6.x WordPress v0.1 - ACLs Fail

This approach is not viable. There is a snag with how ACLs work documented in the issues area. Instead, I am going back to a simplified approach which will then make separation of duty achievable by using container technology.

What this article and the sub-version 128 is good for, is exploration of ACLs and it's limitations.

Introduction

As an application, WordPress is easy to use and feature rich. It has an established community, and in my opinion, the best selection of themes and the most usable blogging cms software package available.

However, though the initial setup seems fast and easy, as an enterprise administrator, it is very insecure. This article provides instructions on setting up a secure WordPress that caters to **multiple** clients. A simpler single client setup will be drawn out of this article.

This approach to setting up WordPress is complex because it focuses on Information Security and uses ACLs. As a trade for the complexity, you end up with a WordPress environment that,

1. Segregates different clients from each other.
2. Minimizes damage if WordPress itself is hacked.
3. Has Two Factor Authentication.
4. Scales quickly.
5. Is Easy to backup and recover.

Though all the steps here show how to do the work, it is helpful to have a good understanding of basic Unix permissions and go through the ACLs tutorial.

WordPress may use an array of technologies. The stack selected for the Bonsai Framework is,

- Web Server = Apache

- Application Server = PHP
- Database = MySQL
- OS = Ubuntu

Install Packages

Install the packages to run WordPress,

```
sudo apt-get install php5 # Installs PHP
sudo apt-get install mysql-server # Installs MySQL
sudo apt-get install php5-mysql # Libraries to connect PHP to MySQL
```

As of Ubuntu 12 (an maybe even earlier), the installer will automatically restart Apache2 for you.

During the MySQL install, you will be prompted for the **root** administration database password. If following the Bonsai Framework, use your standard password algorithm based on the server name.

This article was written and tested against Ubuntu 12.04 which has ACL support built into the kernel. With Ubuntu 12.04, if the ACL packages are not installed the install is straight forward,

```
sudo apt-get install acl
```

Previous versions may be more involved and require modifying the **fstab** file.

Setup Directory Structure with ACLs

This security approach works at various levels to contain process and programs to run only within their specified folders and restrict users to their own respective instances of WordPress.

Permission and Group Structure

We want to provide website hosting for two different clients, The Daily Planet and LexCorp. Employees from the respective companies will belong in the system under the following groups, wgdailyplanet and wglexcorp. Here is how the top level structure,

User Name	Assigned User	Group	Web Root Directory	File Access	Directory Access
dailyplanet01	Clark Kent	wgdailyplanet	/opt/web/php/dailyplanet.com/	Read, Write and Execute	Read, Write and Execute
lexcorp01	Lex Luthor	wglexcorp	/opt/web/php/lexcorp.com/	Read, Write and Execute	Read, Write and Execute
	Apache Server	www-data	/opt/web/php/dailyplanet.com/ /opt/web/php/lexcorp.com/	Read	Read and Execute (required to transverse directories)
	Other			No Access	No Access

We do not want employees from different companies access or even have awareness of each others web directory. At the same time, the Apache Server running as user www-data belonging to group www-data also needs access to the directories. We also want to grant users of the staff group read access for support purposes. Finally, we want all subsequent directories and files under the respective Web Root Directories to inherit the same permissions.

This is just not possible using standard Unix permissions.

As such we will require ACLs. Here is what each directory will look like and how the main user types will interact with directories and files,

Directory	Permissions	ACL	Comment	User from wgailyplanet	User from wglexcorp	User from staff	Normal (other) Users

/opt/web/	rwXr-X--X	n/a	Don't need ACLs here. Use Unix permissions "drwxr-x--x 3 serveradmin staff".	cd but no ls	cd but no ls.	cd, ls	
/opt/web/php/	rwXr-X--X	n/a	Don't need ACLs here. Use Unix permissions "drwxr-x--x 3 serveradmin staff".	cd but no ls.	cd but not ls.	cd, ls	
/opt/web/php/tmp/	rwXr-X---	www-data:rwX	In a shared environment lock down. Consider ACLs to make it easy for staff to review.	Nothing	Nothing	cd, ls	nothing
/opt/web/php/logs/	rwXr-X---	www-data:rwX	In a shared environment lock down. Consider ACLs to make it easy for staff to review.	Nothing (on the one hand this increases security but how does a php programmer debug?)	Nothing	cd, ls	nothing
/opt/web/php/dailyplanet.com/	rwXrwX---	www-data:rX wgdailyplanet:rwX			Nothing	cd, ls	nothing
/opt/web/dailyplanet.com/blog/	rwXr-X---	www-data:rX wgdailyplanet:rwX				cd, ls	nothing
/opt/web/dailyplanet.com/blog/wp-content/	rwXrwX---	www-data:rwX wgdailyplanet:rwX	In order to install plugins, www-data needs write access. Notice the base Unix permissions have to be more open due to how masking works with ACLs.			cd, ls	nothing
/opt/web/php/lexcorp.com/	rwXr-X---	www-data:rX wglexcorp:rwX				cd, ls	nothing
/opt/web/lexcorp.com/blog/	rwXr-X---	www-data:rX wglexcorp:rwX				cd, ls	nothing
/opt/web/lexcorp.com/blog/wp-content/	rwXrwX---	www-data:rwX wglexcorp:rwX	In order to install plugins, www-data needs write access. Notice the base Unix permissions have to be more open due to how masking works with ACLs.			cd, ls	nothing

Further to this, using normal Unix permissions, all directories will be owned by serveradmin:staff.

web Folder

This is where everything starts for web related work,

```
cd /opt/
sudo mkdir ./web/
sudo chown -R serveradmin:staff ./web/
#sudo chmod -R g+s ./web/ # Subsequent files and folder should inherit group. Ask DK if this is a good idea.
#sudo chmod -R u+s ./web/ # Subsequent files and folder should inherit user but this does not seem to work. Ask DK if this is a good idea.
# Next set the permissions.
sudo chmod u+rwX,g+r-w+X,o-rw+X ./web/
# Give other groups the appropriate access via acls.
sudo setfacl -Rm g:www-data:rX ./web/
sudo setfacl -Rm g:staff:rX ./web/
# With ACLs the mask sets the maximum permissions allowed.
# Even if other entries, such as giving a group rwX are set higher, permissions will be downgraded to what the mask allows.
# This is what causes the Effective comment seen when mask collides with umask.
# sudo setfacl -Rm m:rwx ./web/
```

Verify permissions for the web folder,

```
getfacl web
# file: web
# owner: serveradmin
# group: staff
user::rwx
group::r-x
group:www-data:r-x
group:staff:r-x
mask::r-x
other::---
```

Now look at the defaults of the folder the default affects the folders and files created within this folder,

```
getfacl --default web
# file: web
# owner: serveradmin
# group: staff
```

To apply the permissions set to the folder to defaults,

```
getfacl --access ./web/ | sudo setfacl -d -RM - ./web/
```

getfacl --access = retrieves the ACL the permissions applied to the directory only (default permissions are not returned). The details are then piped to setfacl and the parameters read,

-d = Change default permissions for newly created files and folder.
-M = Take as input files. Because the dash is used, the file is instead standard input.
R = Apply changes recursively to folders and files.

The default ACLs should now be changed,

```
getfacl --default ./web/
# file: web/
# owner: serveradmin
# group: staff
user::rwx
group::r-x
group:www-data:r-x
group:staff:r-x
mask::r-x
other::---
```

php Support Folders

This is where all php code will execute. In php centric applications this will also be considered the web root for static files too and reflected in the virtual host configuration.

Create Folders

PHP requires advanced access to specific folders,

```
cd /opt/web/
sudo mkdir php
sudo chown -R serveradmin:staff ./php/
sudo chmod -R u+rwX,g+r-w+X,o-r-w+X ./php/
```

Notice the ACLs step is skipped. The php folder will inherit the permissions of allowing www-data access from the default ACLs defined in the web folder.

Next create the php support directories,

```
cd /opt/web/php
sudo mkdir ./tmp/ ./logs/
sudo chown serveradmin:staff ./tmp/ ./logs/
sudo chmod -R u+rwX,g+r-w+X,o-rwx ./tmp/ ./logs/

sudo setfacl -Rm g:www-data:rwX ./tmp/ ./logs/
getfacl --access ./tmp/ | sudo setfacl -d -RM - ./tmp/ ./logs/
```

The PHP process run under Apache as www-data needs full access to these folder.

Configure PHP to Use Specified Folders

Edit php.ini to make use of the folders.

```
sudo vi /etc/php5/apache2/php.ini
```

Set Running Directory

Search for the open_basedir line and modify to include the directories setup for WordPress,

```
; open_basedir, if set, limits all file operations to the defined directory
; and below.  This directive makes most sense if used in a per-directory
; or per-virtualhost web server configuration file. This directive is
; *NOT* affected by whether Safe Mode is turned On or Off.
; http://php.net/open-basedir
open_basedir = /opt/web/php
```

This helps minimizes the amount of damage that can be done in the event that the system is compromised to the specified directory.

Set Temp Directory

Because **open_basedir** has been set, WordPress no longer has access to the general temporary folder it expects which is required for certain operations (for example to upload plugins through the Administrator web interface).

Modify php.ini further by modifying the upload_tmp_dir line,

```
; Temporary directory for HTTP uploaded files (will use system default if
not
; specified).
; http://php.net/upload-tmp-dir
upload_tmp_dir = /opt/web/php/tmp/
```

Make Changes Take Effect

Restart Apache for the changes to take effect,

```
sudo service apache2 restart
```

You will now find that php scripts will only run in the designated directories specified in **php.ini**.

Setup Website Root

Each website will have it's own root folder under /opt/web/php/. To keep things simple we will keep both static and php content inside of this folder.

Each website specific folder will only allow users belonging to the correct group to enter and work with the directory,

```
cd /opt/web/php
sudo mkdir dailyplanet.com
sudo chown -R serveradmin:staff dailyplanet.com
```

Now to add the necessary groups to their respective virtual hosts,

```
sudo setfacl -Rm g:wgdailyplanet:rwX dailyplanet.com
sudo getfacl --access ./dailyplanet.com/ | sudo setfacl -d -RM -
./dailyplanet.com/
```

Repeat as needed for each website.

Symbolic Link for Group Accounts

Due to the tight security, the directory listing for /opt/web/ and /opt/web/php/ is restricted. So users will not know about the web root.

There are a few solutions to this. In the case where users should only be working with a specific website, that users home directory can be set to the web root.

> Put details here.

In some cases, you may have more than one site be managed by the same work group. In that case, it is simpler to create symbolic link in that user's home directory.

In this example, there is a general web master user, web.master who belongs to both the wgdailyplanet and wglexcorp groups. As a sudo enabled user,

```
sudo su - web.master # Log in as web.master
cd ~
ln -s /opt/web/php/dailyplanet.com dailyplanet.com
ln -s /opt/web/php/lexcorp.com lexcorp.com
```

Setup WordPress

WordPress is incredibly easy to setup and there are many shorter tutorials than this.

This tutorial, takes a more secure approach

> WordPress out of the box can be very quickly broken into. In fact, I personally go so far as to keep the Ubuntu firewall up with port 80 or 443 closed until WordPress is completely hardened. When the setup steps required using the browser, I use ssh tunnelling to access 80 securely. A writeup of using ssh tunnelling should be added to the Bonsai Framework and linked or included here.

Apply ACLs to WordPress Folders

Using the serveradmin account, download and decompress WordPress,

```
cd ~
wget http://wordpress.org/latest.tar.gz
tar -xvpf latest.tar.gz
mv ./wordpress/ ./blog/ # Rename as we not need to make the technology
obvious.

# We MUST give group read, execute and write access. We want, using ACLs,
to grant respective client's group full access to modify files. Because
ACLs are limited by the mask, the group mask needs to be open.
sudo chmod -R u+rwX,g+rwX,o-rwx ./blog/

umask 0002 # Again we need the umask open for ACLs.
cp -R ./blog/ /opt/web/php/dailyplanet.com/ # Copy NOT move so default ACLs
defined in /opt/web/php/ are applied.
```

As a staff user with sudo access,

```
sudo chown -R serveradmin:staff /opt/web/php/dailyplanet.com/blog/
```

....

.... Below this is being rewritten ....

....

At this point the blog directory **should** be setup with the correct permissions inherited, however, there is some kind of odd behaviour,

```
sudo getfacl ./blog/
# file: blog
# owner: serveradmin
# group: staff
user::rwx
group::r-x
group:www-data:r-x
group:staff:r-x
group:wgdailyplanet:rwx        #effective:r-x
mask::r-x
other::---
default:user::rwx
default:group::r-x
default:group:www-data:r-x
default:group:staff:r-x
default:group:wgdailyplanet:rwx
default:mask::rwx
default:other::--x
```

Notice the **#effective:r-x** comment prevents users from wgailyplanet to be able to write to the blog directory.

Fix this,

```
sudo setfacl -Rm g:wgdailyplanet:rwX ./blog/
sudo getfacl --access ./blog/ | sudo setfacl -d -RM - ./dailyplanet.com/
# Still investigating...
# Strangely doing the above using the group www-data also fixed the problem... retest at exactly what point...
```

Now it's fixed,

```
file: blog/
# owner: serveradmin
# group: staff
user::rwx
group::r-x
group:www-data:r-x
group:staff:r-x
group:wgmainframe:rwx
mask::rwx
other::---
default:user::rwx
default:group::r-x
default:group:www-data:r-x
default:group:staff:r-x
default:group:wgmainframe:rwx
default:mask::rwx
default:other::---
```

ACLs Do not Work as Thought

March 25, 2013

Ok, how about watching a directory and adjusting permissions whenever a file is changed or updated,

http://askubuntu.com/questions/43846/how-to-put-a-trigger-on-a-directory

March 22, 2013

This IBM article shows how to change the sftp umask for specific groups using 5.4p1 or higher,

```
Match Group  <group name>
ForceCommand  internal-sftp  -u 73
```

BUT, after testing, realized, I cannot use the umask on sftp to change the permission on a file to be less restrictive than the original file.

March 19, 2013

OK more bad news, sftp and scp use root's umask. There's is a way to globally set the umask but that's not what I want and causes security issues. This link which talks about chrooting specific sftp groups for no shell access looks promising possibly in concert with PAM. Otherwise I might look at file watchers.

March 18, 2013 Evening

Finally found the answer and it's ACLs will not work. Per this post which points out that in the man documentation for ACL(5) in the paragraph OBJECT CREATION AND DEFAULT, the access of ACL of a file ojbect is initialized only when create(), mkdir(), mknod(), mkfifo(), or open() functions are used. **In other words default ACLs are only applied when using create.**

Otherwise it will be a XOR of the existing user mask and default acl mask of the user doing the copy, unzip, untarred ect...) followed by the usual calculation done by subtracting the current umask from 0666 for a file and 0777 for a directory. This is why the #effective permission comment appears.

March 18, 2013 Morning

Found mask of newly created files in the directory are alright. The problem is copying files from another directory where the mask by default is more restrictive. Maybe I need to set sticky bits.

Set sticky bits and it did not make a difference.

This article explains how ACL permissions are calculated and how to determine what a command like touch or cp does, http://unix.stackexchange.com/questions/56278/can-not-explain-acl-behavior

March 17, 2013

Verified his is due to an incorrect mask.... need to run a test but this should work after.

March 14, 2013

According to this article the problem is the mask in the ACL. It says the mask for user, group or default shown shows the maximum permissions allowed. Even when set higher the mask downgrades the permissions to what the mask allows. OK so how to change the mask and if say I set the mask to rwx, does that make it too open or does the ACL kick in and overrides?

Doing more testing creating new files under ../blog/ as the mf.webmaster user belonging to wgmaingrame, I notice the permissions are not correct. More of the effective problem. Directories seem to be fine though.

March 13, 2013

Notice that the mask also changed from r-x to rwx. This might be the key. Actually more test shows a copy of some default created folders and files still has problem and it points to "cp -R" with the "-R" or "-r" both being problems by not respecting ACLs or bringing in the default unix permissions as the problem.

Further Refine File Permissions

Here are how the file permissions will look,

Directory	Purpose	user:group:other	ACL Group	Default ACL Group	Tech Notes
dailyplanet.com/blog/		rwX:r-X:---	Inherited	Inherited	
dailyplanet.com/blog/wp-content		Inherited	Inherited. with, default:group:www-data:rwX	Inherited with, default:group:www-data:rwX	www-data needs full access to download and install plugins and themes.

To setup these permissions with ACLs as a staff user,

```
cd /opt/web/php/dailyplanet.com/blog/
sudo setfacl -Rm g:www-data:rwX ./wp-content/
getfacl --access ./wp-content/ | sudo setfacl -d -RM - ./wp-content/
```

The ACLs ensures that WordPress can install plugins and themes.

Configure MySQL

Secure MySQL

As a staff user run the Secure Installation script included with MySQL,

```
sudo mysql_secure_installation
```

The prompts are very straightforward. Except for "Change the Root password?", answer yes to all prompts by hitting Enter,

```
NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MySQL
      SERVERS IN PRODUCTION USE!  PLEASE READ EACH STEP CAREFULLY!

In order to log into MySQL to secure it, we'll need the current
password for the root user.  If you've just installed MySQL, and
you haven't set the root password yet, the password will be blank,
so you should just press enter here.
Enter current password for root (enter for none):
OK, successfully used password, moving on...
Setting the root password ensures that nobody can log into the MySQL
root user without the proper authorisation.
You already have a root password set, so you can safely answer 'n'.
Change the root password? [Y/n] n
 ... skipping.
By default, a MySQL installation has an anonymous user, allowing anyone
to log into MySQL without having to have a user account created for
them.  This is intended only for testing, and to make the installation
go a bit smoother.  You should remove them before moving into a
production environment.
Remove anonymous users? [Y/n]
 ... Success!
Normally, root should only be allowed to connect from 'localhost'.  This
ensures that someone cannot guess at the root password from the network.
Disallow root login remotely? [Y/n]
 ... Success!
By default, MySQL comes with a database named 'test' that anyone can
access.  This is also intended only for testing, and should be removed
before moving into a production environment.
Remove test database and access to it? [Y/n]
 - Dropping test database...
 ... Success!
 - Removing privileges on test database...
 ... Success!
Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.
Reload privilege tables now? [Y/n]
 ... Success!
Cleaning up...
All done!  If you've completed all of the above steps, your MySQL
installation should now be secure.
Thanks for using MySQL!
```

For now that's it to securing MySQL.

Connect

Connect into MySQL,

```
mysql -u root -p
```

The password to use is the password set during the MySQL install. If everything goes well you will be in the MySQL shell,

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 36
Server version: 5.5.24-0ubuntu0.12.04.1 (Ubuntu)
Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights
reserved.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input
statement.
mysql>
```

The remainder of this section happens inside of the mysql shell.

Create the WordPress Database and Accounts in MySQL

List the databases to makes sure what you want to create does not already exists,

```
SHOW DATABASES;
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
+--------------------+
3 rows in set (0.16 sec)
```

In our case we should have nothing other than the three default databases.

Enter the following MySQL commands,

```
CREATE DATABASE wpdailyplanetdb;
GRANT ALL PRIVILEGES ON wpdailyplanetdb.* TO 'wpdpdbuser'@'localhost'
IDENTIFIED BY 'password';
FLUSH PRIVILEGES;
```

Adjust the variables for your application.

wpdailyplanetdb - Name of the database for the WordPress application instance. We use the domain name of the website.

wpdpdbuser - User account for accessing the database cannot be longer than 16 characters

localhost - Address of the database server. In this example, the database is on the same server so use localhost.

password - Change to password using algorithm based on name of the primary website domain, in this case dailyplanet.

Database Admins will not like granting all privileges. After the initial setup is done we will restrict to more minimal privileges.

Exit MySQL Shell

Exit the MySQL shell,

```
EXIT
```

Configure WordPress

Create Config File for Database Access

Launch a browser and hit the WordPress setup page for your machine at http://dailyplanet.com/blog/wp-admin/install.php and you will be prompted to create a configuration file.

Click the button, **"Create a Configuration File"**.

The next prompt reminds you of all the critical information you will need. The Bonsai Framework takes a high security posture, so the automatic file creation should not work. Click the **"Let's go!"** button.

Enter the required information and click "Submit",

Field	Value	Comment
Database Name	wpdailyplanetdb	The Bonsai Framewok appraoch is to base the user name on the site's primary domain name.
User Name	wpdpdbuser	
Password		This is the application password set during the wpdailyplanetdb database creation step.
Database Host	localhost	Address of the database server. In this example, the database is on the same server so use localhost is used.
Table Prefix	bf_	The Bonsai Framework approach generally does not encourage changing an application's table prefix. However, given the architecture of WordPress and popularity it is recommended to change the prefix to something other than wp_ to make the system less susceptible to attacks.

It is expected that you will receive a message that WordPress can not write the wp-config file and the following prompt will appear on screen,

> Sorry, but I can't write the wp-config.php file.
>
> You can create the wp-config.php manually and paste the following text into it.

Copy the generated wp-config.php file.

> Some shortcuts to ensure you get it all fast,
>
> 1. Click inside of the text box
> 2. Use the keyboard combo CTRL-A (to select all)
> 3. CTRL-C (to copy)

Go to your shell, load your favourite editor and paste the contents of the wp-config.php file,

```
vi /opt/web/php/dailyplanet.com/blog/wp-config.php
```

Also while you are there add the following to the bottom of the file,

```
# WordPress determines write access when it finds a match between the Web
Server ID and the directory being written too.
# This approach does not work with ACLs. During manual file upload the
failure results in a prompt for FTP login information.
# Because we are using ACLs, override the write check by forcing a direct
write.
define('FS_METHOD', 'direct');
```

This is to allow the ACL approach. Yes we could make the owner of the wp-content directory www-data, but that breaks the general approach by the Bonsai Framework.

> I'm not convinced this is the best resolution and it seems to me like a hack. File a bug with WordPress to get ACL information instead. In the interim, consider breaking the Bonsai Framework approach and just chowning www-data as the user with rwX.

Once you have saved the file, go back to your browser and click "Run the install".

Enter Site Information

Finally enter the site information,

Field	Value	Comment
Site Title	dailyplanet	We like to reference our domain name.
Username	tempadmin	You probably do not want to use the default admin for username. WordPress (as of Sep 2012) out of the box, has no facilities to stop dictionary attacks against the administration system. Admin will be the first username guessed by automated attacks. Because the username put here will show up in the default site generated, this will be a temporary administrator account.
Password		As mentioned, WordPress has no facilities to stop dictionary attacks. On top of that, the default setup exposes your administrator account name on the Internet. Choose a **very very long** and **complex** password. (Anyone know of a good site that shows how quickly an entered password would be broken with a dictionary attack, put the link here)
Your E-mail		Whatever email is chosen here, it will not be the final one used by the real administrator account. Keeping in mind that WordPress does not allow duplicate emails, in this example, the administrator will use a personal email and then use a proper email account when the real administrator account is created.
Privacy		This depends on the purpose of your website. Unless this is a private site that should not show up on Google, leave it checked.

Click, "Install WordPress" which should result in a success screen. At this point you are actually done the setup. Do not click **"Log In".**

Customize WordPress

At this point WordPress is already working. There are two urls to take note of,

URL	Area	Purpose
http://www.dailyplanet.com/blog/	Public	You can hit this url right now and see a default working site. This url is where your users will enter.
http://www.dailyplanet.com/blog/wp-admin/	Administration	This url results from clicking the "Log In" button after the WordPress install is complete. It can also be accessed through the Public homepage by click "Log In" located at the bottom right under "META". The Administration area allows the customization and configuration of WordPress. Also, once logged into the administration, if you browse to the public area, you will see additional buttons and options to create posts and edit the website contents.

If you have the Install WordPress Success Screen still up, click **"Log In"** will take you to the Word Press Administration url or use the url in the table above.

Minimal Security - Block Login Attacks

WordPress out of the box can be easily broken into with a brute force dictionary attack for the following combined reasons,

1. The administrator account is available on the public portion of the blog as part of the default sample content.
2. Nothing prevents the brute force attack from trying again and again on the WordPress Administration login page.
3. The Administration page well known.

For these reasons it is best to not make the website publicly available until secured or at least, as mentioned choose a very complex password.

Install one of these plugins.

Plugin	Description	Review
Google Authenticator	The Google Authenticator plugin for WordPress gives you two-factor authentication using the Google Authenticator app for Android/iPhone/Blackberry. If you are security aware, you may already have the Google Authenticator app installed on your smartphone, using it for two-factor authentication on your Gmail or Google Apps account. The two-factor authentication requirement can be enabled on a per-user basis. You could enable it for your administrator account, but log in as usual with less privileged accounts. If You need to maintain your blog using an Android/iPhone app, or any other software using the XMLRPC interface, you can enable the App password feature in this plugin, but please note that enabling the App password feature will make your blog less secure.	Very good plugin. Tricky part is making sure time is synced with same time servers across the phone and server. For example, my iphone was off by 2 minutes because it was set manually to Toronto. Best thing to do is turn on the 4 minute drift allowance. When setting the password make sure there are no spaces otherwise the barcode will not work.
Duo Two-Factor Authentication	This plugin enables Duo Security's two-factor authentication for WordPress logins. Duo provides simple two-factor authentication as a service via: • Phone callback • SMS-delivered one-time passcodes • Duo mobile app to generate one-time passcodes • Duo mobile app for smartphone push authentication • Duo hardware token to generate one-time passcodes This plugins allows a WordPress administrator to quickly add strong two-factor authentication to any WordPress instance without setting up user accounts, directory synchronization, servers, or hardware.	Free signup but it looks like only 1000 transactions for the life of the account. Looks very professional.
BAW More Secure Login	Grid Cards	
Login Security Solution	Most useful feature I find when used with Google Authenticator is that it blocks user for x number of minutes progressively more as more attempts are tried. Also blocks by cookie and ip.	

Should have link to how to ssh in to disable plugins if they misbehave.

Set Up Users

The default user created is an administrator and has more privileges than necessary. The very first step is to create users with specific roles provided by WordPress. The roles are outlined below in order of most privileges to least.

Keep in mind that when creating accounts, Wordpress requires unique email addresses.

Role	Description	UserName
Administrator	Administrators have access to all the administration features.	setupadmin
Editor	Editors can publish posts, manage posts as well as manage other people’s posts, etc.	perrywhite

Author	Authors can publish and manage their own posts, and are able to upload files.	clarkkent, loislane
Contributor	Contributors can write and manage their posts but not publish posts or upload media files.	jimmyolsen
Subscriber	Subscribers can read comments/comment/receive newsletters, etc. but cannot create regular site content.	lexluthor

(explain why we do not use the first admin account we created) Creat the **real** administrator account,

Field	Value	Comment
Site Title	dailyplanet	We like to reference our domain name.
Username	setupadmin	This will be the real administration account. Steps to delete tempadmin will follow shortly.
Password		As mentioned, WordPress has no facilities to stop dictionary attacks. On top of that, the default setup exposes your administrator account name on the Internet. Choose a **very very long** and **complex** password. (Anyone know of a good site that shows how quickly an entered password would be broken with a dictionary attack, put the link here)
Your E-mail	admin@bonsaiframework.com	If there is more than one administrator, you should have a general support email box that only administrators have access to. This email address will be used for password recovery purposes.

...

Past this point is not yet organized or complete.

Lock Down WordPress

WordPress and PHP simply due to the model is inherently insecure when compared to more Enterprise solutions.

As such the Bonsai Framework takes an administrator approach to managing and securing WordPress. Though these instructions allow for multiple clients to co-host, it is recommended to **not** use a co-hosting model for clients that require enterprise privacy. This is especially problematic if clients are granted shell access. It becomes very complex to protect one client from gaining access to another client's WordPress data.

WordPress updates through the built in admin interface will fail unless the restrictions are relaxed. With this security approach, privileges must be temporarily be granted as part of the upgrade process.

Restrict WordPress Database Account

Now that WordPress is setup we can lock down the application database account. As part of good application security, the WordPress application database account should only be granted minimal privileges. Note that during upgrades you will have to increases privileges again.

Connect into MySQL,

```
mysql -u root -p
```

Enter the following MySQL commands,

```
REVOKE ALL PRIVILEGES, GRANT OPTION FROM 'wpdpdbuser'@'localhost';
GRANT SELECT, INSERT, UPDATE ON wpdailyplanetdb.* TO
'wpdpdbuser'@'localhost' IDENTIFIED BY 'password';
FLUSH PRIVILEGES;
```

Adjust the variables for your application.

wpdailyplanetdb - Name of the database for the WordPress application instance. We use the domain name of the website.

wpdpdbuser - User account for accessing the database.

localhost - Address of the database server. In this example, the database is on the same server so use localhost.

password - Change to password using algorithm based on name of the website domain, in this case dailyplanet.

Verify the changes took effect,

```
SHOW GRANTS FOR 'wpdpdbuser'@'localhost';
```

> This point onwards still needs to be flushed out.

File Permissions

From the WordPress article Hardening WordPress we will want to take the approach of creating accounts for select developers or release managers.

wp-config.php - holds the database password and should be locked down (it is thanks to the ACLs)

> Covered on the Ubuntu WordPress guide, for automatic updates to occur, the folder and all its files and sub-folders must be owned by www-data with write access. The Bonsai Framework does not endorse this approach. Client administering the WordPress requiring this functionality should understand the security implications.

...

Writing Next Topics

- Repeat for the second instance.
- Upgrades and Updates

FAQ

Why do some of the php5 installations say to use install libapache2-mod-php5?

The instructions may be old. At least with Ubuntu 12.04 there is no need for this package because it is included with the php5 package.

What is the difference between the php5 and libapache2-mod-php5 packages?

Nothing I can see. It just looks like php5 is an overarching package name.

References

Setup

Ubuntu Server Documentation - https://help.ubuntu.com/12.04/serverguide/php5.html

Security

Has some ok details around suPHP - https://help.ubuntu.com/community/ApacheMySQLPHP#Installing_MYSQL_with_PHP_5

Some good notes on securing PHP from Symantec - http://www.symantec.com/connect/articles/securing-php-step-step

Start some good security practices for WordPress - http://www.howtospoter.com/web-20/wordpress/triple-p-of-total-wordpress-security

Wordpress discussion on permissions, based their recomendations for suPHP the community does not really understand permissions - http://codex.wordpress.org/Changing_File_Permissions

This restricts the php process to specific directories - http://help.godaddy.com/article/1616

At least by looks this looks like it may be a good guide to securing wordpress - http://wpsecure.net/basics/

Wordpress' official article to getting started after setup - http://codex.wordpress.org/First_Steps_With_WordPress

Article which resolved manual upload issue - http://www.charleshooper.net/blog/wordpress-auto-upgrade-and-dumb-permissions/

> Determine if this actually increases security - http://www.suphp.org/Home.html. suPHP and LiteSpeed make the most sense for shared hosting.
>
> This article indicates that suphp is slow as it makes php run as a cgi. Instead a poster recommended using what is available with mod_php - http://serverfault.com/questions/279938/should-i-use-suphp-or-mod-php-for-shared-hosting. Along this thread another poster recommends, http://mpm-itk.sesse.net/ which allows vhosts to be run under different uid and gid.
>
> A great discussion on using permissions, same conclusion I was coming to using www-data group - http://unix.stackexchange.com/questions/30879/what-user-should-apache-and-php-be-running-as-what-permissions-should-var-www
>
> Probably the most complete but also complex solutions is to use ACLs - http://serverfault.com/questions/339948/user-permissions-for-both-apache-and-local-user/357977