most reliable method is to meet the developer in person and

exchange key fingerprints, however you can also rely on the web of trust. This way

you can trust the key transitively though signatures of others, who have met

the developer in person. For example you can see how `Marc's key links to

Linus's key`\_.

Once the key is trusted, the warning will not occur:

.. code-block:: console

$ gpg --verify phpMyAdmin-4.4.9-all-languages.zip.asc

gpg: Signature made Fri Jun 12 13:09:58 2015 CEST using RSA key ID 81AF644A

gpg: Good signature from "Marc Delisle <marc@infomarc.info>" [full]

Should the signature be invalid (the archive has been changed), you would get a

clear error regardless of the fact that the key is trusted or not:

.. code-block:: console

$ gpg --verify phpMyAdmin-4.4.9-all-languages.zip.asc

gpg: Signature made Fri Jun 12 13:09:58 2015 CEST using RSA key ID 81AF644A

gpg: BAD signature from "Marc Delisle <marc@infomarc.info>" [unknown]

.. \_Validating other keys on your public keyring: https://www.gnupg.org/gph/en/manual.html#AEN335

.. \_Marc's key links to Linus's key: http://pgp.cs.uu.nl/mk\_path.cgi?FROM=00411886&TO=81AF644A

.. index::

single: Configuration storage

single: phpMyAdmin configuration storage

single: pmadb

.. \_linked-tables:

phpMyAdmin configuration storage

++++++++++++++++++++++++++++++++

For a whole set of additional features (bookmarks, comments, :term:`SQL`-history,

tracking mechanism, :term:`PDF`-generation, column contents transformation,

etc.) you need to create a set of special tables. Those tables can be located

in your own database, or in a central database for a multi-user installation

(this database would then be accessed by the controluser, so no other user

should have rights to it).

Zero configuration

------------------

In many cases, this database structure can be automatically created and

configured. This is called “Zero Configuration” mode and can be particularly

useful in shared hosting situations. “Zeroconf” mode is on by default, to

disable set :config:option:`$cfg['ZeroConf']` to false.

The following three scenarios are covered by the Zero Configuration mode:

\* When entering a database where the configuration storage tables are not

present, phpMyAdmin offers to create them from the Operations tab.

\* When entering a database where the tables do already exist, the software

automatically detects this and begins using them. This is the most common

situation; after the tables are initially created automatically they are

continually used without disturbing the user; this is also most useful on

shared hosting where the user is not able to edit :file:`config.inc.php` and

usually the user only has access to one database.

\* When having access to multiple databases, if the user first enters the

database containing the configuration storage tables then switches to

another database,

phpMyAdmin continues to use the tables from the first database; the user is

not prompted to create more tables in the new database.

Manual configuration

--------------------

Please look at your ``./sql/`` directory, where you should find a

file called \*create\\_tables.sql\*. (If you are using a Windows server,

pay special attention to :ref:`faq1\_23`).

If you already had this infrastructure and:

\* upgraded to MySQL 4.1.2 or newer, please use

:file:`sql/upgrade\_tables\_mysql\_4\_1\_2+.sql`.

\* upgraded to phpMyAdmin 4.3.0 or newer from 2.5.0 or newer (<= 4.2.x),

please use :file:`sql/upgrade\_column\_info\_4\_3\_0+.sql`.

and then create new tables by importing :file:`sql/create\_tables.sql`.

You can use your phpMyAdmin to create the tables for you. Please be

aware that you may need special (administrator) privileges to create

the database and tables, and that the script may need some tuning,

depending on the database name.

After having imported the :file:`sql/create\_tables.sql` file, you

should specify the table names in your :file:`config.inc.php` file. The

directives used for that can be found in the :ref:`config`.

You will also need to have a controluser

(:config:option:`$cfg['Servers'][$i]['controluser']` and

:config:option:`$cfg['Servers'][$i]['controlpass']` settings)

with the proper rights to those tables. For example you can create it

using following statement:

.. code-block:: mysql

GRANT SELECT, INSERT, UPDATE, DELETE ON <pma\_db>.\* TO 'pma'@'localhost' IDENTIFIED BY 'pmapass';

.. \_upgrading:

Upgrading from an older version

+++++++++++++++++++++++++++++++

\*\*Never\*\* extract the new version over an existing installation

of phpMyAdmin; we had evidence of problems caused by this.

Simply copy :file:`config.inc.php` from your previous installation into

the newly unpacked one. Configuration files from old versions may

require some tweaking as some options have been changed or removed.

For compatibility with PHP 5.3 and later, remove a

``set\_magic\_quotes\_runtime(0);`` statement that you might find near

the end of your configuration file.

You should \*\*not\*\* copy :file:`libraries/config.default.php` over

:file:`config.inc.php` because the default configuration file is version-

specific.

If you have upgraded your MySQL server from a version previous to 4.1.2 to

version 5.x or newer and if you use the phpMyAdmin configuration storage, you

should run the :term:`SQL` script found in

:file:`sql/upgrade\_tables\_mysql\_4\_1\_2+.sql`.

If you have upgraded your phpMyAdmin to 4.3.0 or newer from 2.5.0 or

newer (<= 4.2.x) and if you use the phpMyAdmin configuration storage, you

should run the :term:`SQL` script found in

:file:`sql/upgrade\_column\_info\_4\_3\_0+.sql`.

Do not forget to clear the browser cache and to empty the old session by

logging out and logging in again.

.. index:: Authentication mode

.. \_authentication\_modes:

Using authentication modes

++++++++++++++++++++++++++

:term:`HTTP` and cookie authentication modes are recommended in a \*\*multi-user

environment\*\* where you want to give users access to their own database and

don't want them to play around with others. Nevertheless be aware that MS

Internet Explorer seems to be really buggy about cookies, at least till version

6. Even in a \*\*single-user environment\*\*, you might prefer to use :term:`HTTP`

or cookie mode so that your user/password pair are not in clear in the

configuration file.

:term:`HTTP` and cookie authentication

modes are more secure: the MySQL login information does not need to be

set in the phpMyAdmin configuration file (except possibly for the

:config:option:`$cfg['Servers'][$i]['controluser']`).

However, keep in mind that the password travels in plain text, unless

you are using the HTTPS protocol. In cookie mode, the password is

stored, encrypted with the AES algorithm, in a temporary cookie.

Then each of the \*true\* users should be granted a set of privileges

on a set of particular databases. Normally you shouldn't give global

privileges to an ordinary user, unless you understand the impact of those

privileges (for example, you are creating a superuser).

For example, to grant the user \*real\_user\* with all privileges on

the database \*user\_base\*:

.. code-block:: mysql

GRANT ALL PRIVILEGES ON user\_base.\* TO 'real\_user'@localhost IDENTIFIED BY 'real\_password';

What the user may now do is controlled entirely by the MySQL user management

system. With HTTP or cookie authentication mode, you don't need to fill the

user/password fields inside the :config:option:`$cfg['Servers']`.

.. index:: pair: HTTP; Authentication mode

HTTP authentication mode

------------------------

\* Uses :term:`HTTP` Basic authentication

method and allows you to log in as any valid MySQL user.

\* Is supported with most PHP configurations. For :term:`IIS` (:term:`ISAPI`)

support using :term:`CGI` PHP see :ref:`faq1\_32`, for using with Apache

:term:`CGI` see :ref:`faq1\_35`.

\* See also :ref:`faq4\_4` about not using the :term:`.htaccess` mechanism along with

':term:`HTTP`' authentication mode.

.. index:: pair: Cookie; Authentication mode

.. \_cookie:

Cookie authentication mode

--------------------------

\* Username and password are stored in cookies during the session and password

is deleted when it ends.

\* With this mode, the user can truly log out of phpMyAdmin and log

back in with the same username.

\* If you want to allow users to enter any hostname to connect (rather than only

servers that are configured in :file:`config.inc.php`),

see the :config:option:`$cfg['AllowArbitraryServer']` directive.

\* As mentioned in the :ref:`require` section, having the ``mcrypt`` extension will

speed up access considerably, but is not required.

.. index:: pair: Signon; Authentication mode

.. \_auth\_signon:

Signon authentication mode

--------------------------

\* This mode is a convenient way of using credentials from another

application to authenticate to phpMyAdmin to implement single signon

solution.

\* The other application has to store login information into session

data (see :config:option:`$cfg['Servers'][$i]['SignonSession']`) or you

need to implement script to return the credentials (see

:config:option:`$cfg['Servers'][$i]['SignonScript']`).

\* When no credentials are available, the user is being redirected to

:config:option:`$cfg['Servers'][$i]['SignonURL']`, where you should handle

the login process.

The very basic example of saving credentials in a session is available as

:file:`examples/signon.php`:

.. literalinclude:: ../examples/signon.php

:language: php

Alternatively you can also use this way to integrate with OpenID as shown

in :file:`examples/openid.php`:

.. literalinclude:: ../examples/openid.php

:language: php

If you intend to pass the credentials using some other means than, you have to

implement wrapper in PHP to get that data and set it to

:config:option:`$cfg['Servers'][$i]['SignonScript']`. There is very minimal example

in :file:`examples/signon-script.php`:

.. literalinclude:: ../examples/signon-script.php

:language: php

.. seealso::

:config:option:`$cfg['Servers'][$i]['auth\_type']`,

:config:option:`$cfg['Servers'][$i]['SignonSession']`,

:config:option:`$cfg['Servers'][$i]['SignonScript']`,

:config:option:`$cfg['Servers'][$i]['SignonURL']`

.. index:: pair: Config; Authentication mode

Config authentication mode

--------------------------

\* This mode is sometimes the less secure one because it requires you to fill the

:config:option:`$cfg['Servers'][$i]['user']` and

:config:option:`$cfg['Servers'][$i]['password']`

fields (and as a result, anyone who can read your :file:`config.inc.php`

can discover your username and password).

\* In the :ref:`faqmultiuser` section, there is an entry explaining how

to protect your configuration file.

\* For additional security in this mode, you may wish to consider the

Host authentication :config:option:`$cfg['Servers'][$i]['AllowDeny']['order']`

and :config:option:`$cfg['Servers'][$i]['AllowDeny']['rules']` configuration directives.

\* Unlike cookie and http, does not require a user to log in when first

loading the phpMyAdmin site. This is by design but could allow any

user to access your installation. Use of some restriction method is

suggested, perhaps a :term:`.htaccess` file with the HTTP-AUTH directive or disallowing

incoming HTTP requests at one’s router or firewall will suffice (both

of which are beyond the scope of this manual but easily searchable

with Google).

.. index:: pair: Swekey; Authentication mode

.. \_swekey:

Swekey authentication mode

--------------------------

The Swekey is a low cost authentication USB key that can be used in

web applications. When Swekey authentication is activated, phpMyAdmin

requires the users's Swekey to be plugged before entering the login

page (currently supported for cookie authentication mode only). Swekey

Authentication is disabled by default. To enable it, add the following

line to :file:`config.inc.php`:

.. code-block:: php

$cfg['Servers'][$i]['auth\_swekey\_config'] = '/etc/swekey.conf';

You then have to create the ``swekey.conf`` file that will associate

each user with their Swekey Id. It is important to place this file

outside of your web server's document root (in the example, it is

located in ``/etc``). Feel free to use it with your own users'

information. If you want to purchase a Swekey please visit

`https://www.phpmyadmin.net/auth\\_key/ <https://www.phpmyadmin.net/auth\_key/>`\_

since this link provides funding for phpMyAdmin.

A self documented sample file is provided in the

file :file:`examples/swekey.sample.conf`:

.. literalinclude:: ../examples/swekey.sample.conf

:language: sh

.. seealso:: :config:option:`$cfg['Servers'][$i]['auth\_swekey\_config']`

Securing your phpMyAdmin installation

+++++++++++++++++++++++++++++++++++++

The phpMyAdmin team tries hard to make the application secure, however there

are always ways to make your installation more secure:

\* Remove the ``setup`` directory from phpMyAdmin, you will probably not

use it after the initial setup.

\* Properly choose an authentication method - :ref:`cookie`

is probably the best choice for shared hosting.

\* In case you don't want all MySQL users to be able to access

phpMyAdmin, you can use :config:option:`$cfg['Servers'][$i]['AllowDeny']['rules']` to limit them.

\* Consider hiding phpMyAdmin behind an authentication proxy, so that

users need to authenticate prior to providing MySQL credentials

to phpMyAdmin. You can achieve this by configuring your web server to request

HTTP authentication. For example in Apache this can be done with:

.. code-block:: apache

AuthType Basic

AuthName "Restricted Access"

AuthUserFile /usr/share/phpmyadmin/passwd

Require valid-user

Once you have changed the configuration, you need to create a list of users which

can authenticate. This can be done using the :program:`htpasswd` utility:

.. code-block:: sh

htpasswd -c /usr/share/phpmyadmin/passwd username

\* If you are afraid of automated attacks, enabling Captcha by

:config:option:`$cfg['CaptchaLoginPublicKey']` and

:config:option:`$cfg['CaptchaLoginPrivateKey']` might be an option.