Training: Fail2ban to avoid brute-force attacks

Created by Pedro Geraldo on [2019-06-26](190624-training-fail2ban.docx)

Table of Contents

[Introduction 1](#_Toc12460056)

[Prerequisites 1](#_Toc12460057)

[Installation 1](#_Toc12460058)

[Requirements 1](#_Toc12460059)

[Installing on a Debian system 1](#_Toc12460060)

[Configuration files architecture 1](#_Toc12460061)

[General settings 1](#_Toc12460062)

[Jails 1](#_Toc12460063)

[Jail Options 1](#_Toc12460064)

[Filters 1](#_Toc12460065)

[Actions 1](#_Toc12460066)

[Testing 1](#_Toc12460067)

[Limitations 1](#_Toc12460068)

[Reaction time 1](#_Toc12460069)

[Validation/Conclusion 1](#_Toc12460070)

[References 1](#_Toc12460071)

[End of document 1](#_Toc12460072)

# Introduction

Fail2ban is a Linux tool that has the ability to scan log files (e.g. /var/log/apache/error\_log) and ban IPs that show malicious signs (e.g. too many password failures, seeking for exploits, etc). Generally Fail2Ban is then used to update firewall rules (usually on IPTABLES) to reject the IP addresses for a specified amount of time, although any arbitrary other action (e.g. sending an email) could also be configured. Out of the box Fail2Ban comes with filters for various services (apache, courier, ssh, asterisk, etc). However, Fail2Ban is able to reduce the rate of incorrect authentications attempts however it cannot eliminate the risk that weak authentication or other forms of vulnerabilities.

## Prerequisites

You should be able to understand some basics [bash](http://switzernet.com/3/support/people/elen-virabyan/100906-simple-unix-bash-commands/), [Regex](https://docs.switzernet.com/3/public/110324-training-vi-regex/#_Toc289072045) and [IPTABLES](https://docs.switzernet.com/3/support/190529-training-iptables-firewall-rules/) as well, to help understand how things work.

# Installation

## Requirements

In order to use ***Fail2ban***, the following software are required (minimum):

* Python >= 2.4
	+ **Warning**: Python 2.4 does not re-open the syslog socket when it encounters a failure. This is a Python bug. If you want to use "logtarget = SYSLOG", please use a newer version of Python like 2.5.
* Iptables (or nftables)

## Installing on a Debian system

Installing ***Fail2ban*** on a Debian based system is very straightforward, just install the package through Synaptic or

apt-get install fail2ban

# Configuration files architecture

Before we begin, it is important to clarify some terms used in the following sections.

filter : a filter defines a regular expression which must match a pattern corresponding to a log-in failure or any other expression
action : an action defines several commands which are executed at different moments
jail : a jail is a combination of one filter and one or several actions. Fail2ban can handle several jails at the same time

The standard path for the configuration is in /etc/fail2ban. This can be set with the -c option of fail2ban-client. A typical configuration looks like this:

/etc/fail2ban/

├── action.d

│ ├── dummy.conf

│ ├── hostsdeny.conf

│ ├── iptables.conf

│ ├── mail-whois.conf

│ ├── mail.conf

│ └── shorewall.conf

├── fail2ban.conf

├── fail2ban.local

├── filter.d

│ ├── apache-auth.conf

│ ├── apache-noscript.conf

│ ├── asterisk.conf

│ ├── couriersmtp.conf

│ ├── postfix.conf

│ ├── proftpd.conf

│ ├── sshd.conf

├── jail.conf

└── jail.local

Every .conf file can be overridden with a file named .local. The .conf file is read first, then .local, with later settings overriding earlier ones. Thus, a .local file doesn't have to include everything in the corresponding .conf file, only those settings that you wish to override.

Modifications should take place in the .local and not in the .conf. This avoids merging problem when upgrading. These files are well documented and detailed information should be available there.

## General settings

The file *fail2ban.conf* contains general settings for the **fail2ban-server** daemon, such as the logging level and target. You can also specify here the socket path used for communication between the client and the server.

## Jails

The most important file is probably *jail.conf*, which contains the declaration of your jails. By default, some sections are inserted as templates. You must enable the sections of interest and adapt to your local configuration. Here is an example of the ssh-iptables section:

[ssh-iptables]

#enabled = false

enabled = true

filter = sshd

action = iptables[name=SSH, port=ssh, protocol=tcp]

# mail-whois[name=SSH, dest=yourmail@mail.com]

#logpath = /var/log/sshd.log

logpath = /var/log/auth.log

maxretry = 5

With these settings a few things will happen:

1. the section ssh-iptables is *enabled*;
2. the filter *sshd*.conf in sub-directory *filter.d* will be processed;
3. the action(s) described in *iptables*.conf (sub-directory *action.d*) will be executed if the outcome of the filter process is true. In this example, the additional action *mail-whois*.conf is commented out.
4. the log file to be scanned by the filter is *auth.log*.

Filter and actions are combined to create jails. Only one filter is allowed per jail, but it is possible to specify several actions, on separate lines. For example, you can react to a SSH break-in attempt by first adding a new firewall rule, then retrieving some information about the offending host using *whois* and finally sending an e-mail notification. Or maybe you just want to receive a notification on your Jabber account when someone accesses the page */donotaccess.html* on your web server.

***Fail2ban*** is not limited to SSH. It contains default filters and actions for many daemons and services. You can easily modify them or create new ones. If you take a look in the filter.d you will notice a few default filters that don't occur in the standard jail.conf that come with the sources. In this example we take the "sshd-ddos.conf". To integrate the filter into fail2ban edit your jail.conf:

[ssh-ddos]

enabled = true

port = ssh,sftp

filter = sshd-ddos

logpath = /var/log/messages

maxretry = 2

Always remember to adjust $logpath to your log-file as mentioned above.

### Jail Options

Every jail can be customized by tuning following options:

|  |
| --- |
| Jail Options |
| **Name** | **Default** | **Description** |
| **filter** |  | Name of the filter to be used by the jail to detect matches. Each single match by a filter increments the counter within the jail |
| **logpath** | **/var/log/messages** | Path to the log file which is provided to the filter |
| **maxretry** | **3** | Number of matches (i.e. value of the counter) which triggers ban action on the IP. |
| **findtime** | **600 sec** | The counter is set to zero if no match is found within "findtime" seconds. |
| **bantime** | **600 sec** | Duration (in seconds) for IP to be banned for. Negative number for "permanent" ban. |

## Filters

The directory *filter.d* contains mainly regular expressions which are used to detect break-in attempts, password failures, etc. Here is an example for *filter.d/sshd.conf* with 3 possible regular expressions to match the lines of the logfile:

failregex = Authentication failure for .\* from <HOST>

 Failed [-/\w]+ for .\* from <HOST>

 ROOT LOGIN REFUSED .\* FROM <HOST>

 [iI](?:llegal|nvalid) user .\* from <HOST>

In the above example the default regex has been changed to allow the absence of *user* in a line such as:

Jan 10 07:02:37 homebrou sshd[18419]: Failed password for root from 222.76.213.151 port 55236 ssh2

If you're creating your own failregex expressions, here are some things you should know:

* A failregex can have multiple lines, any one of which may match a line of the log file.
* In every line of failregex, the part that matches the host name or IP address must be wrapped in a (?P<host> ... ) sandwich. This is a [Python-specific regex extension](http://docs.python.org/library/re.html) that assigns the contents of the match to the name <host>. The <host> tag is how you tell fail2ban which host was connecting, so it has to be present in every line of failregex. If it's not, fail2ban will issue an error message about "No 'host' group".
* As a convenience, you can use the predefined entity <HOST> in your regexes. <HOST> is an alias for (?:::f{4,6}:)?(?P<host>\S+), which matches either a hostname or an IPv4 address (possibly embedded in an IPv6 address).
* In the action scripts, the tag <ip> will be replaced by the IP address of the host that was matched in the <host> tag.
* In order for a log line to match your failregex, it actually has to match in two parts: the beginning of the line has to match a timestamp pattern or regex, and the remainder of the line has to match your failregex. If the failregex is anchored with a leading ^, then the anchor refers to the start of the remainder of the line, *after* the timestamp and intervening whitespace.
* The pattern or regex to match the time stamp is currently not documented, and not available for users to read or set. See [Debian bug #491253](http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=491253%7C). This is a problem if your log has a timestamp format that fail2ban doesn't expect, since it will then fail to match any lines. Because of this, you should test any new failregex against a sample log line, as in the examples below, to be sure that it will match. If fail2ban doesn't recognize your log timestamp, then you have two options: either reconfigure your daemon to log with a timestamp in a more common format, such as in the example log line above; or file a bug report asking to have your timestamp format included.

As an example of the above points, run the following commands in your console and compare the results:

fail2ban-regex 'Jul 18 12:13:01 [1.2.3.4] authentication failed' 'authentication failed'

fail2ban-regex 'Jul 18 12:13:01 [1.2.3.4] authentication failed' '\[<HOST>\] authentication failed'

fail2ban-regex '18-07-2008 12:13:01 [1.2.3.4] authentication failed' '\[<HOST>\] authentication failed'

fail2ban-regex '18-7-2008 12:13:01 [1.2.3.4] authentication failed' '\[<HOST>\] authentication failed'

The 1st command fails, with a "No 'host' group" error message. The 2nd command succeeds, and finds the host address 1.2.3.4. The 3rd command succeeds, and finds the host address 1.2.3.4 (at least with v0.8.4). The 4th command fails. It does tell you why---it says "...no valid date/time found for..."---and clearly it's not able to match the unusual timestamp format---which is *not* part of your failregex.

## Actions

The directory *action.d* contains different scripts defining actions. The actions are executed at well-defined moments during the execution of ***Fail2ban***: when starting/stopping a jail, banning/unbanning a host, etc.

## Testing

Many things can be tested after configuration, but the following commands can help to verify your settings:

# fail2ban-client -d

will dump the current configuration.

# fail2ban-regex "line" "failregex"

will test a *single* regular expression **failregex** (such as given in *sshd.conf*) with a *single* line of your logfile. Don't forget the double quotes around your *line* and *failregex* definitions.

**fail2ban-regex** accepts files too. Thus, it is easier to test and debug your own regular expressions.

# fail2ban-regex /var/log/auth.log /etc/fail2ban/filter.d/sshd.conf

You can use a combination of both:

# fail2ban-regex /var/log/auth.log "Failed [-/\w]+ for .\* from <HOST>"

# Limitations

## Reaction time

First of all, remember that ***Fail2ban*** is a log parser. It cannot do anything before something is written in the log files. Lots of syslog daemons buffer their outputs. This can impact performance of ***Fail2ban***. Thus, it could be good to disable buffering of your syslog daemon.

It is quite difficult to evaluate the reaction time. ***Fail2ban*** waits 1 second before checking for new logs to be scanned. This should be fine in most cases. However, it is possible to get more login failures than specified by **maxretry**.

Here is an excerpt of a */var/log/auth.log*:

Jan 15 15:28:46 homebrou sshd[29778]: Invalid user recruit from 81.74.87.66

Jan 15 15:28:46 homebrou sshd[29778]: error: Could not get shadow information for NOUSER

Jan 15 15:28:46 homebrou sshd[29778]: Failed password for invalid user recruit from 81.74.87.66 port 47672 ssh2

Jan 15 18:33:27 homebrou sshd[2156]: Did not receive identification string from 88.191.23.27

Jan 15 19:23:37 homebrou sshd[3418]: Invalid user test from 88.191.23.27

Jan 15 19:23:37 homebrou sshd[3418]: error: Could not get shadow information for NOUSER

Jan 15 19:23:37 homebrou sshd[3418]: Failed password for invalid user test from 88.191.23.27 port 41017 ssh2

Jan 15 19:23:38 homebrou sshd[3420]: Invalid user test from 88.191.23.27

Jan 15 19:23:38 homebrou sshd[3420]: error: Could not get shadow information for NOUSER

Jan 15 19:23:38 homebrou sshd[3420]: Failed password for invalid user test from 88.191.23.27 port 41096 ssh2

Jan 15 19:23:38 homebrou sshd[3422]: Invalid user test from 88.191.23.27

Jan 15 19:23:38 homebrou sshd[3422]: error: Could not get shadow information for NOUSER

Jan 15 19:23:38 homebrou sshd[3422]: Failed password for invalid user test from 88.191.23.27 port 41162 ssh2

Jan 15 19:23:38 homebrou sshd[3424]: Invalid user test from 88.191.23.27

Jan 15 19:23:38 homebrou sshd[3424]: error: Could not get shadow information for NOUSER

Jan 15 19:23:38 homebrou sshd[3424]: Failed password for invalid user test from 88.191.23.27 port 41209 ssh2

Jan 15 19:23:39 homebrou sshd[3426]: Invalid user test from 88.191.23.27

Jan 15 19:23:39 homebrou sshd[3426]: error: Could not get shadow information for NOUSER

Jan 15 19:23:39 homebrou sshd[3426]: Failed password for invalid user test from 88.191.23.27 port 41267 ssh2

Jan 15 19:23:39 homebrou sshd[3428]: Invalid user test from 88.191.23.27

Jan 15 19:23:39 homebrou sshd[3428]: error: Could not get shadow information for NOUSER

Jan 15 19:23:39 homebrou sshd[3428]: Failed password for invalid user test from 88.191.23.27 port 41323 ssh2

Jan 15 19:23:40 homebrou sshd[3430]: Invalid user test from 88.191.23.27

Jan 15 19:23:40 homebrou sshd[3430]: error: Could not get shadow information for NOUSER

Jan 15 19:23:40 homebrou sshd[3430]: Failed password for invalid user test from 88.191.23.27 port 41376 ssh2

Jan 15 19:23:40 homebrou sshd[3433]: Invalid user test from 88.191.23.27

Jan 15 19:23:40 homebrou sshd[3433]: error: Could not get shadow information for NOUSER

Jan 15 19:23:40 homebrou sshd[3433]: Failed password for invalid user test from 88.191.23.27 port 41433 ssh2

Jan 15 19:23:41 homebrou sshd[3435]: Invalid user test from 88.191.23.27

Jan 15 19:23:41 homebrou sshd[3435]: error: Could not get shadow information for NOUSER

Jan 15 19:23:41 homebrou sshd[3435]: Failed password for invalid user test from 88.191.23.27 port 41484 ssh2

Jan 16 12:13:43 homebrou sshd[32249]: Did not receive identification string from 209.126.131.150

Looking at */var/log/fail2ban.log* around the corresponding period:

2007-01-15 15:38:47,142 fail2ban.actions: WARNING [ssh-iptables] Unban 81.74.87.66

2007-01-15 19:23:41,175 fail2ban.actions: WARNING [ssh-iptables] Ban 88.191.23.27

2007-01-15 19:23:42,373 fail2ban.actions: WARNING [ssh-iptables] 88.191.23.27 already banned

2007-01-15 19:33:41,508 fail2ban.actions: WARNING [ssh-iptables] Unban 88.191.23.27

2007-01-16 12:29:50,496 fail2ban.actions: WARNING [ssh-iptables] Ban 209.126.131.150

Thus, you can see that between 19:23:37 and 19:23:41, i.e. within 4 seconds, 9 login (ssh) attempts (instead of only 3) from 88.191.23.27 have been recorded in *auth.log* before it has been banned by ***Fail2ban***.

# Validation

For validation, ask for a server as training environment, install fail2ban (if it is not already) and create a doc where you add screenshots of the following tasks

1. In a jail.local file, enable the jail for ssh, set the max attempts to 2 and the ban time to 3 minutes
2. Make sure you have a filter (in filter.d/sshd.conf) for unsuccessful attempts and underline it on the image (if it doesn’t exist, create one)
3. In your local computer, attempt several ssh attempts to the training server and report its behaviour.

**Note:** the server needs to be “cleaned” after this, so that other colleagues can do the training without spoilers.

# References

<http://switzernet.com/3/support/people/elen-virabyan/100906-simple-unix-bash-commands/>

<https://docs.switzernet.com/3/public/110324-training-vi-regex/#_Toc289072039>

<https://docs.switzernet.com/3/support/190529-training-iptables-firewall-rules/>

<https://www.fail2ban.org/wiki/index.php/MANUAL_0_8>

# End of document

\* \* \*

© 4z.com