
WP fail2ban Manual

Charles Lecklider

May 19, 2020

Contents

1	Introduction	3
1.1	History	3
1.1.1	LTS Version	3
1.2	Future	4
1.3	Features	4
1.3.1	CloudFlare and Proxy Servers	4
1.3.2	Comments	4
1.3.3	Pingbacks	4
1.3.4	Spam	4
1.3.5	User Enumeration	4
1.3.6	Work-Arounds for Broken syslogd	4
1.3.7	Blocking Users	4
1.3.8	<i>mu-plugins</i> Support	4
2	Installation	5
2.1	Is <i>WP fail2ban</i> Already Installed?	5
2.2	Overview	5
2.2.1	Premium	6
3	Configuration	7
3.1	WP fail2ban	7
3.2	Logging	7
3.2.1	Choosing the Events to Log	7
3.2.2	Advanced Users	7
3.3	fail2ban	8
3.3.1	Filters	9
3.4	<i>mu-plugins</i> Support	10
3.4.1	Loading Early	10
3.4.2	Forcing Usage	10
4	Usage	13
4.1	Event Log	13
4.2	Report: Events by Country	13
5	Add-ons	15
6	Developers	17

6.1	Overview	17
6.1.1	Register Plugin	17
6.1.2	Register Message	18
6.1.3	Log Message	19
6.1.4	Design	19
6.2	Example	19
7	Release Notes	21
7.1	4.2.0	21
7.2	4.2.1	21
7.2.1	Upgrade	21
7.3	4.2.2	21
7.3.1	Upgrade	22
7.4	4.2.3	22
7.4.1	Upgrade	22
7.5	4.2.4	22
7.5.1	Upgrade	22
7.6	4.2.5	22
7.6.1	Upgrade	22
7.7	4.2.5.1	22
7.7.1	Upgrade	23
7.8	4.2.6	23
7.8.1	Upgrade	23
7.9	4.2.7	23
7.9.1	Upgrade	23
7.10	4.2.7.1	23
7.10.1	Upgrade	23
7.11	4.2.8	23
7.11.1	Upgrade	24
8	define() Constants	25
8.1	All	25
8.1.1	WP_FAIL2BAN_AUTH_LOG	25
8.1.2	WP_FAIL2BAN_BLOCKED_USERS	25
8.1.3	WP_FAIL2BAN_BLOCK_USER_ENUMERATION	26
8.1.4	WP_FAIL2BAN_COMMENT_EXTRA_LOG	26
8.1.5	WP_FAIL2BAN_COMMENT_LOG	26
8.1.6	WP_FAIL2BAN_HTTP_HOST	26
8.1.7	WP_FAIL2BAN_LOG_COMMENTS	27
8.1.8	WP_FAIL2BAN_LOG_COMMENTS_EXTRA	27
8.1.9	WP_FAIL2BAN_LOG_PASSWORD_REQUEST	28
8.1.10	WP_FAIL2BAN_LOG_PINGBACKS	28
8.1.11	WP_FAIL2BAN_LOG_SPAM	28
8.1.12	WP_FAIL2BAN_OPENLOG_OPTIONS	28
8.1.13	WP_FAIL2BAN_PASSWORD_REQUEST_LOG	28
8.1.14	WP_FAIL2BAN_PINGBACK_ERROR_LOG	28
8.1.15	WP_FAIL2BAN_PINGBACK_LOG	29
8.1.16	WP_FAIL2BAN_PLUGIN_AUTH_LOG	29
8.1.17	WP_FAIL2BAN_PLUGIN_COMMENT_LOG	29
8.1.18	WP_FAIL2BAN_PLUGIN_LOG_AUTH	29
8.1.19	WP_FAIL2BAN_PLUGIN_LOG_COMMENT	29
8.1.20	WP_FAIL2BAN_PLUGIN_LOG_OTHER	30
8.1.21	WP_FAIL2BAN_PLUGIN_LOG_PASSWORD	30
8.1.22	WP_FAIL2BAN_PLUGIN_LOG_REST	30

8.1.23	WP_FAIL2BAN_PLUGIN_LOG_SPAM	30
8.1.24	WP_FAIL2BAN_PLUGIN_LOG_XMLRPC	30
8.1.25	WP_FAIL2BAN_PLUGIN_OTHER_LOG	31
8.1.26	WP_FAIL2BAN_PLUGIN_PASSWORD_LOG	31
8.1.27	WP_FAIL2BAN_PLUGIN_REST_LOG	31
8.1.28	WP_FAIL2BAN_PLUGIN_SPAM_LOG	31
8.1.29	WP_FAIL2BAN_PLUGIN_XMLRPC_LOG	31
8.1.30	WP_FAIL2BAN_PROXIES	32
8.1.31	WP_FAIL2BAN_REMOTE_ADDR	32
8.1.32	WP_FAIL2BAN_SPAM_LOG	32
8.1.33	WP_FAIL2BAN_SYSLOG_SHORT_TAG	32
8.1.34	WP_FAIL2BAN_TRUNCATE_HOST	33
8.1.35	WP_FAIL2BAN_XMLRPC_LOG	33
8.2	Development	33
8.3	Logging	33
8.4	Plugins	33
8.5	Remote IPs	33
8.6	Reserved	33
8.7	syslog	33
8.8	Users	33
9	Facilities	35
10	Logfiles	37
11	Filter Files	39
11.1	wordpress-hard.conf	39
11.2	wordpress-soft.conf	40
11.3	wordpress-extra.conf	40

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

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 10¹ 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.

1.1.1 LTS³ Version

My plan was to retire the 3.x branch, but you know what they say about plans. . . .

It turns out that at least one large hosting provider pre-installs *WPf2b* in `mu-plugins`. It's always great to see your work being used widely, but it would have been even better if they had let me know - I could have tested that combination before releasing version 4.

As a result there will now be a 3.7. This will be the very last version 3 release, it will be tailored to being pre-installed in `mu-plugins`, and it will be supported indefinitely.

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.

³ Long-Term Support

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.

1.3 Features

1.3.1 CloudFlare and Proxy Servers

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

1.3.2 Comments

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

1.3.3 Pingbacks

WPf2b logs failed pingbacks, and can log all pingbacks. For a brief overview see [*WP_FAIL2BAN_LOG_PINGBACKS*](#).

1.3.4 Spam

WPf2b can log comments marked as spam. See [*WP_FAIL2BAN_LOG_SPAM*](#).

1.3.5 User Enumeration

WPf2b can block user enumeration. See [*WP_FAIL2BAN_BLOCK_USER_ENUMERATION*](#).

1.3.6 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*](#).

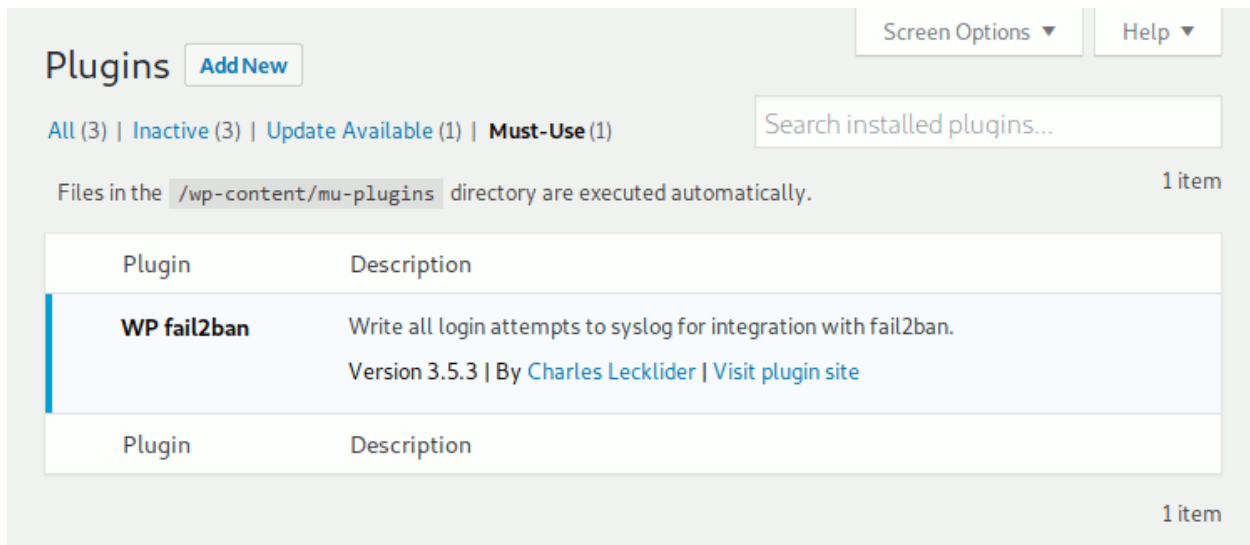
1.3.7 Blocking Users

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*](#).

1.3.8 *mu-plugins* Support

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

2.1 Is *WP fail2ban* Already Installed?



The screenshot shows the WordPress 'Plugins' management interface. At the top, there are buttons for 'Screen Options' and 'Help'. Below the 'Plugins' title is an 'Add New' button. A navigation bar shows 'All (3) | Inactive (3) | Update Available (1) | Must-Use (1)'. A search box contains the text 'Search installed plugins...'. A message states 'Files in the /wp-content/mu-plugins directory are executed automatically.' with a '1 item' count. A table lists installed plugins:

Plugin	Description
WP fail2ban	Write all login attempts to syslog for integration with fail2ban. Version 3.5.3 By Charles Lecklider Visit plugin site
Plugin	Description

At the bottom right of the table area, it says '1 item'.

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

2.2 Overview

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

2.2.1 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.

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

3.1 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.

3.2 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.

3.2.1 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.

3.2.2 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
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](#).

3.3 fail2ban

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

3.3.1 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.

3.4 *mu-plugins* Support

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

1. You need to load *WPf2b* before other security plugins¹,
2. You don't trust the site administrators.

3.4.1 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.

3.4.2 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 work².

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.

4.1 Event Log



4.2 Report: Events by Country



CHAPTER 5

Add-ons

Gravity Forms

New in version 4.2.

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

6.1 Overview

The basic steps are:

6.1.1 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.

6.1.2 Register Message

Example

```

1  $args = [
2      'slug'           => 'my-plugin-msg-slug-1',
3      'fail'          => 'hard',
4      'priority'      => LOG_NOTICE,
5      'event_class'   => 'Password',
6      'event_id'      => 0x001F,
7      'message'       => 'My message with ___VAR1___ and ___VAR2___',
8      'vars'          => [
9          'VAR1'       => '\d+',
10         'VAR2'       => '*.'
11     ]
12 ];
13 try {
14     do_action('wp_fail2ban_register_message', 'my-plugin-slug', $args);
15 } catch(\InvalidArgumentException $e) {
16     // Missing entry or invalid type
17 } catch(\UnexpectedValueException $e) {
18     // Invalid value
19 }

```

Details

do_action

wp_fail2ban_register_message *Wp2b* 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-related 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*.

6.1.3 Log Message

6.1.4 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.

6.2 Example

```

1  /**
2   *
3   */
4  function myplugin_wpf2b_register()
5  {
6     // Register the plugin
7     try {
8         do_action('wp_fail2ban_register_plugin', 'my-plugin-slug', 'My Plugin Name');
9     } catch(\LengthException $e) {
10        // slug or name too long
11    } catch(\RuntimeException $e) {
12        // database error
13    }
14
15    // Register a message
16    $args = [
17        'slug'      => 'my-plugin-msg-slug-1',
18        'fail'      => 'hard',
19        'priority'  => LOG_NOTICE,
20        'event_class' => 'Password',
21        'event_id'  => 0x001F,
22        'message'   => 'My message with ___VAR1___ and ___VAR2___',
23        'vars'      => [
24            'VAR1' => '\d+',

```

(continues on next page)

```
25     'VAR2' => '*.'
26   ]
27 ];
28 try {
29   do_action('wp_fail2ban_register_message', 'my-plugin-slug', $args);
30 } catch(\InvalidArgumentException $e) {
31   // Missing entry or invalid type
32 } catch(\UnexpectedValueException $e) {
33   // Invalid value
34 }
35 }
36 add_action('wp_fail2ban_register', __NAMESPACE__.'\myplugin_wpf2b_register');
37
38 /**
39  *
40  */
41 function myplugin_foobar()
42 {
43     $vars = [
44         'VAR1' => 12345,
45         'VAR2' => 'xyz'
46     ];
47     do_action('my-plugin-slug', 'my-plugin-msg-slug-1', $vars);
48 }
```

7.1 4.2.0

- Not released.

7.2 4.2.1

- Completed support for `WP_FAIL2BAN_COMMENT_EXTRA_LOG`.
- Add support for 3rd-party plugins; see *Developers*.
 - Add-on for Contact Form 7 (experimental).
 - Add-on for Gravity Forms (experimental).
- Change logging for known-user with incorrect password; previously logged as unknown user and matched by *hard* filters (due to limitations in older versions of WordPress), now logged as known user and matched by *soft*.
- Bugfix for email-as-username - now logged correctly and matched by soft, not hard, filters.
- Bugfix for regression in code to prevent Free/Premium conflict.

7.2.1 Upgrade

You do not need to update your filters from 4.1.0.

7.3 4.2.2

- Fix 5.3 compatibility.

7.3.1 Upgrade

You do not need to update your filters from 4.1.0.

7.4 4.2.3

- Workaround for some versions of PHP 7.x that would cause `define()` to be ignored.
- Add config note to settings tabs.
- Fix documentation links.

7.4.1 Upgrade

This is a bugfix release. You do not need to update your filters from 4.1.0.

7.5 4.2.4

- Add filter for login failed message.
- Fix logging spam comments from admin area.
- Fix Settings link from Plugins page.
- Update Freemius library

7.5.1 Upgrade

This is a minor release. You do not need to update your filters from 4.1.0.

7.6 4.2.5

- Properly fix PHP 5.3 support; tested on CentOS 6. Does not support any UI or Premium features.
- Fix potential issue with `WP_FAIL2BAN_BLOCK_USER_ENUMERATION` if calling REST API or XMLRPC from admin area.

7.6.1 Upgrade

This is a minor release. You do not need to update your filters from 4.1.0.

7.7 4.2.5.1

- Fix premium activation issue with PHP < 7.0.

7.7.1 Upgrade

This is a premium-only patch release. If you are on PHP 7.0 or later you do not need to upgrade.

7.8 4.2.6

- Add support for [Remote Tools](#) add-on.
- Add support for the new ClassicPress security page.
- Improved user enumeration blocking.

7.8.1 Upgrade

This is a minor release. You do not need to update your filters from 4.1.0.

7.9 4.2.7

- Fix error when blocking user enumeration via REST.
- Fix buttons on Settings tabs.

7.9.1 Upgrade

This is a bugfix release. You do not need to update your filters from 4.1.0.

7.10 4.2.7.1

- Fix error when blocking user enumeration via *oembed* (h/t @wordpressfab).

7.10.1 Upgrade

This is a bugfix release. You do not need to update your filters from 4.1.0.

7.11 4.2.8

- Add link to new [support forum](#).
- Fix user enumeration conflict with Gutenberg (h/t @dinghy).
- Fix notices wrt admin menu (h/t @marioivangf).
- Fix harmless XDebug notice (h/t @dinghy).
- Update Freemius library.

7.11.1 Upgrade

This is a bugfix release. You do not need to update your filters from 4.1.0. This is the *last* release that supports PHP 5.3.

define() Constants

8.1 All

8.1.1 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);
```

8.1.2 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']);
```

8.1.3 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);
```

8.1.4 WP_FAIL2BAN_COMMENT_EXTRA_LOG

New in version 4.0.5.

Default: LOG_AUTH

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

8.1.5 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](#)

8.1.6 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.

8.1.7 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`

8.1.8 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`.

8.1.9 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`.

8.1.10 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);
```

8.1.11 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`

8.1.12 WP_FAIL2BAN_OPENLOG_OPTIONS

New in version 3.5.0.

8.1.13 WP_FAIL2BAN_PASSWORD_REQUEST_LOG

New in version 4.0.0.

8.1.14 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);
```

8.1.15 WP_FAIL2BAN_PINGBACK_LOG

New in version 2.2.0.

See *WP_FAIL2BAN_LOG_PINGBACKS*.

8.1.16 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*

8.1.17 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*

8.1.18 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*

8.1.19 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*

8.1.20 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*

8.1.21 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*

8.1.22 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*

8.1.23 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*

8.1.24 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*

8.1.25 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*

8.1.26 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*

8.1.27 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*

8.1.28 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*

8.1.29 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*

8.1.30 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, *Wp2b* will log the IP address from the *X-Forwarded-For* header
- If the remote address does not appear in the **WP_FAIL2BAN_PROXIES** list, *Wp2b* will return a 403 error
- If there's no *X-Forwarded-For* header, *Wp2b* 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');
```

Wp2b doesn't do anything clever with the list - beware of typos!

8.1.31 WP_FAIL2BAN_REMOTE_ADDR

New in version 3.6.0.

Some themes and plugins anonymise requests

8.1.32 WP_FAIL2BAN_SPAM_LOG

New in version 4.0.0.

8.1.33 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 *Wp2b* 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 *Wp2b* 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.

8.1.34 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.

8.1.35 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');
```

8.2 Development

8.3 Logging

8.4 Plugins

8.5 Remote IPs

8.6 Reserved

8.7 syslog

8.8 Users

CHAPTER 9

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 10

Logfiles

OS	Level	LOG_AUTH	LOG_AUTHPRIV	LOG_USER
CentOS 7			/var/log/ secure	
FreeBSD	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

11.1 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
```

11.2 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
```

11.3 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
```