OWASP Wordpress Security Implementation Guideline

From OWASP

Considerations

This project aims for a unified approach on WordPress security design and implementation. It is definitely more than a checklist, it's a guide for secure implementation and an invitation to consider and to analyze each individual case.

There is a long list of recommended resources for securing aspects of the WordPress implementation. The project is aimed to offer open source or free resources instead of commercial ones. Some plugins have a free version and a paid one that offers extra functionality. In such cases, the focus of the project was on the free version.

General security

This section is meant to be just a reminder that all the other hardening measures are useless if an attacker can gain access to WordPress users' computers. We're not going to spend the time and effort to go into details but rather enumerate the common good practices each security conscious user should have in mind. There are plenty of good resources to help anyone accomplish security basics.

Device security

When we talk about devices capable of accessing the WordPress administration interface we don't just talk about computers but mobile devices as well. The following is a list of items that needs to be taken into account when securing the devices that will be accessing the WordPress instances. Some of them may refer to PCs and mobile devices, others just to one of the devices.

- Password protect the device
- Use strong passwords
- Keep the OS updated
- Encrypt the storage
- Have an anti-virus installed and updated
- Have a malware/spyware scanner installed and perform regular scans and updates
- Have a firewall installed and configured
- Secure your browser (http://www.cert.org/historical/tech_tips/securing-web-browser-index.cfm)

Infrastructure security

Before hardening the core of WordPress an implementer must consider hardening the services on which the instance will be installed. Sometimes the underlying infrastructure is not under the control of the implementer. While there are things that can be hardened on WordPress to mitigate things that are supposed to be fixed on the infrastructure side, one should always consider defense in depth. The implementer can contact the infrastructure administrator and ask for specific hardening in order to further protect the applications that will be installed on top of that, in this case WordPress.

The foundation of infrastructure hardening is operating system hardening. This is a broad subject and highly dependent on the OS, the main concerns being around privileges, access control, authentication and logging. It's a topic outside the coverage of the current project and these are things that must be covered by experienced System Administrators.

WordPress can be installed on a multitude of platforms but the main focus below is on the most common components, Apache and MySQL. The general rules though apply to all supported infrastructure components.

Following best design practices, the tiers of the WordPress instance should be separated. However the presentation and application layers of WordPress are bound together. Thus only one separation is possible, the one with the database. For small applications it's not a common practice, but for larger sites this becomes a must from a security but also a performance perspective.

As was the case with general security, this is just a list of things that should be performed in order to harden the infrastructure and not the means to do it.

Apache hardening

- Update regularly
- Disable directory listing
- Secure the communication with the server by generating and using SSL certificates
- Disable unnecessary modules
  - Good candidates for this are: *userdir*, *suexec*, *cgi/cgid*, *include*, *autoindex*
- Run the daemon as a separate user and group
- Use *Allow* and *Deny* to restrict access to directories
- Use *mod_security* module to secure Apache
- Disable following of *symbolic links*
- Turn off server sides includes and CGI execution
- Limit request size
- Configure other settings like *TimeOut*, *MaxClients*, *KeepAliveTimeout*, *LimitRequestFields*, *LimitRequestFieldSize* in order to prevent DoS attacks
- Enable and configure proper logging
- Modify server banner

PHP hardening

- Update regularly
- Don't install PHP as a CGI binary
- Disable unnecessary PHP modules
- Disable unused potentially dangerous PHP functions (good examples: *exec*,*passthru*,*shell_exec*,*system*, etc.)
- Log errors internally
- Disable verbose error reporting on the client side
- Turn off remote code execution (if it's not needed; the core WordPress doesn't need this functionality)
- Disable magic quotes
- Limit PHP access to file system
- Protect from DoS
  - Control POST size
  - Limit script time execution
  - Limit memory usage
- Consider implementing the Suhoshin security extension (http://www.suhosin.org/stories/index.html)

- Hide the version of PHP in use
- Hide the .php extension

MySQL hardening

There is an entire OWASP project dedicated to MySQL hardening (https://www.owasp.org/index.php/OWASP_Backend_Security_Project_MySQL_Hardening). The main action items are:

- Update regularly
- Disable or restrict remote access
- Filesystem access restrictions and ACLs
- Designing a chroot-jail
- Encrypting network traffic (this is a must if the database layer is physically separated from the application layer)
- Encrypting raw databases on filesystem level
    - Redundant if disk encryption is in place at the OS layer
    - However, by using *dmcrypt*, one can generate an extra layer of encryption
- Backup encryption
- Configuration
    - Connectivity: maximum number of concurrent connections and related settings
    - Logging
    - Access control and privilege management
    - Set up root password
    - Rename root account
    - Delete unused users and databases
    - Remove installation history

A PHP security checker is available here (https://github.com/sektioneins/pcc). This is a one-page php file designed to analyze PHP configuration and rank the findings based on severity.

Remote access

- Don’t use FTP (use sFTP where possible)
- If SSH access is available, use scp (http://linux.die.net/man/1/scp) or WinSCP (http://winscp.net/eng/index.php) for file transfer
- Consider using VPN or SSH tunnels (http://www.pentest.ro/ssh-tunnels-an-alternative-to-vpn/) to the server for accessing the WordPress administrative interface

WordPress security

There are three main components of WordPress that need to be considered from a security perspective when implementing the solution.

- Core – the basic default installation files that provide most of the functionality
- Plugins – special written code to improve and extend the basic functionality
- Theme – the presentation layer which may come with some limited extended functionality

Updates

It is of vital importance to keep WordPress core, plugins and themes updated. Once an update is released, it needs to be applied as soon as possible to close any security holes.

Functional problems with updates must be considered. It is possible that an update will break some of the functionality so a backup is recommended before updating the core.

WordPress Core

The WordPress core has three different types of updates:

- Core development updates, known as the "bleeding edge"
- Minor core updates, such as maintenance and security releases
- Major core release updates

Starting with version 3.7, automatic background updates were introduced by default for minor core updates releases (generally security updates). This default behavior can be overridden by editing the wp-config.php file and adding or modifying the following statement

define('WP_AUTO_UPDATE_CORE', true);

When set to true all updates will be enabled. Translations are updated by default with the minor core updates.

Themes and Plugins

The themes and plugins can be updated automatically using filters. The best place to put a filter is in a must-use plugin (http://codex.wordpress.org/Must_Use_Plugins). WordPress doesn’t recommend putting filters in the wp-config.php file because of conflicts with other parts of the code.

To enable automatic updates for themes and plugins, add the following code

add_filter('auto_update_plugin', '__return_true');

add_filter('auto_update_theme', '__return_true');

Removal of unused plugins and themes

Depending on the server configuration, the files in the WordPress folder can be accessed from the Internet regardless of whether they are used or not. Even if a plugin is disabled, the files are still there and they are accessible from the Internet.

When a new vulnerability is discovered, the attackers write scripts to look for the vulnerable files. Knowing the location of vulnerable plugins increases their chances of infiltrating a vulnerable instance.

Any plugins and themes that are not actively used must be deleted.

Plugins & Themes Security

Plugins and themes are a great addition to the functionality offered by the WordPress core. WordPress’ success is based on these elements. It’s easy to develop a new theme, add new functions with plugins. This ease of development comes with the security downside. In the rush for functionality, the developers often forget about security. Looking at the CVE list for WordPress (https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=wordpress) it's worth noticing that in the past years most of the security defects are affecting the plugins and themes and not WordPress core.

Developing on top of WordPress should be regarded as a regular development job and follow a standard secure development lifecycle. Concrete action items for this chapter include source code review and penetration testing of plugins and themes.

When choosing to use an already developed plugin by a 3rd party, a security audit should be performed. Good differentiators for available plugins are:

- Publication in the official plugin store at https://wordpress.org/plugins/
- User ratings and comments
- Version number (is it a young plugin/theme or has it faced the challenges of time?)
- Last update
- Update frequency
- Compatibility with the current version of the WordPress core

In order to perform a source code audit, the following tools can be used:

- RIPS (http://rips-scanner.sourceforge.net/)
- PHP-sat (http://www.program-transformation.org/PHP/PhpSat)
- Yasca (http://www.scovetta.com/yasca.html)
- Manual analysis using grep (http://resources.infosecinstitute.com/finding-bugs-in-php-using-grep/), GrepBugs (https://grepbugs.com/)

Things to pay extra attention during the source code audit:

- Obfuscated code
- BASE64 encode function
- System call functions (exec, passthru, system, shell_exec, etc.)
- PHP code execution (eval, assert, preg_replace, etc.)
- Information disclosure functions (phpinfo, getenv, getmygid/pid/uid, etc.)
- Filesystem functions (fopen, bz/gzopen, chgrp/own/mod, etc.)

Backup

The backup process is essential. The configuration of the backup process can make the distinction between a clean and fast recovery or a loss of data and prolonged downtime.

What needs to be included in the backup?

- The WordPress Files
    - WordPress Core Installation
    - WordPress Plugins
    - WordPress Themes
    - Images and Files
    - JavaScript and PHP scripts, and other code files
    - Additional Files and Static Web Pages
- The Database

It's easy to say that a full backup of the /public_html folder is needed. However there are situations in which this is not feasible nor enough. There are situations in which large quantities of data is generated in the public folder (statistics, temporary data, etc.) that is useless in the backup process. There's also the situation in which configuration files are placed outside the public directory. They also need backup.

The plan is to identify the files and folders that must be part of the backup process and save these in a remote location.

For database backup, the mysql command line can be used or administrative interfaces like phpMyAdmin.

How often should the backup be performed? It all depends on how often the instance is updated from a content perspective. If there are multiple updates a day, it's a good idea to have a daily backup. If there's a new article every several days, than a weekly or monthly backup is the way to go.

It's a good practice to keep multiple backups and have them time stamped. This is because a breach might not be noticed immediately and a clean recovery can only be performed from a backup which is several iterations old.

Verifying that the backup is functional is part of the process. A backup that does not allow quick and full recovery is useless. The idea is to have a clean server and perform a full recovery from the backup, then check all the functionality and make sure nothing is missing.

Automation

The steps above are manual and labor intensive. There is a full list of plugins that can help this process: https://wordpress.org/plugins/tags/backup

The one free alternative offering full backup capabilities that stands out of the list is BackWPup (https://wordpress.org/plugins/backwpup/). The free version can be used to save your complete installation including /wp-content/ and push it to an external Backup Service, like Dropbox, S3, FTP (not a good idea) and many more.

From a security perspective, it's worth noticing that an attacker who compromised the installation may be able to retrieve credentials and access the remote location of the backups, thus being able to manipulate or delete them. As a good precaution, on the remote side where the backups are stored, an independent process should take the backups and move them to a location inaccessible from the WordPress installation.

User roles and proper usage

Understanding the roles and properly assigning them to users is essential in the segregation of duties process.

The WordPress roles are:

- Super Admin – somebody with access to the site network administration features and all other features
- Administrator – somebody who has access to all the administration features within a single site
- Editor – somebody who can publish and manage posts including the posts of other users
- Author – somebody who can publish and manage their own posts
- Contributor – somebody who can write and manage their own posts but cannot publish them
- Subscriber – somebody who can only manage their profile

The least privilege principle must be considered when assigning roles.

A full list of privileges and a comparison between roles is available at http://codex.wordpress.org/Roles_and_Capabilities.

Supporting plugins:

- Members Plugin (https://wordpress.org/plugins/members/)
- Role Scoper Plugin (https://wordpress.org/plugins/role-scoper/)
- User Access Manager (http://wordpress.org/extend/plugins/user-access-manager/)
- Advanced Access Manager (http://wordpress.org/extend/plugins/advanced-access-manager/)
- User Role Editor (http://wordpress.org/extend/plugins/user-role-editor/)

Restrict the access to the admin interface

Restricting the access to the admin interface should be considered as no regular user is in need of access to this area. For a site with few users it makes sense to whitelist their IP addresses. Additionally, the access can be restricted only to the localhost and have the users VPN in or create a tunnel to the server (if SSH is enabled) and then access the admin interface.

To restrict the access to the wp-admin folder, a file called .htaccess needs to be created in that folder. The content of the file should be:

Order Deny,Allow

Deny from all

Allow from 127.0.0.1

Multiple IP addresses separated by whitespaces can be added and the use of wildcards (*) is permitted.

Prevent brute-forcing

Brute-forcing is the easy way in for an attacker. As discussed in the General Security chapter, a prerequisite for preventing bruteforcing is to have strong passwords (https://www.owasp.org/index.php/Authentication_Cheat_Sheet#Password_Complexity). Apart from that, an additional layer of protection can be added in the form of CAPTCHA (http://en.wikipedia.org/wiki/CAPTCHA).

One good plugin candidate is Google Captcha (reCAPTCHA) (https://wordpress.org/plugins/google-captcha/). The advantage of this plugin is that it can be used to add the extra layer of protection on other areas as well (like registration and comments).

CAPTCHA is not a perfect solution by any means. There are services offering real-time CAPTCHA solving for a few cents per challenge. However it takes seconds to solve a CAPTCHA even for a good service like this, thus this sort of attack becomes unfeasible.

Another preventive measure is to lock-out accounts after a series of failed attempts. There is no plugin at the moment that can lock a user after several failed attempts for a period of time, there are plugins blocking IP addresses that are brute-forcing the login mechanism. This approach is not the best when dealing with distributed attacks.

Implement two factor authentication

To add another layer of security on the authentication mechanism, two factor authentication can be enabled. Two factor authentication is a method of securing accounts requiring that you not only know something (a password) to log in but also that you possess something (your mobile device). The benefit of this approach to security is that even if someone guesses your password, they need to have also stolen your possession in order to break into your account.

Supporting plugins:

- Clef (unfortunately the product was retired in April 2017)
- Google Authenticator
- MiniOrange and other 2FA plugins

https://en-gb.wordpress.org/plugins/tags/2fa/

Remove or change the default administrator account

There are two main reasons for creating a new administrator or modifying the old one:

- After the installation the default username is “admin”; an attacker trying to brute-force his way in will try default usernames
- The default id of the admin account is 1; an attacker who discovers a SQL injection is will try to update the user with id = 1

Both tasks can be performed manually in the database without the need to delete the admin account or can be performed in the administration User Interface. Create a new administrator, log in with the new credentials and delete the default one.

Disable user registration if not needed

If user management is performed manually or through integration with other user management systems, there is no need for this functionality to be enabled in WordPress.

To disable user registration, log in as an administrator, go to **Settings -> General** and make sure the **“Anyone can register”** box is unchecked.

Change the database prefix

In case a 0-day SQL injection vulnerability is discovered, an attacker will try to exploit the known tables from a default WordPress installation. To prevent this from happening, the default prefix of the tables needs to be changed. This can be performed in several ways:

- During the installation process
- Manually via *mysql* command line or *phpMyAdmin* for all the tables; after this, the wp-config.php file must be configured to reflect the changes ($table_prefix = "ves1uaq3_";)
- With a plugin (Change DB Prefix (https://wordpress.org/plugins/db-prefix-change/))

Control comments

WordPress was initially a blogging platform so the ability to add comments was part of the success story. Things changed with the shift of WordPress towards a CMS so comments might not be necessary in all instances. There are several things that need to be considered when dealing with this topic:

- Are comments needed? If not, they should be disabled. Log in as administrator. For new posts go to **Settings -> Discussion** and uncheck "**Allow people to post comments on new articles**". For existing posts, go to **Posts**, select all of them, **Bulk Actions -> Edit** and choose "**do not allow**" near **Comments** before hitting **Update posts**.
- If comments are required, who should be able to post them? If only registered users should be allowed to add comments, go to **Settings -> Discussion** and check the "**Users must be registered and logged in to comment**" box.
- Should comments be reviewed before publishing? If so, the "**Comment must be manually approved**" box must be checked.
- If comments are not reviewed before publishing, using an anti-spam plugin like the default Akismet (https://wordpress.org/plugins/akismet/) is advised

As a general rule of thumb, all the options under **Settings -> Discussion** should be carefully reviewed.

Check file permissions

Permissions on files and directories determine who is allowed to read, write and execute them. Permission settings will vary from situation to situation and between shared hosting and dedicated hosting.

Following is a list of desired permissions on sensitive items and fallback options:

- wp-config.php
  - Desired: 400
  - Fallback: 440, 600, 640
- uploads folder
  - Desired: 755
  - Fallback: 766, 777 (not recommended)
- .htaccess files
  - Desired: 400
  - Fallback: 440, 444, 600, 640

Delete readme.html and install.php

The readme.html file may reveal sensitive information and is not needed from a functional perspective.

The install.php is a residue of the installation process and even though it does not allow it to be restarted it's not needed and should be removed.

The license.txt reveals the year of last Wordpress update - a fact that attackers can use to scan for outdated WordPress installations.

Action item:

- Delete the /<WordPress_root>/readme.html /<WordPress_root>/license.txt and /<WordPress_root>/wp-admin/install.php files

Add blank index.php files where needed

Especially in shared environments where the settings of the web server are outside the control of the WordPress implementer, directory listing might be enabled. To add an extra layer of security, blank index.php files should be added to the folders that don't have indexes in order to prevent browsing of the resources. The main folders that need to be considered are:

- wp-includes
- wp-content
- wp-content/plugins
- wp-content/themes
- wp-content/uploads

Move wp-config.php file outside the web root folder

The wp-config.php file is a very important configuration file. It contains a lot of sensitive information about your WordPress site, like your database information for example.

WordPress will automatically look for this file in the folder above the WordPress root folder if it does not exist in the root folder. Moving this file out of the public_html folder means the file will not be accessible from the Internet.

Create secret keys

Starting with the release of WordPress 2.6, a new set of security features for passwords and password hashing and cookie security is included. This feature works without doing anything, but it's not particularly powerful without some extra steps. In order to greatly increase the security of the WordPress installation, secret keys must be set up. This should be part of the standard installation process. Whenever there's suspicion that the secret keys have been compromised, the administrator must change them. Changing the secret keys will invalidate all sessions so users will need to re-authenticate.

Setting up or changing secret keys can be done by adding or editing the following lines to the wp-config.php file, right after the other define statements:

define('AUTH_KEY', 'put your unique phrase here');

define('SECURE_AUTH_KEY', 'put your unique phrase here');

define('LOGGED_IN_KEY', 'put your unique phrase here');

define('NONCE_KEY', 'put your unique phrase here');

You don't have to remember the keys, just make them long, random and complicated -- or better yet, use the online generator (https://api.wordpress.org/secret-key/1.1/salt/).

Enforce transport layer encryption for administrative tasks

It was discussed earlier that SSL should be configured and used to access the WordPress instance. Usually sites are available over port 80 and 443. This means that the users are free to choose if they use a clear text or an encrypted communication channel.

In order to force the usage of SSL (at least) for sensitive actions, the following lines must be added to the wp-config.php file:

define('FORCE_SSL_LOGIN', true);

define('FORCE_SSL_ADMIN', true);

Use a Web Application Firewall (WAF)

A WAF should be in place at the web server layer. Because that is not always accessible to the implementer, a WAF plugin can be used to add this layer of protection.

A good plugin candidate is NinjaFirewall (https://wordpress.org/plugins/ninjafirewall/).

Security plugins

This section is a list of security plugins and a short description of their functionality. As previously mentioned, the focus is on free plugins.

- iThemes Security (https://wordpress.org/plugins/better-wp-security/) – iThemes Security (formerly Better WP Security) gives you over 30+ ways to secure and protect your WordPress site. In its free version it can obscure, detect, protect and recover a WordPress installation
- BulletProof Security (https://wordpress.org/plugins/bulletproof-security/) – the free version offers:
  - .htaccess Website Security Protection (Firewalls)
  - Login Security & Monitoring
  - DB Backup
  - DB Backup Logging
  - DB Table Prefix Changer
  - Security Logging
  - HTTP Error Logging
  - FrontEnd/BackEnd Maintenance Mode
- All In One WP Security & Firewall (https://wordpress.org/plugins/all-in-one-wp-security-and-firewall/)
  - User Account/Login/Registration Security
  - Database & File System Security
  - htaccess and wp-config.php File Backup and Restore
  - Blacklist Functionality
  - Firewall Functionality
  - Brute-force login attack prevention
  - Security Scanner
- Sucuri Security - Auditing, Malware Scanner and Security Hardening (https://wordpress.org/plugins/sucuri-scanner/)
  - Security Activity Auditing
  - File Integrity Monitoring
  - Remote Malware Scanning
  - Blacklist Monitoring
  - Effective Security Hardening

    - Post-Hack Security Actions
    - Security Notifications
    - Website Firewall (add on)
- Wordfence Security Plugin (https://en-gb.wordpress.org/plugins/wordfence/)
    - Web Application Firewall
    - Threat Defence Feed
    - Real-time blocking of known attackers
    - Rate-limiting
    - Login Security
    - File integrity monitoring and scanning
    - Real-time Monitoring

Disable the Plugin and Theme Editor

Occasionally you may wish to disable the plugin or theme editor to prevent overzealous users from being able to edit sensitive files and potentially crash the site. Disabling these also provides an additional layer of security if a hacker gains access to a well-privileged user account.

Open your wp-config.php file and add the following constant:

define('DISALLOW_FILE_EDIT',true);

Keep a WordPress Activity Log

Logging is very important, especially on multi user platforms such as WordPress. The OWASP Top 10 project lists insufficient logging & monitoring as one of the most common security flaws in web applications.

By default WordPress does not keep an activity log (audit log) of what logged-in users do on the website. So you can use a plugin to keep a record of the activity of logged-in users. Some of the security plugins mentioned in the previous section have some logging capabilities, though they are very basic.

Below are some of the most popular WordPress activity log plugins you can work with if you want better coverage:

WP Security Audit Log (https://www.wpsecurityauditlog.com/)

- comprehensive WordPress activity log solution
- supports WordPress multisite networks
- most advanced coverage in terms of what it can keep a log of
- out of the box support for popular WordPress plugins such as WooCommerce, bbPress and Advanced Custom Fields (ACF).

WP Stream (http://wp-stream.com/)

- easy to use
- ideal for users who may not need detailed activity log
- WP-CLI command interface for querying records
- integrates with Slack

Simple History (https://wordpress.org/plugins/simple-history/)

- ideal for users who may not need detailed activity log
- has Limit Login Attempts included
- activity log is available via a RSS feed
- easy to integrate with

Large-scale integration

Implementing one WordPress site and maintaining it is a doable job for an administrator. In large corporate environments there may be hundreds of instances that need management, configuration and maintenance. This can easily become an unmanageable situation. When dealing with large number of instances, a centralized approach is needed.

Creating a standard image

The first step is to create a standard WordPress installation with all the security configuration and plugins in place. This should be a blank installation with no data that can be easily replicated when a new instance needs to be created.

A process for new instances must be in place and approach at least the following subjects:

- General configuration
- Database connectivity
- Setting the administrator account

LDAP integration & Single Sign On

User management for large WordPress sites can be a hassle. In corporate environments users are in general centrally managed and assigned to different groups. WordPress can make use of this already established situation. Whether it's Active Directory (http://en.wikipedia.org/wiki/Active_Directory) or other LDAP compatible service, this establishment is already used in the organization trying to implement WordPress. It's easy to set up groups based on WordPress roles and assign users to different groups, based on their required level of access. Once the integration is achieved, one can go further towards an elegant solution by implementing Single Sign On.

Supporting plugins:

- Active Directory Integration (https://wordpress.org/plugins/active-directory-integration/)
- Active Directory SSO (https://wordpress.org/support/plugin/active-directory-sso)
- Simple LDAP Login (https://wordpress.org/plugins/simple-ldap-login/)

Multisites

A large environment requires multiple instances of WordPress. Managing each individual instance can become impossible for a single person or a small team. This is where a built-in feature of WordPress comes in handy, multisite or network of sites (http://codex.wordpress.org/Create_A_Network).

A multisite network can be very similar to a personal version of WordPress.com. End users can create their own sites on demand, just like end users of WordPress.com can create blogs on demand. If there's no need to allow end users to create their own sites on demand, the administrator of the network can create a multisite network in which only he can add new sites.

A multisite network is a collection of sites that all share the same WordPress installation. They can also share plugins and themes. The individual sites in the network are virtual sites in the sense that they do not have their own directories on your server, although they do have separate directories for media uploads within the shared installation, and they do have separate tables in the database.

WordPress does a good job in providing the necessary documentation for:

- Installation (http://codex.wordpress.org/Create_A_Network)
- Administration (http://codex.wordpress.org/Multisite_Network_Administration)
- Debugging (http://codex.wordpress.org/Debugging_a_WordPress_Network)
- Migration (http://codex.wordpress.org/Migrating_Multiple_Blogs_into_WordPress_3.0_Multisite)

The benefit of the multisite feature is centralized management of security. Plugins can be checked once for security defects and when a stable and secure version is available it can be pushed to all the sites in the same time.

This built-in solution might not always be the best choice. For example, all the plugins are shared between different sites and the administrators of those sites choose which plugins to enable and which to disable.

Unified management of multiple installations

If multiple separate instances of WordPress need to be managed centrally, there are several solutions (most of them have at least some form of commercial addons) that can accomplish the task:

- InfiniteWP (http://infinitewp.com/) is a free, self-hosted multiple WordPress management platform that simplifies WordPress management tasks into simple clicks. Features:
  - One master login
  - One click updates
  - Instant backup & restore
  - Plugins & themes management
- MainWP (https://mainwp.com/) is an open source, free and self-hosted WordPress management platform similar to InfiniteWP.
- ManageWP (https://managewp.com/)
- WPRemote (https://wpremote.com/) lets administrators monitor an unlimited number of WordPress websites. Through the WP Remote dashboard they can update WordPress and update plugins and themes. A snapshot (backup) of the websites can be downloaded from the interface

Resources

The project started with a discussion between Dan Vasile (https://www.linkedin.com/in/dancatalinvasile) (the initiator) and Anders Vinther (https://www.linkedin.com/in/andersvinther) who has already published a guide (http://www.wpsecuritychecklist.com/) about secure WordPress implementation. Based on the information there, a part of the skeleton and content of the current project was created.

Browser security

- http://www.cert.org/historical/tech_tips/securing-web-browser-index.cfm

Apache hardening

- http://httpd.apache.org/docs/current/misc/security_tips.html
- http://www.tecmint.com/apache-security-tips/
- https://wiki.debian.org/Apache/Hardening

PHP hardening

- http://php.net/manual/en/security.php
- http://www.cyberciti.biz/tips/php-security-best-practices-tutorial.html
- http://www.suhosin.org/stories/index.html

MySQL hardening

- https://www.owasp.org/index.php/OWASP_Backend_Security_Project_MySQL_Hardening
- http://www.greensql.com/content/mysql-security-best-practices-hardening-mysql-tips

Wordpress

- http://codex.wordpress.org/Configuring_Automatic_Background_Updates
- http://stackoverflow.com/questions/3115559/exploitable-php-functions
- http://codex.wordpress.org/WordPress_Backups
- http://codex.wordpress.org/Roles_and_Capabilities
- http://en.support.wordpress.com/security/two-step-authentication/
- http://codex.wordpress.org/Create_A_Network
- http://codex.wordpress.org/Before_You_Create_A_Network
- http://codex.wordpress.org/Migrating_Multiple_Blogs_into_WordPress_3.0_Multisite
- http://codex.wordpress.org/Editing_wp-config.php

Project About

PROJECT INFO ***What does this OWASP project offer you?***	**RELEASE(S) INFO** ***What releases are available for this project?***
what ***is this project?*** ***Name:*** OWASP Wordpress Security Checklist Project (home page) ***Purpose:*** While there are several good articles on how to secure a Wordpress installation, there is no project on this topic on which people can discuss and contribute to a definitive and homogeneous checklist. ***License:*** Creative Commons Attribution ShareAlike 3.0 License (best for documentation projects)	**current release** Not Yet Published
who ***is working on this project?***	**last reviewed release**

Project Leader(s):

- Dan Vasile @ (mailto:Dan.Vasile@owasp.org)

Not Yet Reviewed

other releases

how ***can you learn more?***

Project Pamphlet: Not Yet Created

Project Presentation:

Mailing list: Mailing List Archives (https://lists.owasp.org/mailman/listinfo/owasp_wordpress_security_checklist_project)

Project Roadmap: View (https://www.owasp.org/index.php/Projects/OWASP_Wordpress_Security_Checklist_Project/Roadmap)

Key Contacts

- Contact Dan Vasile @ (mailto:Dan.Vasile@owasp.org) to contribute to this project
- Contact Dan Vasile @ (mailto:Dan.Vasile@owasp.org) to review or sponsor this project
- Contact the GPC to report a problem or concern about this project or to update information.

Retrieved from "https://www.owasp.org/index.php?title=OWASP_Wordpress_Security_Implementation_Guideline&oldid=242771"

Categories: OWASP Project | WordPress