

Fail2Ban

Links & Installation

[Fail2ban Documentation](#)

[Linode Fail2ban Guide](#)

```
sudo yum install fail2ban
sudo apt install fail2ban
sudo pacman -Syu fail2ban
```

etc..

Configuration Files

`/etc/fail2ban/`

To modify configs, copy any `fail2ban.conf` to `fail2ban.local` and modify the copied `fail2ban.local` configuration file. Fail2ban will automatically override the settings in `fail2ban.conf` with those in `fail2ban.local`

```
sudo cp /etc/fail2ban/fail2ban.conf /etc/fail2ban/fail2ban.local
```

This file is also not intended to be modified directly. Run the command below to create a local configuration to edit -

```
sudo cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local
```

jail.local

This file holds all of your local fail2ban settings for this host. Some of the important lines and settings to look at are seen below

- `ignoreip`
 - An IP address for fail2ban to ignore
- `maxretry`
 - Number of retries before being locked out

#... File Reduced ...

Destination email address used solely for the interpolations in
jail.{conf,local,d/*} configuration files.

destemail = user@gmail.com

Sender email address used solely for some actions

sender = admin@hostname

E-mail action. Since 0.8.1 Fail2Ban uses sendmail MTA for the
mailing. Change mta configuration parameter to mail if you want to
revert to conventional 'mail'.

mta = mail

Default protocol

protocol = tcp

#... File Reduced ...

Choose default action. To change, just override value of 'action' with the
interpolation to the chosen action shortcut (e.g. action_mw, action_mwl, etc) in jail.local
globally (section [DEFAULT]) or per specific section

action = %(action_mwl)s

#

JAILS

#

#

SSH servers

#

[sshd]

To use more aggressive sshd modes set filter parameter "mode" in jail.local:
normal (default), ddos, extra or aggressive (combines all).
See "tests/files/logs/sshd" or "filter.d/sshd.conf" for usage example and[Definition]

failregex = ^<HOST> .*GET.*(\.php|\.asp|\.exe|\.pl|\.cgi|\.scgi)

ignoreregex = details.

```

#mode  = normal
enabled = true
port   = 22
logpath = %(sshd_log)s
backend = %(sshd_backend)s

#... File Reduced ...

[nginx-http-auth]

enabled = true
port    = http,https
logpath = %(nginx_error_log)s

# To use 'nginx-limit-req' jail you should have `ngx_http_limit_req_module`
# and define `limit_req` and `limit_req_zone` as described in nginx documentation
# http://nginx.org/en/docs/http/ngx_http_limit_req_module.html
# or for example see in 'config/filter.d/nginx-limit-req.conf'
[nginx-limit-req]
port    = http,https
logpath = %(nginx_error_log)s

[nginx-botsearch]

enabled = true
port    = http,https
logpath = %(nginx_error_log)s
maxretry = 2

#... File Reduced ...

```

its important to notice the line `action = %(action_mwl)s` - this line defines which default action we will take when a fail2ban is broken. These actions are defined within our `jail.local`, but Ill paste them here as well

```

# Action shortcuts. To be used to define action parameter

# Default banning action (e.g. iptables, iptables-new,
# iptables-multiport, shorewall, etc) It is used to define
# action_* variables. Can be overridden globally or per

```

```
# section within jail.local file
banaction = iptables-multiport
banaction_allports = iptables-allports

# The simplest action to take: ban only
action_ = %(banaction)s[name=%(__name__)s, bantime="%%(bantime)s", port="%%(port)s",
protocol="%%(protocol)s", chain="%%(chain)s"]

# ban & send an e-mail with whois report to the destemail.
action_mw = %(banaction)s[name=%(__name__)s, bantime="%%(bantime)s", port="%%(port)s",
protocol="%%(protocol)s", chain="%%(chain)s"]
    %(mta)s-whois[name=%(__name__)s, sender="%%(sender)s", dest="%%(destemail)s",
protocol="%%(protocol)s", chain="%%(chain)s"]

# ban & send an e-mail with whois report and relevant log lines
# to the destemail.
action_mwl = %(banaction)s[name=%(__name__)s, bantime="%%(bantime)s", port="%%(port)s",
protocol="%%(protocol)s", chain="%%(chain)s"]
    %(mta)s-whois-lines[name=%(__name__)s, sender="%%(sender)s", dest="%%(destemail)s",
logpath=%(logpath)s, chain="%%(chain)s"]

# See the IMPORTANT note in action.d/xarf-login-attack for when to use this action
#
# ban & send a xarf e-mail to abuse contact of IP address and include relevant log lines
# to the destemail.
action_xarf = %(banaction)s[name=%(__name__)s, bantime="%%(bantime)s", port="%%(port)s",
protocol="%%(protocol)s", chain="%%(chain)s"]
    xarf-login-attack[service=%(__name__)s, sender="%%(sender)s", logpath=%(logpath)s, port="%%(port)s"]

# ban IP on CloudFlare & send an e-mail with whois report and relevant log lines
# to the destemail.
action_cf_mwl = cloudflare[cfuser="%%(cfemail)s", cftoken="%%(cfapikey)s"]
    %(mta)s-whois-lines[name=%(__name__)s, sender="%%(sender)s", dest="%%(destemail)s",
logpath=%(logpath)s, chain="%%(chain)s"]

# Report block via blocklist.de fail2ban reporting service API
#
# See the IMPORTANT note in action.d/blocklist_de.conf for when to use this action.
# Specify expected parameters in file action.d/blocklist_de.local or if the interpolation
# `action_blocklist_de` used for the action, set value of `blocklist_de_apikey`
```

```

# in your `jail.local` globally (section [DEFAULT]) or per specific jail section (resp. in
# corresponding jail.d/my-jail.local file).
#
action_blocklist_de = blocklist_de[email="% (sender)s", service=%(filter)s, apikey="% (blocklist_de_apikey)s",
agent="% (fail2ban_agent)s"]

# Report ban via badips.com, and use as blacklist
#
# See BadIPsAction docstring in config/action.d/badips.py for
# documentation for this action.
#
# NOTE: This action relies on banaction being present on start and therefore
# should be last action defined for a jail.
#
action_badips = badips.py[category="% (__name__)s", banaction="% (banaction)s", agent="% (fail2ban_agent)s"]
#
# Report ban via badips.com (uses action.d/badips.conf for reporting only)
#
action_badips_report = badips[category="% (__name__)s", agent="% (fail2ban_agent)s"]

# Report ban via abuseipdb.com.
#
# See action.d/abuseipdb.conf for usage example and details.
#
action_abuseipdb = abuseipdb

```

You can use any action above that you'd like, and even create your own or modify them as you see fit.

Custom Jails

Using Regex and fail2ban's filters, we can create our own jails within fail2ban to define rules specific to requests we may be receiving on our application. For example, add the below to `/etc/fail2ban/jail.local` to define a rule to block attempts to run scripts on the webserver.

```

# Add these lines to /etc/fail2ban/jail.local

[nginx-noscript]

enabled = true
port    = http,https
filter  = nginx-noscript

```

```
logpath = /var/log/nginx/access.log
maxretry = 6
```

Now, we need to define the filter for the jail we just created. Fail2ban stores these within `/etc/fail2ban/filter.d/`. So, for our example we will create the `nginx-noscript.conf` file within this directory. Its important that the name we choose here corresponds with what we named our jail in `jail.local`.

```
# New definition for /etc/fail2ban/filter.d/nginx-noscript.conf jail
[Definition]

failregex = ^<HOST> .*GET.*(\.php|\.asp|\.exe|\.pl|\.cgi|\.scgi)

ignoreregex =
```

If you want to test this regex, you can use `fail2ban-regex` to do so on any log file. The command below is an example of testing a regex statement on an nginx log. This command will output all the matching lines within the log that are captured by the regex, which would result in a ban from fail2ban -

```
sudo fail2ban-regex /var/log/nginx/access.log '^<HOST>.*\"(.|)|\\x.*\"$' --print-all-matched
```

Jail Status

To check on the status of running jails, see the command below

```
sudo fail2ban-client status
```

When users are banned under a jail, you can see a list of them by running the following, where `nginx-http-auth` can be changed out for any name of a running jail.

```
sudo fail2ban-client status nginx-http-auth
```

To unban an IP from a jail, run the below

```
sudo fail2ban-client set nginx-http-auth unbanip 124.45.123.777.
```

To unban all IPs from a given jail

```
sudo fail2ban-client restart --unban nginx-http-auth
```

Log Files

Fail2ban's logs will look similar to the below -

```
sudo cat /var/log/fail2ban.log | tail
2019-11-24 17:16:24,307 fail2ban.filter      [27297]: INFO    [nginx-noscript] Found 62.234.108.37 - 2019-11-24
17:16:24
2019-11-24 17:16:25,009 fail2ban.filter      [27297]: INFO    [nginx-noscript] Found 62.234.108.37 - 2019-11-24
17:16:24
2019-11-24 17:16:25,425 fail2ban.filter      [27297]: INFO    [nginx-noscript] Found 62.234.108.37 - 2019-11-24
17:16:25
2019-11-24 17:16:25,481 fail2ban.actions     [27297]: NOTICE [nginx-noscript] Ban 62.234.108.37 - 2019-11-
24 17:17:25
```

Fail2ban keeps these logs within the `/var/log/fail2ban.log` file, we can use these logs with the commands below to create useful reports for hardening your server or tuning your rules. Take notice of the difference in the use of `zgrep` and `grep` below, where we are either searching recent logs or all the logs stored on the system.

```
# Report on all logs for summary of bans triggered sorted by jail, grouped by dates -
```

```
sudo zgrep -h "Ban " /var/log/fail2ban.log* | awk '{print $6,$1}' | sort | uniq -c
3 [nginx-noscript] 2019-10-27
4 [nginx-noscript] 2019-10-28
4 [nginx-noscript] 2019-10-29
6 [nginx-jail2] 2019-10-27
1 [nginx-jail2] 2019-10-28
2 [nginx-jail2] 2019-10-29
```

```
# Log report for bans triggered today only, grouped by IP and hostname -
```

```
grep "Ban " /var/log/fail2ban.log | grep `date +%Y-%m-%d` | awk '{print $NF}' | sort | awk '{print $1,"("$1")"}' |
logresolve | uniq -c | sort -n
1 217.147.85.78 (217.147.85.78)
1 61-219-11-153.HINET-IP.hinet.net (61.219.11.153)
1 85.93.20.70 (85.93.20.70)
1 ip50.ip-51-83-234.eu (51.83.234.50)
1 vmi185089.contaboserver.net (5.189.189.207)
1 vmi214529.contaboserver.net (213.136.87.57)
2 62.234.108.37 (62.234.108.37)
```

```
# Log report for bans triggered today only, grouped by IP and jail -
```

```
sudo grep "Ban " /var/log/fail2ban.log | awk '{print $6,$8}' | sort | uniq -c | sort -n
1 [nginx-jail1] 61.219.11.153
```

```
1 [nginx-jail1] 85.93.20.70
1 [nginx-jail2] 138.68.247.104
1 [nginx-jail2] 213.136.87.57
1 [nginx-jail2] 217.147.85.78
1 [nginx-jail2] 5.189.189.207
1 [nginx-noscript] 51.83.234.50
2 [nginx-noscript] 62.234.108.37
5 [nginx-noscript] 51.83.234.50
4 [nginx-noscript] 62.234.108.37
```

Log report for all bans triggered within a logfile, sorted by date, grouped by jail

```
sudo grep "Ban " /var/log/fail2ban.log.1 | awk '{print $1, $6}'|sort | uniq -c
```

```
1 2020-02-17 [nginx-noscript]
3 2020-02-18 [nginx-noscript]
1 2020-02-18 [nginx-wplogin]
158 2020-02-19 [nginx-noscan]
2 2020-02-19 [nginx-noscript]
15 2020-02-19 [sshd-badproto]
```

Since nginx-noscan is a permanent ban, the high number above is the jail restoring bans after a manual reboot on the 19th

Report on all logs for summary of bans triggered, grouped by IP and jail -

```
sudo awk '($NF-1) = /Ban/){print $NF,"("$NF")"}' /var/log/fail2ban.log* | sort | logresolve | uniq -c | sort -n
```

```
1 mail.grayson-college.info (162.253.219.14)
1 new.wigroup.com.br (159.89.144.7)
1 srvcpnl02.ativy.com (201.7.210.50)
1 vps-01.naftalie.net (142.44.240.254)
2 106.12.54.100 (106.12.54.100)
2 106.13.228.250 (106.13.228.250)
2 111.20.55.66 (111.20.55.66)
```

Log report for bans triggered today only, grouped and sorted by IP -

```
sudo awk '($NF-1) = /Ban/){print $NF}' /var/log/fail2ban.log | sort | uniq -c | sort -n
```

```
1 131.108.164.19
1 138.68.247.104
1 213.136.87.57
1 217.147.85.78
1 5.189.189.207
1 51.83.234.50
1 61.219.11.153
```



```
1 85.93.20.70
2 62.234.108.37
```

Report on all logs for summary of bans triggered, grouped and sorted by IP -

```
sudo awk '($NF-1) = /Ban/){print $NF,"("$NF")"}' /var/log/fail2ban.log* | sort | logresolve | uniq -c | sort -n
```

```
2 79.143.186.114
3 79.143.187.243
2 79.143.188.161
2 80.211.6.136
2 80.211.85.67
2 80.241.220.101
2 80.241.221.67
3 80.82.70.118
1 85.93.20.70
2 87.98.136.163
3 89.208.209.125
2 89.238.186.229
2 91.121.106.6
2 91.121.157.178
3 91.121.70.155
3 91.121.76.97
2 91.123.204.139
3 91.194.90.159
1 94.180.250.158
```

Report on all logs for summary of bans triggered, grouped and sorted by truncated IPs -

```
zgrep -h "Ban " /var/log/fail2ban.log* | awk '{print $NF}' | awk -F\.. '{print $1"."$2"."}' | sort | uniq -c | sort -n |
tail
```

```
1 101.200.
1 103.60.
1 103.98.
1 106.120.
1 106.13.
1 106.54.
1 107.6.
1 114.115.
1 114.215.
1 122.51.
1 123.207.
1 129.146.
```

□ ...

Output reduced

...

8 46.101.

10 91.121.

11 159.65.

11 79.143.

12 51.68.

13 51.38.

19 207.180.

22 173.212.

22 5.189.

33 173.249.

Report on all logs for summary of bans triggered, grouped and sorted by truncated IPs -

Pipe through tail to create a smaller report of most offensive subnets

```
zgrep -h "Ban " /var/log/fail2ban.log* | awk '{print $NF}' | awk -F\. '{print $1"."$2"."}' | sort | uniq -c | sort -n | tail
```

8 46.101.

10 91.121.

11 159.65.

11 79.143.

12 51.68.

13 51.38.

19 207.180.

22 173.212.

22 5.189.

33 173.249.

“ Art of the web

Revision #11

Created 14 April 2019 03:10:22 by Shaun Reed

Updated 22 July 2021 13:59:30 by Shaun Reed