WORDPRESS CONFIGURATION *CHEAT SHEET*

RIPSTECH

DISABLE DEBUGGING

The debug functionality should not be active in a production environment, as it might provide useful information to potential attackers.

```
1 | define( 'WP_DEBUG', false );
2 | #if WP_DEBUG_LOG is enabled, you have to enable WP_DEBUG as well
3 | define( 'WP_DEBUG_LOG', true );
4 | define( 'WP_DEBUG_DISPLAY', false );
```

The debug information is very helpful during the development of a WordPress application. However, it is important to make sure that the parameter is set back to false before loading the system onto a remote server. Errors should be logged, but they must not be visible to unauthorized users.

USE STRONG DATABASE CREDENTIALS

Each WordPress installation should have its own database with a dedicated user, and a secure and unique password.

```
1 | define( 'DB_NAME', 'unique_database_name' );
2 | define( 'DB_USER', 'unique_database_user' );
3 | define( 'DB_PASSWORD', 'strong_and_unique_password' );
```

When installing WordPress, a table prefix can be specified. This can allow you to run several installations using one database. However, sharing one database between different installations is dangerous because if one instance gets hijacked, the other installations will also be at risk. In addition, you should avoid using the database root account, as it has full access to all databases on the server.

USE UNIQUE KEYS AND SALTS

The salts and keys must be unique for each WordPress installation, as this is the only way to ensure secure user management.

```
1 | #These values are intentionally left blank to avoid copy-pastes
2 | define( 'AUTH_KEY',         '' );
3 | define( 'SECURE_AUTH_KEY',  '' );
4 | define( 'LOGGED_IN_KEY',    '' );
5 | define( 'NONCE_KEY',        '' );
6 | define( 'AUTH_SALT',        '' );
7 | define( 'SECURE_AUTH_SALT', '' );
8 | define( 'LOGGED_IN_SALT',   '' );
9 | define( 'NONCE_SALT',       '' );
```

The salts and keys are important for different functionalities in a WordPress application. Among other things for a secure login and a session management. The values are automatically chosen randomly during the installation. However, a WordPress installation might be duplicated. On deployment, it is advised to generate new values for these constants. These values can be retrieved via the following API.

https://api.wordpress.org/secret-key/1.1/salt/

USE SSL ENCRYPTION

No one should be able to read the traffic between you server and your users. Use SSL encryption and force WordPress to use only this type of connection.

```
1 | define( 'WP_SITEURL', 'https://www.mydomain.com' );
2 | define( 'WP_HOME', 'https://www.mydomain.com' );
3 |
4 | define( 'FORCE_SSL_ADMIN', true );
5 | define( 'FORCE_SSL_LOGIN',true );
```

It is now standard practice to encrypt your traffic. Many browser manufacturers mark un-encrypted connections as dangerous. Often web hosters offer their customers simple free certificates to enable SSL encryption. Also providers like Let's Encrypt offer free certificates.

RIPSTECH

WORDPRESS CONFIGURATION *CHEAT SHEET*

PROHIBIT DATABASE REPAIR

WordPress supports automatic database repair. This should not be possible without a backup of the database.

```
1 | define( 'WP_ALLOW_REPAIR', false );
```

The path wp-admin/maint/repair.php, which even unauthenticated users can access, triggers this automatic process. This might be helpful for a broken or fragmented database schema. However, this should never be done without a proper backup of the database.

DISALLOW UNFILTERED CONTENT

Admins and Editors are able to publish unfiltered HTML or files. If that is not absolutely necessary, activate these filters.

```
1 | define( 'DISALLOW_UNFILTERED_HTML', true );
2 | define( 'ALLOW_UNFILTERED_UPLOADS', false );
```

By default, Admins and Editors are able to write unfiltered HTML in post title, post content, and comments. The filetype filter can also be deactivated. This filter should remain activated.

ENABLE AUTO SECURITY UPDATE

Your WordPress should always be up to date so that it can resist the latest attacks. Activate the automatic security updates.

```
1 | define( 'AUTOMATIC_UPDATER_DISABLED', false );
2 | define( 'WP_AUTO_UPDATE_CORE', 'minor' );
```

Since WordPress is a wide-spread application, many attackers test for known vulnerabilities in outdated installations. Don't forget to perform a backup before each core upgrade, so only the minor update with current security patches should be installed automatically.

https://codex.wordpress.org/Configuring_Automatic_Background_Updates

BLOCK EXTERNAL REQUEST

Access from your installation to external resources should be restricted. It is advisable to only accept trusted sources.

```
1 | define( 'WP_HTTP_BLOCK_EXTERNAL', true );
2 | define( 'WP_ACCESSIBLE_HOSTS', 'api.wordpress.org,
            *.github.com, www.trusteddomain.com'  );
```

Not only the external access to WordPress should be secured, but also the access from the system to external resources. Here, the principle of white-list instead of blacklist applies.

DISALLOW FILE MODIFICATIONS

Prohibit the editing of code lines in plugins and themes by your admins and editors.
Deactivate the file editor for all extensions.

```
1 | define( 'DISALLOW_FILE_EDIT', true );
```

By manipulating code, a trusted extension can become a gateway for attackers.

DISALLOW FILE MODIFICATIONS

To harden your installation completely you can prevent any changes to your system by prohibiting the installation of plugins and themes.

```
1 | define( 'DISALLOW_FILE_MODS', true );
```

Themes and plugins can not only bring advantages for your WordPress. You must trust the creators of these extensions. Activate this barrier to prevent your admins and editors from loading untrusted extensions.