fail2ban add custom filters

i love fail2ban and i think it should be enabled and configured by default in any linux server (and maybe even workstations) .. if you don't know it: fail2ban checks your logs and blocks ip's who repeatedly do something stupid on your system.. in more detail: fail2ban has a set of **filters** which contain regular expressions that are matched against a specific log file. In addition to that, fail2ban has **jails** which you define for each filter. A jail defines, how many times a filter's regex may match within what time frame for the same IP address. It also defines, how long that IP should be blocked and how exactly this should be achieved once it exceeds the allowed thresholds of regex matches.

usually your favorite distribution comes packed with filters and all you may have to do is enable them.. however, sometimes you might want to write your own rule, be it for your own application or for some case which is simply not covered by the default rules..

here is an example of how I added a new rule for postfix running on an **ubuntu server** which filters out some bots who try to brute-force smtpd accounts. most of those are covered by the default filter in ubuntu, however, i had a case of a bot which tried to authenticate on a smtpd which only allowed TLS but it did not use TLS.. so the bot will honestly never suceed with this method of course, but it still was flooding my logs, so i decided to do something against that..

NOTE Please keep in mind, that path names and best practices on where to save your configs vary slightly from distribution to distribution.. debian and therefore ubuntu style is to not edit distribution provided config files and instead use the `.d` directory with the same basename as the config file you want to edit and then create a new `.conf` in there which overwrites the defaults you want to change or adds to the config.. this of course only works for apps that support includes and the necessary tools in their config parsing mechanisms, but luckily fail2ban is one of those, so we keep it debian friendly, which will help when you upgrade your system (it won't ask you if you want to keep your old config or overwrite it with the default).

first let's look a the log entries which identify that sucker:

```
Dec  6 07:22:10 mail postfix/submission/smtpd[24962]: connect from
unknown[170.106.11.80]
Dec  6 07:22:11 mail postfix/submission/smtpd[24962]: lost connection after
EHLO from unknown[170.106.11.80]
Dec  6 07:22:11 mail postfix/submission/smtpd[24962]: disconnect from
unknown[170.106.11.80] ehlo=1 auth=0/1 commands=1/2
```

we certainly don't want to block lline 1, as this would limit the number of connections per given timeframe you are allowed to make.. i don't want to limit that generally.. I also don't want to scan for line 2 as EHLO does not seem very harmful and you never know, maybe some appliatons will use that extensively for whatever reason.. but line 3 looks specific enough that I think it probably should not match any “normal” operations too often.. so let's go for that one

first we create a new filter in `/etc/fail2ban/filter.d/postfix-ehlo.conf`

/etc/fail2ban/filter.d/postfix-ehlo.conf

```
[INCLUDES]
before = common.conf
```

```
[Definition]
_daemon = postfix/smtpd
failregex = ^%(__prefix_line)spostfix/submission/smtpd.*disconnect from
unknown\[<HOST>\] ehlo=1 auth=0/1 commands=1/2$
ignoreregex =
```

HINT to figure out the regex, you can use `grep -P "my regex.." /var/log/mail.log` for example and then replace the `Dec 6 07:22:11 mail` part with `%(prefix_line)s`

now we can test this filter against our log file using `fail2ban-client`:

```
fail2ban-regex /var/log/mail.log /etc/fail2ban/filter.d/postfix-ehlo.conf
```

the important line we are looking for is this one:

```
Lines: 27233 lines, 0 ignored, 9005 matched, 18228 missed
```

s so you can see, the regex matched 9005 lines, that seems about right..

next up is our **jail** for the filter we just created. we create a new file called '/etc/fail2ban/jail.d/postfix-ehlo.conf'

/etc/fail2ban/jail.d/postfix-ehlo.conf

```
enabled  = true
port     = smtp,465,submission
logpath  = %(postfix_log)s
```

I got the `%(postfix_log)` from an existing postifx jail in the same folder.. in any other case I would have simply hardcoded the actual log directory like `/var/log/mail.log` or whatever it is in your case..

there are many more options you could set, like `action`, `maxretry`, `bantime`, `findtime` etc, but most of them are already set in the default config, so no need to adjust those

so now lets reload our config to enable our shiny new rule:

```
fail2ban-client reload
```

if whoever you want to block is still active, you should see him blocked in a short time.. check your logs like

```
# grep Ban /var/log/fail2ban.log
2020-12-06 07:22:19,440 fail2ban.actions        [24010]: NOTICE  [postfix-
ehlo] Ban 170.106.11.80
```

and yell "haha, gotcha"!

or you can also see your success via the `fail2ban-client`:

```
# fail2ban-client status postfix-ehlo
Status for the jail: postfix-ehlo
|- Filter
|  |- Currently failed: 0
|  |- Total failed: 8
|  `- File list:    /var/log/mail.log
`- Actions
   |- Currently banned: 0
   |- Total banned: 1
   `- Banned IP list:   170.106.11.80
```

there you go :)

by the way, to activate any of the already existing jails, you need to set the `enabled` parameter to `yes`. but rather than modifying the distribution supplied configs, I'd recommend to create your own config for example `/etc/fail2ban/jail.d/activejails.conf` with a content like so:

```
[sshd]
enabled = true

[postfix]
enabled = true

[dovecot]
enabled = true
```

the names of the pre-configured jails can be found in `/etc/fail2ban/jails.conf` and are usually identical to the filter name and file name of the filter in `/etc/fail2ban/filter.d/`

after enabling your filters, reload and check with the `fail2ban-client`

```
# fail2ban-client reload
OK
# fail2ban-client status
Status
|- Number of jail:  4
`- Jail list:   dovecot, postfix, postfix-ehlo, sshd
```