WP fail2ban Manual

Charles Lecklider

Aug 04, 2020

Contents

WP fail2ban is a WordPress plugin to write a myriad of events to syslog for integration with fail2ban.

CHAPTER 1

Introduction

1.1 History

As with many Open Source projects, *P fail2ban* started as way to scratch a particular itch. I had a dedicated server that was getting some unwelcome attention from various bots, and while it was trivial to configure *fail2ban* for `ssh` etc, WordPress was another story. Thus *WP fail2ban* was born late November 2011.

Since then it's slowly but steadily accumulated features, and much to my surprise, gained a considerable number of installs (30,000+ at the time of writing) - I really had no idea so many other people would be interested!

Between versions 3.5 and 3.6 there was a bit of a delay. I switched my development environment from Windows 10[1] to a FreeBSD workstation and a Linux laptop, life then decided to take its turn and get in the way for a bit, all while the shadow of Gutenberg loomed large over the future of WordPress. With the advent of ClassicPress[2] things started to look sunnier, so I dusted off the repo, put together some better documentation, braved the horrors of `svn`, and in November 2018 released 3.6 as a pseudo 7th anniversary present.

1.2 Future

Version 4 was born from a desire to visualise the things *WPf2b* was logging; being entirely separate and distinct from the core functionality, adding this as freemium features seemed like a good plan. Time will tell.

This logical separation will continue for all future versions - if you were happy with the way 3.6 worked you'll be happy with future versions too.

[1] It took me a while to realise that Microsoft really do want to turn Windows 10 into a toy, but I got there eventually.
[2] In the interests of full disclosure: I'm a Founding Committee Member and at the time of writing, Security Team Lead.

CHAPTER 2

Features

2.1 NEW - Multisite Support

Version 4.3 introduces proper support for multisite networks.

2.2 NEW - Block username logins

Sometimes it's not possible to block user enumeration (for example, if your theme provides Author profiles). Version 4.3 adds support for requiring the use of email addresses for login.

2.3 NEW - Filter for Empty Username Login Attempts

Some bots will try to login without a username. Version 4.3 logs these attempts and provides an "extra" filter to match them.

2.4 NEW - syslog Dashboard Widget

Ever wondered what's being logged? The new dashboard widget shows the last 5 messages; the Premium version keeps a full history to help you analyse and prevent attacks.

2.5 Remote Tools Add-on

The Remote Tools add-on provides extra features without adding bloat to the core plugin. For more details see the add-on page.

2.6 Support for 3rd-party Plugins

Version 4.2 introduced a simple API for authors to integrate their plugins with *WPf2b*, with 2 *experimental* add-ons:

- Contact Form 7
- Gravity Forms

2.7 CloudFlare and Proxy Servers

WPf2b can be configured to work with CloudFlare and other proxy servers. For a brief overview see *WP_FAIL2BAN_PROXIES*.

2.8 Comments

WPf2b can log both successful comments (see *WP_FAIL2BAN_LOG_COMMENTS*), and unsuccessful comments (see *WP_FAIL2BAN_LOG_COMMENTS_EXTRA*).

2.9 Pingbacks

WPf2b logs failed pingbacks, and can log all pingbacks. For a brief overview see *WP_FAIL2BAN_LOG_PINGBACKS*.

2.10 Spam

WPf2b can log comments marked as spam. See *WP_FAIL2BAN_LOG_SPAM*.

2.11 User Enumeration

WPf2b can block user enumeration. See *WP_FAIL2BAN_BLOCK_USER_ENUMERATION*.

2.12 Work-Arounds for Broken syslogd

WPf2b can be configured to work around most syslogd weirdness. For a brief overview see *WP_FAIL2BAN_SYSLOG_SHORT_TAG* and *WP_FAIL2BAN_HTTP_HOST*.

2.13 Blocking Users

WPf2b can be configured to short-cut the login process when the username matches a regex. For a brief overview see *WP_FAIL2BAN_BLOCKED_USERS*.

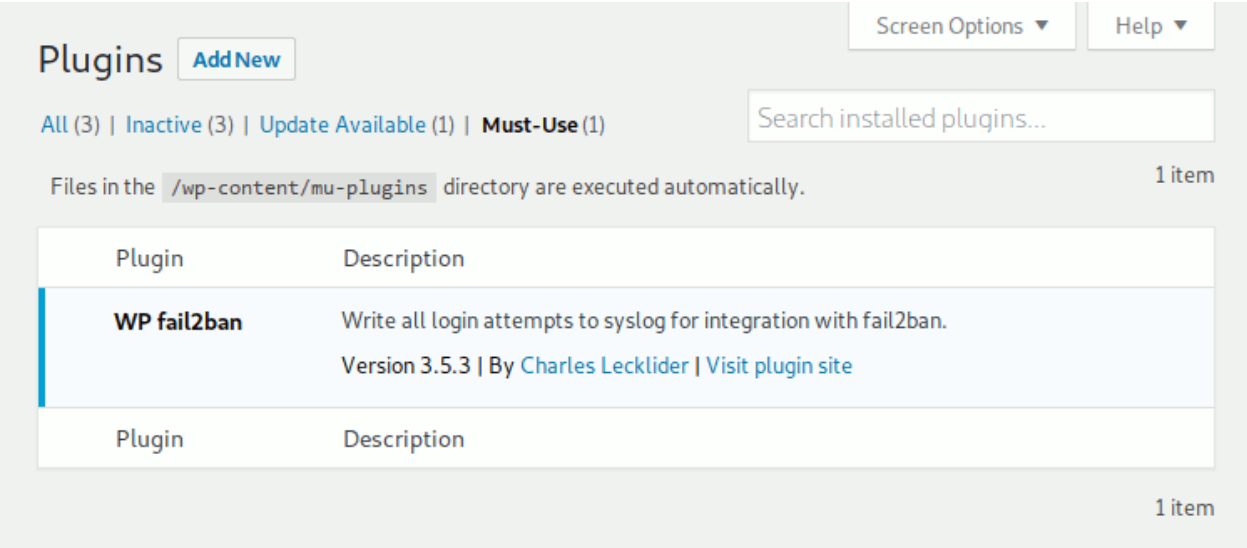
2.14 *mu-plugins* Support

WPf2b can easily be configured as a must-use plugin.

CHAPTER 3

Installation

3.1 Is *WP fail2ban* Already Installed?



WP fail2ban pre-installed in *mu-plugins* in a new DigitalOcean WordPress droplet.

3.2 Overview

WPf2b installs just like any other WordPress plugin - you need do nothing differently.

3.2.1 Premium

The Premium version installs via Freemius.

Database

Activating *WPf2b* Premium creates two database tables:

- `wp_fail2ban_log`
- `wp_fail2ban_plugins`

WPf2b Premium never drops the database tables - it's your data.

CHAPTER 4

Configuration

Now you have *WPf2b* installed and activated it's time to make it do something useful.

4.1 WP fail2ban

The Free version of *WPf2b* is configured by defining constants in `wp-config.php`. If you're using the Premium version, or you know your way around `wp-config.php` already, skip ahead to *Logging*.

The first step is to check you can edit your `wp-config.php` file. If you're not sure how to do that you'll need to contact your hosting provider - for now you can skip ahead to configuring *fail2ban*.

The second step is to **take a backup** of `wp-config.php`. We're not going to touch any other part of WordPress, so if anything goes wrong and your site stops working, restoring this backup should get you running again.

4.2 Logging

The key concept behind *WPf2b* is logging *Events* to `syslog`. If *WPf2b* doesn't log an Event, or logs it to the wrong place, `fail2ban` won't work as it should. If in doubt go with the defaults - they should work for most systems, and once you understand how the pieces fit together you can revisit this.

4.2.1 Choosing the Events to Log

If you're unfamiliar with `fail2ban` and `syslog` I recommend **not** enabling any extra logging to start with - skip ahead to configuring *fail2ban*. *WPf2b* automatically handles the most important things with sensible defaults that should work for most systems.

4.2.2 Advanced Users

Events

Over the years *WPf2b* has accumulated a lot of logging ability (and there're even more on the way):

Event	Reference
Auth OK	*WP_FAIL2BAN_AUTH_LOG*
Auth Fail	
Blocked User	*WP_FAIL2BAN_BLOCKED_USERS*
Blocked User Enumeration	*WP_FAIL2BAN_BLOCK_USER_ENUMERATION*
Blocked Username Login	*WP_FAIL2BAN_BLOCK_USERNAME_LOGIN*
Comment	*WP_FAIL2BAN_LOG_COMMENTS*
Comment: Spam	*WP_FAIL2BAN_LOG_SPAM*
Attempted Comment: Post not found	*WP_FAIL2BAN_LOG_COMMENTS_EXTRA*
Attempted Comment: Closed post	*WP_FAIL2BAN_LOG_COMMENTS_EXTRA*
Attempted Comment: Trash post	*WP_FAIL2BAN_LOG_COMMENTS_EXTRA*
Attempted Comment: Draft post	*WP_FAIL2BAN_LOG_COMMENTS_EXTRA*
Attempted Comment: Password-protected post	*WP_FAIL2BAN_LOG_COMMENTS_EXTRA*
Pingback	*WP_FAIL2BAN_LOG_PINGBACKS*
Pingback error	*WP_FAIL2BAN_PINGBACK_ERROR_LOG*

You should consider enabling *Comment: Spam* and *Attempted Comment: Closed post*, and, if you don't use WordPress's commenting system at all, you should enable **all** the *Attempted Comment* Events.

Facilities

By default, *WPf2b* uses the following `syslog` Facilities and *Levels*:

What	Default	Level
Auth OK	*LOG_AUTH*	INFO
Auth Fail		NOTICE
Blocked User		
Blocked User Enum		
Comment	*LOG_USER*	INFO
Comment: Spam	*LOG_AUTH*	NOTICE
Comment: Post not found		
Comment: Closed post		
Comment: Trash post		
Comment: Draft post		
Comment: Password-protected post		
Pingback	*LOG_USER*	INFO
Pingback error	*LOG_AUTH*	NOTICE

Unfortunately, there is no way of knowing *a priori* which Facility goes where. There is a table of default locations of *Logfiles* for various OSs; if you're running something not listed there and you know where the various Facilities go, please either submit a PR on GitHub, or let me know in the forum.

4.3 fail2ban

`fail2ban` can be tricky to configure correctly; with so many flavours of Linux it's impossible to provide anything but general guidance.

4.3.1 Filters

The filter files included are intended only as a starting point for those who want *WPf2b* to work "out of the box".

There is no "one size fits all" configuration possible for *fail2ban* - what may be a soft failure for one site should be treated as a hard failure for another, and vice versa. Careful thought should be given to what is appropriate for your environment.

Typical Settings

1. Copy *wordpress-hard.conf* and *wordpress-soft.conf* to your *fail2ban/filters.d* directory
2. Edit *jail.local* to include something like:

```
[wordpress-hard]
enabled = true
filter = wordpress-hard
logpath = /var/log/auth.log
maxretry = 1
port = http,https

[wordpress-soft]
enabled = true
filter = wordpress-soft
logpath = /var/log/auth.log
maxretry = 3
port = http,https
```

Note: Make sure you change `logpath` to the correct log for your OS. If your OS uses *systemd* you may need to install a real syslog service.

---

3. Reload or restart *fail2ban*

wordpress-hard.conf and *wordpress-soft.conf*

There are some things that are almost always malicious, e.g. blocked users and pingbacks with errors. *wordpress-hard.conf* is designed to catch these so that you can ban the IP immediately.

Other things are relatively benign, like a failed login. You can't let people try forever, but banning the IP immediately would be wrong too. *wordpress-soft.conf* is designed to catch these so that you can set a higher retry limit before banning the IP.

For the avoidance of doubt: you should be using *both* filters.

wordpress-extra.conf

Version 4 introduced a number of new logging options which didn't fit cleanly into either of the *hard* or *soft* filters - they're *extra*.

For example, if your site doesn't use WordPress comments at all, you could add the rules matching attempted comments to the *hard* filter. Again, there is no "one size fits all" for these rules.

4.4 *mu-plugins* Support

There are two main reasons for using *mu-plugins*:

1. You need to load *WPf2b* before other security plugins[1],
2. You don't trust the site administrators.

4.4.1 Loading Early

One of the better ways is to install *WPf2b* as usual and then create a symlink in `mu-plugins`:

```
# ln -s ../plugins/wp-fail2ban/wp-fail2ban.php
# ls -l
total 1
lrwxr-xr-x  1  www  www  38  4 Nov 16:24 wp-fail2ban.php -> ../plugins/wp-fail2ban/wp-
↪fail2ban.php
```

or for the Premium version:

```
# ln -s ../plugins/wp-fail2ban-premium/wp-fail2ban.php
# ls -l
total 1
lrwxr-xr-x  1  www  www  38  4 Nov 16:24 wp-fail2ban.php -> ../plugins/wp-fail2ban-
↪premium/wp-fail2ban.php
```

This has the advantage that you can update *WPf2b* as usual without having to update `mu-plugins` directly. For the free version you don't need to activate *WPf2b*, but you do for the Premium version.

4.4.2 Forcing Usage

The main objective here is to stop people fiddling with things, so there are necessarily some restrictions on configuring *WPf2b*.

WPf2b must be configured in `wp-config.php` - you can't use the Premium config UI; not only does it make no sense, it won't work[2].

The actual configuration itself is simple; for the **Free** version:

1. Extract the **Free** version of *WPf2b* into a directory called *wp-fail2ban* within *mu-plugins*.
2. symlink `wp-fail2ban.php`:

```
# ln -s wp-fail2ban/wp-fail2ban.php
# ls -l
total 1
lrwxr-xr-x  1  www  www  38  4 Nov 16:24 wp-fail2ban.php -> wp-fail2ban/wp-fail2ban.
↪php
```

3. **Keep** *WPf2b* **up-to-date**.

For the **Premium** version:

1. Extract the **Premium** version of *WPf2b* into a directory called *wp-fail2ban-premium* within *mu-plugins*.
2. symlink `wp-fail2ban.php`:

[1] For example, WordFence, which assumes it's the only one.
[2] It may look like it works now, but in a future release it will be blocked.

```
# ln -s wp-fail2ban-premium/wp-fail2ban.php
# ls -l
total 1
lrwxr-xr-x  1  www  www  38  4 Nov 16:24 wp-fail2ban.php -> wp-fail2ban-premium/wp-
↪fail2ban.php
```

3. **Keep *WPf2b* up-to-date**.

Keeping *WPf2b* up-to-date

It's that last step that catches out most people - WordPress doesn't check `mu-plugins` for updates, so by configuring *WPf2b* in this way **you are taking responsibility** for keeping *WPf2b* up-to-date. I do my best, but I cannot guarantee there will never be a critical problem with *WPf2b* - you and you alone are responsible for checking for updates and installing them.

4.5 REST API

The REST API is a Premium-only feature.

4.5.1 Ensure the PHP *hash* Extension is Enabled

The `hash` extension is included by default in PHP 5.3 and later, but it must also be enabled. The *WPf2b* REST API will not load or run without the `hash` extension.

4.5.2 Define the Shared Secret

The Shared Secret must be defined in *wp-config.php* - it cannot be set via the UI - and it must be at least 64 characters long.

```
define('WP_FAIL2BAN_REST_SECRET', 'MINIMUM-OF-64-RANDOM-CHARACTERS');
```

An easy way to get a suitable Secret is to use one of the salts generated by the WordPress API. For example:

```
define('AUTH_KEY',          ' )h%mo9/sa3v<kc$aLuM,R~mj/fwjtCEv2*{Kva<jmq-V<OC}
↪ A6T(O*<*B-+$ka');
define('SECURE_AUTH_KEY',   'I0-b/MK/sLnsLXBhi>*~2sEXapCoWC6T;?IhjTn38lz~
↪LJ(S29BoiSjlf6~yM=?H');
define('LOGGED_IN_KEY',     'yr|(+vLu+PyyTvZ|r (h_IO!bOX,nU-?+Z&=hVB>ekQT~ t
↪$BCU`$65b<DdM5cmm');
define('NONCE_KEY',         'GFkF{@sIoPp<b,<mUyu5,i)hs/4hF0Al axhi:7KWmcgl||Z
↪{Fi]z@&qlZFv1Mq+');
define('AUTH_SALT',         '|+(C<`Qjb4<dPpbE0I-i59+PD_*Ch^<G{7EuAP_40WGi;&5␣
↪v:>gYFQ:+=S-zm`P');
define('SECURE_AUTH_SALT', '-%6q4fpO8~vlmiW`Ge|Ia!UGK{oB?p;RIX7h
↪%IDEVCoRuv9awsujP nz5@&YrH8B');
define('LOGGED_IN_SALT',    'N&w}80wPp3[p}=>reU;+&|G.*Rn!(g.z=UG5,68^tE}03
↪{3gRYWR^m/Mg-Fu?G<W');
define('NONCE_SALT',        'x%x3Y=}Gkc8YgEEeZsE_mAnE0>MCJFvT<=cl&W=2o=U5o1J+␣
↪BF-YTZT4Xau!X{B');
```

Warning:

1. The Secret **must be random data**.
2. The Secret **must be at least 64 characters long**.
3. The Secret **must be unique**.
4. **Do not use these example salts**[1].

You will compromise the security of your site if you fail to follow these rules.

4.5.3 Enable the *WPf2b* REST API

Either:

1.

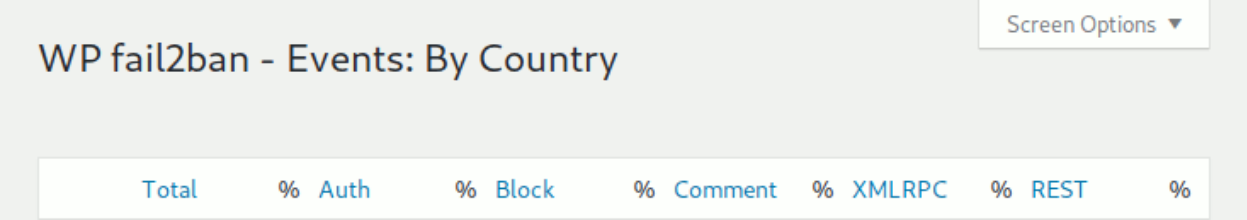
[1] You can try, but the REST API won't load if you do.

CHAPTER 5

Usage

5.1 Event Log

5.2 Report: Events by Country



CHAPTER 6

Add-ons

Gravity Forms

CHAPTER 7

Developers

New in version 4.2.

Version 4.2 introduced the ability for 3rd-party plugins to integrate with *WPf2b*.

7.1 Overview

The basic steps are:

7.1.1 Register Plugin

Description

Usage

```
try {
    do_action('wp_fail2ban_register_plugin', 'my-plugin-slug', 'My Plugin Name');
} catch(\LengthException $e) {
    // slug or name too long
} catch(\RuntimeException $e) {
    // database error
}
```

Parameters

wp_fail2ban_register_plugin WPf2b action.

my-plugin-slug The plugin slug to register. Must be < 256 chars.

My Plugin Name The display name of the plugin being registered. Must be < 256 chars.

Exceptions

LengthException Either the `slug` or `name` is too long.

7.1.2 Register Message

Example

```php
$args = [
    'slug'        => 'my-plugin-msg-slug-1',
    'fail'        => 'hard',
    'priority'    => LOG_NOTICE,
    'event_class' => 'Password',
    'event_id'    => 0x001F,
    'message'     => 'My message with ___VAR1___ and ___VAR2___',
    'vars'        => [
        'VAR1' => '\d+',
        'VAR2' => '*.'
    ]
];
try {
    do_action('wp_fail2ban_register_message', 'my-plugin-slug', $args);
} catch(\InvalidArgumentException $e) {
    // Missing entry or invalid type
} catch(\UnexpectedValueException $e) {
    // Invalid value
}
```

Details

do_action

wp_fail2ban_register_message *WPf2b* action.

my-plugin-slug The plugin slug used in *Register Plugin*.

$args

slug Message slug.

fail Recommended action.

priority *syslog* priority to use. Only the following priorities are supported:

- LOG_CRIT
- LOG_ERR
- LOG_WARNING
- LOG_NOTICE
- LOG_INFO
- LOG_DEBUG

event_class Class of Event. This is one of:

Auth Authentication-related Events. Note that Blocking Events will have their own class in the future.

Comment Comment-related Events.

XMLRPC XML-RPC-related Events.

Password Password-releated Events.

REST REST API-related Events.

Spam Spam-related Events.

event_id Event ID - 16 bits you can do with as you please.

message Message with substitutions. Note that " from *<IP>*" is appended.

vars An array of substitutions mapped to regular expressions.

When logging a message the substitutions are checked and substituted if present. The regex will be used to generate a matching rule for *fail2ban*.

7.1.3 Log Message

7.1.4 Design

To allow 3rd-party plugins to add support for *WPf2b* more easily, the API uses actions. This avoids the need to check if *WPf2b* is installed, then import a file, check for versions, and so on. Integration code can be written that will work if *WPf2b* is installed, and do nothing if not.

Note: Because `do_action` has no return value *WPf2b* will throw an Exception if there is an error.

7.2 Example

```php
/**
 *
 */
function myplugin_wpf2b_register()
{
    // Register the plugin
    try {
        do_action('wp_fail2ban_register_plugin', 'my-plugin-slug', 'My Plugin Name');
    } catch(\LengthException $e) {
        // slug or name too long
    } catch(\RuntimeException $e) {
        // database error
    }

    // Register a message
    $args = [
        'slug'        => 'my-plugin-msg-slug-1',
        'fail'        => 'hard',
        'priority'    => LOG_NOTICE,
        'event_class' => 'Password',
        'event_id'    => 0x001F,
        'message'     => 'My message with ___VAR1___ and ___VAR2___',
        'vars'        => [
            'VAR1' => '\d+',
```

(continues on next page)

(continued from previous page)

```
            'VAR2' => '*.'
        ]
    ];
    try {
        do_action('wp_fail2ban_register_message', 'my-plugin-slug', $args);
    } catch(\InvalidArgumentException $e) {
        // Missing entry or invalid type
    } catch(\UnexpectedValueException $e) {
        // Invalid value
    }
}
add_action('wp_fail2ban_register', __NAMESPACE__.'\myplugin_wpf2b_register');

/**
 *
 */
function myplugin_foobar()
{
    $vars = [
        'VAR1' => 12345,
        'VAR2' => 'xyz'
    ];
    do_action('my-plugin-slug', 'my-plugin-msg-slug-1', $vars);
}
```

CHAPTER 8

Release Notes

8.1 4.3.0

- Add new dashboard widget: last 5 *syslog* messages.
- Add full multisite support.
- Add username login blocking (force login with email).
- Add separate logging for login attempts with an empty username.
- Improve user enumeration blocking compatibility with the WordPress block editor (Gutenberg).
- Bump the minimum PHP version to 5.6.

8.1.1 Patches

4.3.0.1

Premium Only

- Fix issue when *WP_FAIL2BAN_BLOCK_USERNAME_LOGIN* enabled and *WP_FAIL2BAN_BLOCKED_USERS* not configured.

4.3.0.2

Premium Only

- Fix issue where some events weren't logged.

4.3.0.3

Premium Only

- Fix incorrect total for Event Log.
- Fix database renumber for Pingbacks.

4.3.0.4

- Fix plugin event registration.
- Add colour to "Last 5 Messages" dashboard widget.

4.3.0.5

- Fix empty username detection for multisite.
- Fix harmless warning when activating new multisite install.
- Fix esoteric edge-case where *wp-load.php* is loaded via a script run from the CLI in a directory with a *functions.php* file.

4.3.0.6

- Fix Forbidden error on Posts page for roles below Editor when user enum blocking enabled. [WordPress only]

8.1.2 Upgrade

To take advantage of the new features you will need up update your *fail2ban* filters; existing filters will continue to work as before. **Premium users:** Please backup your database before upgrading.

CHAPTER 9

define() Constants

9.1 All

9.1.1 WP_FAIL2BAN_AUTH_LOG

New in version 2.2.0.

By default, *WPf2b* uses **LOG_AUTH** for logging authentication success or failure. However, some systems use **LOG_AUTHPRIV** instead, but there's no good run-time way to tell. If your system uses **LOG_AUTHPRIV** you should add the following to `wp-config.php`:

```
define('WP_FAIL2BAN_AUTH_LOG', LOG_AUTHPRIV);
```

9.1.2 WP_FAIL2BAN_BLOCKED_USERS

New in version 2.0.0.

The bots that try to brute-force WordPress logins aren't that clever (no doubt that will change), but they may only make one request per IP every few hours in an attempt to avoid things like *fail2ban*. With large botnets this can still create significant load.

Based on a suggestion from *@jmadea*, *WPf2b* now allows you to specify a regex that will shortcut the login process if the requested username matches.

For example, putting the following in `wp-config.php`:

```
define('WP_FAIL2BAN_BLOCKED_USERS', '^admin$');
```

will block any attempt to log in as **admin** before most of the core WordPress code is run. Unless you go crazy with it, a regex is usually cheaper than a call to the database so this should help keep things running during an attack.

WPf2b doesn't do anything to the regex other than make it case-insensitive.

If you're running PHP 7, you can now specify an array of users instead:

```
define('WP_FAIL2BAN_BLOCKED_USERS', ['admin', 'another', 'user']);
```

9.1.3 WP_FAIL2BAN_BLOCK_USERNAME_LOGIN

New in version 4.3.0.

```
define('WP_FAIL2BAN_BLOCK_USERNAME_LOGIN', true);
```

9.1.4 WP_FAIL2BAN_BLOCK_USER_ENUMERATION

New in version 2.1.0.

Changed in version 4.0.0: Now also blocks enumeration via the REST API.

Brute-forcing WP requires knowing a valid username. Unfortunately, WP makes this all but trivial.

Based on a suggestion from *@geeklol* and a plugin by *@ROIBOT*, *WPf2b* can now block user enumeration attempts. Just add the following to `wp-config.php`:

```
define('WP_FAIL2BAN_BLOCK_USER_ENUMERATION', true);
```

9.1.5 WP_FAIL2BAN_COMMENT_EXTRA_LOG

New in version 4.0.5.

Default: LOG_AUTH

```
define('WP_FAIL2BAN_COMMENT_EXTRA_LOG', LOG_LOCAL5);
```

9.1.6 WP_FAIL2BAN_COMMENT_LOG

New in version 3.5.0.

By default, *WPf2b* uses **LOG_USER** for logging comments. If you'd rather it used a different facility you can change it by adding something like the following to `wp-config.php`:

```
define('WP_FAIL2BAN_COMMENT_LOG', LOG_LOCAL3);
```

See also:

- *WP_FAIL2BAN_LOG_COMMENTS*
- *WP_FAIL2BAN_LOG_COMMENTS_EXTRA*

9.1.7 WP_FAIL2BAN_DISABLE_LAST_LOG

New in version 4.3.0.

WPf2b v4.3.0 introduced a new dashboard widget to display the last 5 `syslog` messages.

These messages are stored in the options table; for most sites this won't be an issue, but, if you're already doing a lot of updates to the options table or have some other esoteric configuration, you might want to disable this feature:

```
define('WP_FAIL2BAN_DISABLE_LAST_LOG', true);
```

9.1.8 WP_FAIL2BAN_EX_LOG_HEADERS

Premium Only

New in version 4.3.0.

Lorem

```
define('WP_FAIL2BAN_EX_LOG_HEADERS', true);
```

9.1.9 WP_FAIL2BAN_EX_LOG_POST_DATA

Premium Only

New in version 4.3.0.

Lorem

```
define('WP_FAIL2BAN_EX_LOG_POST_DATA', true);
```

9.1.10 WP_FAIL2BAN_EX_LOG_REFERER

Premium Only

New in version 4.3.0.

Lorem

```
define('WP_FAIL2BAN_EX_LOG_REFERER', true);
```

9.1.11 WP_FAIL2BAN_EX_LOG_URL

Premium Only

New in version 4.3.0.

Lorem

```
define('WP_FAIL2BAN_EX_LOG_URL', true);
```

9.1.12 WP_FAIL2BAN_EX_LOG_USER_AGENT

Premium Only

New in version 4.3.0.

Lorem

```
define('WP_FAIL2BAN_EX_LOG_USER_AGENT', true);
```

9.1.13 WP_FAIL2BAN_EX_MAXMIND_LICENSE

Premium Only

New in version 4.3.0.

Lorem

```
define('WP_FAIL2BAN_EX_MAXMIND_LICENSE', true);
```

9.1.14 WP_FAIL2BAN_HTTP_HOST

New in version 3.0.0.

This is for some flavours of Linux where *WP_FAIL2BAN_SYSLOG_SHORT_TAG* isn't enough.

If you configure your web server to set an environment variable named **WP_FAIL2BAN_SYSLOG_SHORT_TAG** on a per-virtual host basis, *WPf2b* will use that in the syslog tag. This allows you to configure a unique tag per site in a way that makes sense for your configuration, rather than some arbitrary truncation or hashing within the plugin.

Note: This feature has not been tested as extensively as others. While I'm confident it works, FreeBSD doesn't have this problem so this feature will always be second-tier.

9.1.15 WP_FAIL2BAN_LOG_COMMENTS

New in version 3.5.0.

WPf2b can now log comments. To enable this feature, add the following to `wp-config.php`:

```
define('WP_FAIL2BAN_LOG_COMMENTS', true);
```

The comment ID and IP will be written to *WP_FAIL2BAN_COMMENT_LOG* and matched by *wordpress-extra.conf*.

See also:

- *WP_FAIL2BAN_COMMENT_LOG*

9.1.16 WP_FAIL2BAN_LOG_COMMENTS_EXTRA

New in version 4.0.0.

WPf2b can optionally log the following comment-related events:

Not found Attempted comment on a non-existent post

WPF2B_EVENT_COMMENT_NOT_FOUND

Closed Attempted comment on a post with closed comments

WPF2B_EVENT_COMMENT_CLOSED

Trash Attempted comment on a post in Trash

WPF2B_EVENT_COMMENT_TRASH

Draft Attempted comment on a Draft post

WPF2B_EVENT_COMMENT_DRAFT

Password-protected Attempted comment on a password-protected post

WPF2B_EVENT_COMMENT_PASSWORD

To enable this feature OR the event constants; for example, to enable *Closed* and *Draft*:

```
define('WP_FAIL2BAN_LOG_COMMENTS_EXTRA', WPF2B_EVENT_COMMENT_CLOSED | WPF2B_EVENT_
↪COMMENT_DRAFT);
```

The Post ID and IP will be written to *WP_FAIL2BAN_COMMENT_LOG* and matched by *wordpress-extra.conf*.

9.1.17 WP_FAIL2BAN_LOG_DISABLE_LAST

New in version 4.3.0.

Lorem

```
define('WP_FAIL2BAN_LOG_DISABLE_LAST', true);
```

9.1.18 WP_FAIL2BAN_LOG_PASSWORD_REQUEST

New in version 3.5.0.

WPf2b can log password reset requests. Add the following to `wp-config.php`:

```
define('WP_FAIL2BAN_LOG_PASSWORD_REQUEST', true);
```

The username and IP will be written to *WP_FAIL2BAN_PASSWORD_REQUEST_LOG* and matched by *wordpress-extra.conf*.

9.1.19 WP_FAIL2BAN_LOG_PINGBACKS

New in version 2.2.0.

Based on a suggestion from *@maghe*, *WPf2b* can now log pingbacks. To enable this feature, add the following to `wp-config.php`:

```
define('WP_FAIL2BAN_LOG_PINGBACKS', true);
```

By default, *WPf2b* uses **LOG_USER** for logging pingbacks. If you'd rather it used a different facility you can change it by adding something like the following to `wp-config.php`:

```
define('WP_FAIL2BAN_PINGBACK_LOG', LOG_LOCAL3);
```

9.1.20 WP_FAIL2BAN_LOG_SPAM

New in version 3.5.0.

WPf2b can now log spam comments. To enable this feature, add the following to `wp-config.php`:

```
define('WP_FAIL2BAN_LOG_SPAM', true);
```

The comment ID and IP will be written to *WP_FAIL2BAN_SPAM_LOG* and matched by *wordpress-hard.conf*.

See also:

- *WP_FAIL2BAN_SPAM_LOG*

9.1.21 WP_FAIL2BAN_OPENLOG_OPTIONS

New in version 3.5.0.

9.1.22 WP_FAIL2BAN_PASSWORD_REQUEST_LOG

New in version 4.0.0.

9.1.23 WP_FAIL2BAN_PINGBACK_ERROR_LOG

New in version 4.0.5: Reserved for future use.

Default: LOG_AUTH

```
define('WP_FAIL2BAN_PINGBACK_ERROR_LOG', LOG_LOCAL3);
```

9.1.24 WP_FAIL2BAN_PINGBACK_LOG

New in version 2.2.0.

See *WP_FAIL2BAN_LOG_PINGBACKS*.

9.1.25 WP_FAIL2BAN_PLUGIN_AUTH_LOG

New in version 4.2.0.

Facility for “Auth” class plugin messages.

See also:

- *WP_FAIL2BAN_PLUGIN_LOG_AUTH*
- *Facilities*

9.1.26 WP_FAIL2BAN_PLUGIN_COMMENT_LOG

New in version 4.2.0.

Facility for "Comment" class plugin messages.

See also:

- *WP_FAIL2BAN_PLUGIN_LOG_COMMENT*
- *Facilities*

9.1.27 WP_FAIL2BAN_PLUGIN_LOG_AUTH

New in version 4.2.0.

To enable logging plugin "Auth" class messages, add the following to `wp-config.php`:

```
define('WP_FAIL2BAN_PLUGIN_LOG_AUTH', true);
```

See also:

- *WP_FAIL2BAN_PLUGIN_AUTH_LOG*

9.1.28 WP_FAIL2BAN_PLUGIN_LOG_COMMENT

New in version 4.2.0.

To enable logging plugin "Comment" class messages, add the following to `wp-config.php`:

```
define('WP_FAIL2BAN_PLUGIN_LOG_COMMENT', true);
```

See also:

- *WP_FAIL2BAN_PLUGIN_COMMENT_LOG*

9.1.29 WP_FAIL2BAN_PLUGIN_LOG_OTHER

New in version 4.2.0.

To enable logging plugin "Other" class messages, add the following to `wp-config.php`:

```
define('WP_FAIL2BAN_PLUGIN_LOG_OTHER', true);
```

See also:

- *WP_FAIL2BAN_PLUGIN_OTHER_LOG*

9.1.30 WP_FAIL2BAN_PLUGIN_LOG_PASSWORD

New in version 4.2.0.

To enable logging plugin "Password" class messages, add the following to `wp-config.php`:

```
define('WP_FAIL2BAN_PLUGIN_LOG_PASSWORD', true);
```

See also:

- *WP_FAIL2BAN_PLUGIN_PASSWORD_LOG*

9.1.31 WP_FAIL2BAN_PLUGIN_LOG_REST

New in version 4.2.0.

To enable logging plugin "REST" class messages, add the following to `wp-config.php`:

```
define('WP_FAIL2BAN_PLUGIN_LOG_REST', true);
```

See also:

- *WP_FAIL2BAN_PLUGIN_REST_LOG*

9.1.32 WP_FAIL2BAN_PLUGIN_LOG_SPAM

New in version 4.2.0.

To enable logging plugin "Spam" class messages, add the following to `wp-config.php`:

```
define('WP_FAIL2BAN_PLUGIN_LOG_SPAM', true);
```

See also:

- *WP_FAIL2BAN_PLUGIN_SPAM_LOG*

9.1.33 WP_FAIL2BAN_PLUGIN_LOG_XMLRPC

New in version 4.2.0.

To enable logging plugin "XMLRPC" class messages, add the following to `wp-config.php`:

```
define('WP_FAIL2BAN_PLUGIN_LOG_XMLRPC', true);
```

See also:

- *WP_FAIL2BAN_PLUGIN_XMLRPC_LOG*

9.1.34 WP_FAIL2BAN_PLUGIN_OTHER_LOG

New in version 4.2.0.

Facility for "Other" class plugin messages.

See also:

- *WP_FAIL2BAN_PLUGIN_LOG_OTHER*
- *Facilities*

9.1.35 WP_FAIL2BAN_PLUGIN_PASSWORD_LOG

New in version 4.2.0.

Facility for "Password" class plugin messages.

See also:

- *WP_FAIL2BAN_PLUGIN_LOG_PASSWORD*
- *Facilities*

9.1.36 WP_FAIL2BAN_PLUGIN_REST_LOG

New in version 4.2.0.

Facility for "REST" class plugin messages.

See also:

- *WP_FAIL2BAN_PLUGIN_LOG_REST*
- *Facilities*

9.1.37 WP_FAIL2BAN_PLUGIN_SPAM_LOG

New in version 4.2.0.

Facility for "Spam" class plugin messages.

See also:

- *WP_FAIL2BAN_PLUGIN_LOG_SPAM*
- *Facilities*

9.1.38 WP_FAIL2BAN_PLUGIN_XMLRPC_LOG

New in version 4.2.0.

Facility for "XML-RPC" class plugin messages.

See also:

- *WP_FAIL2BAN_PLUGIN_LOG_XMLRPC*
- *Facilities*

9.1.39 WP_FAIL2BAN_PROXIES

New in version 2.0.0.

Changed in version 4.0.0: Entries can be ignored by prefixing with **#**

The idea here is to list the IP addresses of the trusted proxies that will appear as the remote IP for the request. When defined:

- If the remote address appears in the **WP_FAIL2BAN_PROXIES** list, *WPf2b* will log the IP address from the *X-Forwarded-For* header
- If the remote address does not appear in the **WP_FAIL2BAN_PROXIES** list, *WPf2b* will return a 403 error
- If there's no *X-Forwarded-For* header, *WPf2b* will behave as if **WP_FAIL2BAN_PROXIES** isn't defined

To set **WP_FAIL2BAN_PROXIES**, add something like the following to `wp-config.php`:

```
define('WP_FAIL2BAN_PROXIES','192.168.0.42,192.168.42.0/24');
```

WPf2b doesn't do anything clever with the list - beware of typos!

If you're running PHP 7 you can use an array instead:

```
define('WP_FAIL2BAN_PROXIES', [
    '192.168.0.42',
    '192.168.42.0/24'
]);
```

9.1.40 WP_FAIL2BAN_REMOTE_ADDR

New in version 3.6.0.

Some themes and plugins anonymise requests

9.1.41 WP_FAIL2BAN_REST_API

Premium feature.

New in version 4.3.0.

WPf2b now has a RESTful API for remote configuration and monitoring. To enable this feature, add the following to `wp-config.php`:

```
define('WP_FAIL2BAN_REST_API', true);
```

You must also define *WP_FAIL2BAN_REST_SECRET*.

See also:

- *WP_FAIL2BAN_REST_SECRET*

9.1.42 WP_FAIL2BAN_REST_SECRET

Premium feature.

New in version 4.3.0.

WPf2b now has a RESTful API for remote configuration and monitoring. To enable this feature, add the following to `wp-config.php`:

```
define('WP_FAIL2BAN_REST_SECRET', 'N&w}80wPp3[p}=>reU;+&|G.*Rn!(g.z=UG5,68^tE}03
↪{3gRYWR^m/Mg-Fu?G<W');
```

Warning:

1. The Secret **must be random data**.
2. The Secret **must be at least 64 characters long**.
3. The Secret **must be unique**.

You will compromise the security of your site if you fail to follow these rules.

See also:

- *WP_FAIL2BAN_REST_API*

9.1.43 WP_FAIL2BAN_SPAM_LOG

New in version 4.0.0.

9.1.44 WP_FAIL2BAN_SYSLOG_SHORT_TAG

New in version 3.0.0.

Some flavours of Linux come with a *syslogd* that can't cope with the normal message format *WPf2b* uses; basically, they assume that the first part of the message (the tag) won't exceed some (small) number of characters, and mangle the message if it does. This breaks the regex in the *fail2ban* filter and so nothing gets blocked.

Adding:

```
define('WP_FAIL2BAN_SYSLOG_SHORT_TAG', true);
```

to `functions.php` will make *WPf2b* use `wp` as the syslog tag, rather than the normal `wordpress`. This buys you 7 characters which may be enough to work around the problem, but if it's not enough you should look at *WP_FAIL2BAN_HTTP_HOST* or *WP_FAIL2BAN_TRUNCATE_HOST* too.

9.1.45 WP_FAIL2BAN_TRUNCATE_HOST

New in version 3.5.0.

If you've set *WP_FAIL2BAN_SYSLOG_SHORT_TAG* and defining *WP_FAIL2BAN_HTTP_HOST* for each virtual host isn't appropriate, you can set **WP_FAIL2BAN_TRUNCATE_HOST** to whatever value you need to make *syslog* happy:

```
define('WP_FAIL2BAN_TRUNCATE_HOST', 8);
```

This does exactly what the name suggests: truncates the host name to the length you specify. As a result there's no guarantee that what's left will be enough to identify the site.

9.1.46 WP_FAIL2BAN_XMLRPC_LOG

New in version 3.6.0.

This is for debugging and future development.

Attackers are doing weird things with XML-RPC, so this logs the raw post data to the file specified:

```
define('WP_FAIL2BAN_XMLRPC_LOG', '/var/log/xml-rpc.log');
```

9.2 Development

9.3 Logging

9.4 Plugins

9.5 Remote IPs

9.6 Reserved

9.7 syslog

9.8 Users

CHAPTER 10

Facilities

While the full list of facilities is reproduced here for completeness, using anything but **LOG_AUTH**, **LOG_AUTHPRIV**, and/or **LOG_LOCAL0**..**7** is unlikely to have the desired results.

Facility	Description
LOG_AUTH	security/authorization messages (use LOG_AUTHPRIV instead in systems where that constant is defined)
LOG_AUTHPRIV	security/authorization messages (private)
LOG_CRON	clock daemon (cron and at)
LOG_DAEMON	other system daemons
LOG_KERN	kernel messages
LOG_LOCAL0…7	reserved for local use, these are not available in Windows
LOG_LPR	line printer subsystem
LOG_MAIL	mail subsystem
LOG_NEWS	USENET news subsystem
LOG_SYSLOG	messages generated internally by syslogd
LOG_USER	generic user-level messages
LOG_UUCP	UUCP subsystem

CHAPTER 11

Logfiles

OS	Level	LOG_AUTH	LOG_AUTHPRIV	LOG_USER
CentOS 7			`/var/log/secure`	
FeeBSD	INFO	`/var/log/auth/log`	`/var/log/auth/log`	•
	NOTICE	`/var/log/auth/log`	`/var/log/auth/log`	`/var/log/messages`
Ubuntu 18	(all)	`/var/log/auth.log`	`/var/log/auth.log`	`/var/log/syslog`

CHAPTER 12

Filter Files

12.1 wordpress-hard.conf

```
# Fail2Ban filter for WordPress hard failures
# Auto-generated: 2019-04-18T14:45:30+00:00
#

[INCLUDES]

before = common.conf

[Definition]

_daemon = (?:wordpress|wp)

failregex = ^%(__prefix_line)sAuthentication attempt for unknown user .* from <HOST>$
            ^%(__prefix_line)sREST authentication attempt for unknown user .* from
↪<HOST>$
            ^%(__prefix_line)sXML-RPC authentication attempt for unknown user .* from
↪<HOST>$
            ^%(__prefix_line)sSpam comment \d+ from <HOST>$
            ^%(__prefix_line)sBlocked user enumeration attempt from <HOST>$
            ^%(__prefix_line)sBlocked authentication attempt for .* from <HOST>$
            ^%(__prefix_line)sXML-RPC multicall authentication failure from <HOST>$
            ^%(__prefix_line)sPingback error .* generated from <HOST>$

ignoreregex =

# DEV Notes:
# Requires the 'WP fail2ban' plugin:
# https://wp-fail2ban.com/
#
# Author: Charles Lecklider
```

12.2 wordpress-soft.conf

```
# Fail2Ban filter for WordPress soft failures
# Auto-generated: 2019-04-18T14:45:30+00:00
#

[INCLUDES]

before = common.conf

[Definition]

_daemon = (?:wordpress|wp)

failregex = ^%(__prefix_line)sAuthentication failure for .* from <HOST>$
            ^%(__prefix_line)sREST authentication failure for .* from <HOST>$
            ^%(__prefix_line)sXML-RPC authentication failure for .* from <HOST>$

ignoreregex =

# DEV Notes:
# Requires the 'WP fail2ban' plugin:
# https://wp-fail2ban.com/
#
# Author: Charles Lecklider
```

12.3 wordpress-extra.conf

```
# Fail2Ban filter for WordPress extra failures
# Auto-generated: 2019-04-18T14:45:30+00:00
#

[INCLUDES]

before = common.conf

[Definition]

_daemon = (?:wordpress|wp)

failregex = ^%(__prefix_line)sComment \d+ from <HOST>$
            ^%(__prefix_line)sComment post not found \d+ from <HOST>$
            ^%(__prefix_line)sComments closed on post \d+ from <HOST>$
            ^%(__prefix_line)sComment attempt on trash post \d+ from <HOST>$
            ^%(__prefix_line)sComment attempt on draft post \d+ from <HOST>$
            ^%(__prefix_line)sComment attempt on password-protected post \d+ from
↪<HOST>$
            ^%(__prefix_line)sPassword reset requested for .* from <HOST>$

ignoreregex =

# DEV Notes:
# Requires the 'WP fail2ban' plugin:
# https://wp-fail2ban.com/
```

(continues on next page)

(continued from previous page)

```
#
# Author: Charles Lecklider
```