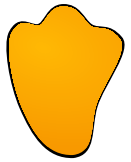
Intrusion prevention per tutti: introduzione a Fail2ban



LDMI 2024

Luca **Mauri** | 2024-10-26

Di cosa si tratta

- Fail2Ban è uno strumento di **sicurezza** *open-source* che rientra nella categoria di *Intrusion Prevention System*
- Le sue caratteristiche
  - Identifica le minacce
  - Blocca automaticamente gli IP sospetti
  - Protegge servizi come SSH, HTTP, FTP, Postfix
  - Gestione dinamica delle blacklist
  - Scritto in Python
  - Si appoggia strumenti come `iptables`, `Packet Filter` o `tcp_wrappers`

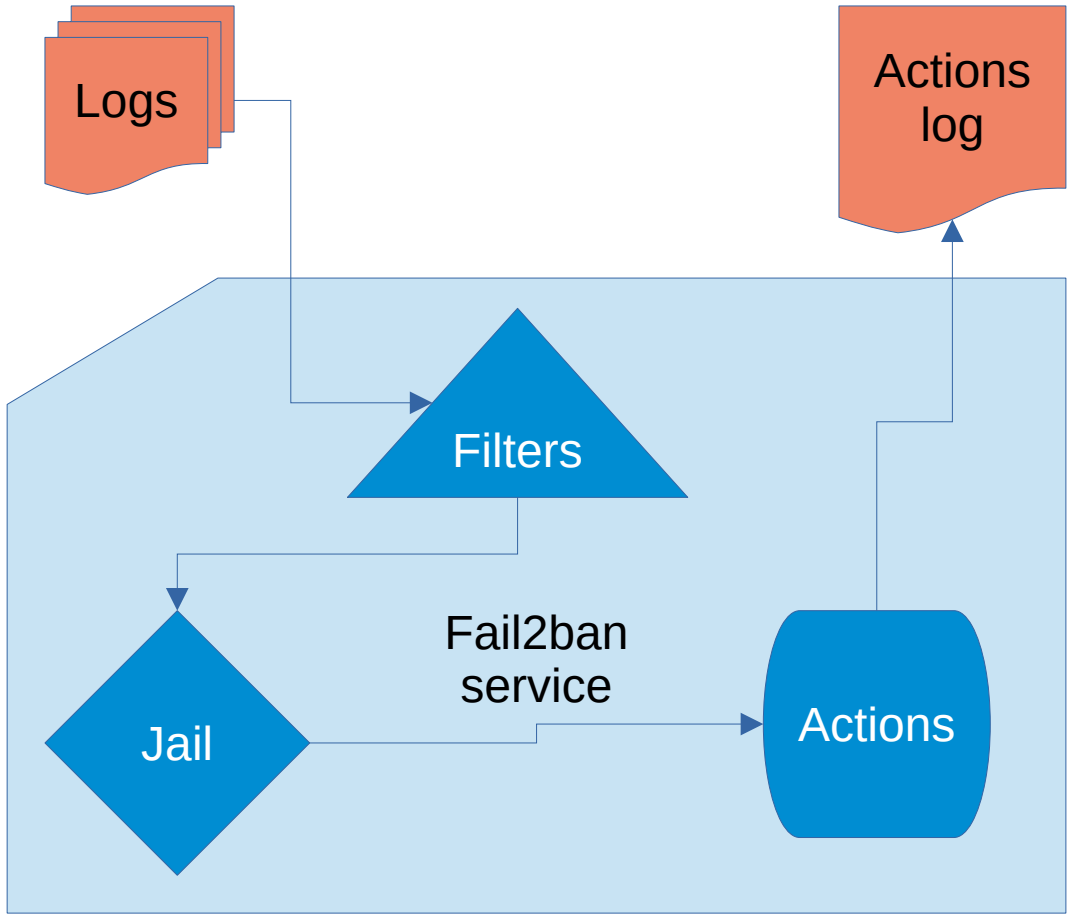
Breve storia

- Creato nel **2004** da Cyril Jaquier e da lui gestito fino al 2010
  - Da quel momento è diventato un progetto *community-driven*
  - La **comunità** attiva che contribuisce con aggiornamenti e filtri personalizzati
- Versioni principali:
  - 0.7.x: riscrittura quasi completa con architettura client/server e multithreading
  - 0.9.x: Migliorato supporto per più servizi
  - 1.x: Maggiore modularità
- È diventato uno strumento **standard** per la sicurezza dei server Linux e viene utilizzato in molte distribuzioni.

Dettagli tecnici

Flusso di lavoro

- Il servizio Fail2Ban legge i **log** dei servizi configurati
- Applica i **filtri** per identificare le violazioni o pattern specifici
- In base al numero di anomalie e alla loro frequenza, applica una ***jail***
- La jail genera una o più **azioni**, per esempio un blocco sul firewall



Architettura

- Fail2Ban **Server**: Il servizio principale che esegue il monitoraggio e l'analisi dei log.
- ***Filters***: Definiscono le regole per identificare i tentativi di accesso non autorizzati (failregex).
- ***Jails***: Configurazioni specifiche per ciascun servizio monitorato, che definiscono le azioni da intraprendere.
- ***Actions***: definiscono i comandi da eseguire
- In estrema sintesi:
  - Le *jail* dicono a Fail2ban cosa **guardare** e **quali** azioni applicare
  - I filtri spiegano **come** guardare
  - Le azioni definiscono nei dettagli **cosa** fare

Installazione e note generali

- L'installazione è normalmente molto semplice perché, nella maggior parte delle distribuzioni, il software è già **pacchettizzato**
- Necessita un interprete Python
- Installa un servizio (`fail2ban.service`), per default **disabilitato**
- Comprende configurazioni standard (come `fail2ban.conf` e `jail.conf`) da personalizzare con relativi file `.local`
  - Più dettagli a seguire

```
/etc/fail2ban
 |-action.d
 |-fail2ban.d
 |-filter.d
 |-jail.d
  fail2ban.conf
  fail2ban.local
  jail.conf
  jail.local
  paths-arch.conf
  paths-common.conf
  paths-debian.conf
  paths-opensuse.conf
```

Installazione e note generali

- All'avvio di fail2ban, i file che vengono letti in **successione** per generare la configurazione di lavoro
- L'ordine è:
  - Prima `fail2ban.conf` e `jail.conf`
  - Il relativo file `fail2ban.local`
  - Poi tutti i file `jail.d/*.conf`
  - Segue `jail.local`
  - Tutti i file `jail.d/*.local`
  - I file nelle cartelle sono processati in ordine alfabetico

```
/etc/fail2ban
 |-action.d
 |-fail2ban.d
 |-filter.d
 |-jail.d
  fail2ban.conf
  fail2ban.local
  jail.conf
  jail.local
  paths-arch.conf
  paths-common.conf
  paths-debian.conf
  paths-opensuse.conf
```

La configurazione principale

- Il file `fail2ban.conf` contiene le configurazioni base del **servizio**:
  - Non va modificato, pena la **sovrascrittura** in un futuro aggiornamento, ma affiancato dal file `fail2ban.local`
- Contiene le impostazioni **basilari** del software

```
[DEFAULT]
loglevel = INFO
logtarget = /var/log/fail2ban.log
syslogsocket = auto
socket =
/var/run/fail2ban/fail2ban.sock
pidfile =
/var/run/fail2ban/fail2ban.pid
allowipv6 = auto
dbfile =
/var/lib/fail2ban/fail2ban.sqlite3
dbpurgeage = 1d
dbmaxmatches = 10
```

Il file *jail* principale

- Le *jail* dicono a Fail2ban cosa **guardare** e **quali** azioni applicare
- Il file jail.conf contiene le configurazioni base delle varie *jail*.
  - Stesso principio di prima con jail.local
  - Fatte le impostazioni predefinite nel file, singole *jail* possono essere memorizzate nella directory jail.d/

```
[DEFAULT]
ignoreip = 127.0.0.1/8
bantime = 10m
findtime = 10m
maxretry = 3
backend = auto
usedns = warn
enabled = false
destemail = root@localhost
sendername = Fail2Ban
banaction = iptables-multiport
mta = sendmail
protocol = tcp
chain = INPUT
action_ = %(banaction)s[name=%(__name__)s, port="%
(port)s", protocol="%(protocol)s", chain="%
(chain)s"]
action_mw = %(banaction)s[name=%(__name__)s, port="%
(port)s", protocol="%(protocol)s", chain="%
(chain)s"]
            %(mta)s-whois[name=%(__name__)s, dest="%
(destemail)s", protocol="%(protocol)s", chain="%
(chain)s", sendername="%(sendername)s"]
action_mwl = %(banaction)s[name=%(__name__)s,
port="%(port)s", protocol="%(protocol)s", chain="%
(chain)s"]
            %(mta)s-whois-lines[name=%(__name__)s,
dest="%(destemail)s", logpath=%(logpath)s, chain="%
(chain)s", sendername="%(sendername)s"]
action = %(action_)s
```

Il file *jail* principale

- Vediamo la struttura **base** di `jail.conf`
  - `findtime` il software conterà gli accessi falliti in una finestra di 10
  - `banaction` punta all'omonimo file in action.d
  - `%(var_name)s` serve per la sostituzione di stringhe definite più in alto nel file

```
[DEFAULT]
ignoreip = 127.0.0.1/8
bantime = 10m
findtime = 10m
maxretry = 3
backend = auto
usedns = warn
destemail = root@localhost
sendername = Fail2Ban
banaction = iptables-multiport
mta = sendmail
protocol = tcp
chain = INPUT
action_ = %(banaction)s[name=%(__name__)s, port="%
(port)s", protocol="%(protocol)s", chain="%
(chain)s"]
action_mw = %(banaction)s[name=%(__name__)s, port="%
(port)s", protocol="%(protocol)s", chain="%
(chain)s"]
            %(mta)s-whois[name=%(__name__)s, dest="%
(destemail)s", protocol="%(protocol)s", chain="%
(chain)s", sendername="%(sendername)s"]
action_mwl = %(banaction)s[name=%(__name__)s,
port="%(port)s", protocol="%(protocol)s", chain="%
(chain)s"]
            %(mta)s-whois-lines[name=%(__name__)s,
dest="%(destemail)s", logpath=%(logpath)s, chain="%
(chain)s", sendername="%(sendername)s"]
action = %(action_)s
```

Anatomia delle *jails*

- Come descritto sopra, le *jails* possono essere definite in diverse **ubicazioni**, ma il loro funzionamento è sempre identico
- Vediamo uno degli esempi più basilari con **SSH**
  - La *jail* è attiva
  - La porta è chiamata per nome (da `/etc/services`)
  - Il filtro si chiama `sshd`
  - Massimo 6 ripetizione dell'azione (nella **finestra**)
- Tutto il resto viene dal DEFAULT
  - Per esempio la `banaction`!

```
[SSH]
enabled     = true
port        = ssh
filter      = sshd
logpath     = /var/log/auth.log
maxretry    = 6
```

Anatomia dei *filters*

- I filtri spiegano **come** guardare
- Continuiamo con l'esempio di SSH
  - Carica una **configurazione** basica (comprende per esempio la definizione di `__prefix_line`)
  - Definisce il **servizio** da analizzare
  - Imposta (una o più) *regular expression* con cui effettuare il **match**
  - Si può impostare una regex per **ignorare** specifiche linee nel file

```
[INCLUDES]
before = common.conf

[Definition]
_daemon = sshd
failregex = ^%(__prefix_line)s(?:error:
PAM: )?[aA]uthentication (?:failure|
error) for .* from <HOST>( via \S+)?\s*$
            ^%(__prefix_line)s(?:error:
PAM: )?User not known to the underlying
authentication module for .* from
<HOST>\s*$
            ^%(__prefix_line)sFailed \S+
for .*? from <HOST>(?: port \d*)?(?:
ssh\d*)?(: (ruser .*|(\S+ ID \S+ \(serial
\d+\) CA )?\S+ %(__md5hex)s(, client user
".*", client host ".*")?))?\s*$
            …
ignoreregex =
```

Digrssione sui parametri

- Un estratto qui sotto di un log di SSH
- La prima parte di ogni entry è specifica per lo OS su cui stiamo eseguendo il software
  - Ecco la ragione per il parametro `__prefix_line`

```
May  6 18:18:52 localhost sshd[3534]: pam_unix(sshd:auth): authentication failure;
logname= uid=0 euid=0 tty=ssh ruser= rhost=101.79.130.213
May  6 18:18:54 localhost sshd[3534]: Failed password for invalid user phil from
101.79.130.213 port 38354 ssh2
May  6 18:18:54 localhost sshd[3534]: Received disconnect from 101.79.130.213: 11: Bye
Bye [preauth]
```

Breve digressione sulle *regex*

- La variante di RegEx usata è quella di **Python** dato che F2B è scritto in quel linguaggio
- Nella documentazione del software esiste una **sezione** specifica https://fail2ban.readthedocs.io/en/latest/filters.html
- C'è uno **strumento** da riga di comando chiamato `fail2ban-regex` che consente di testare le espressioni su frammenti di log in maniera rapida
  - `fail2ban-regex` *`riga pattern`*
  - `fail2ban-regex /home/user/doc/test.log myFilter.conf`

Anatomia delle *actions*

- Le *actions* definiscono nei dettagli **cosa** fare (ricordarsi il *default*!)
- Per continuare con **SSH**, la azione chiamata precedentemente è `iptables-multiport` (file omonimo)
  - Importa le restrizioni (rifiuta i pacchetti e rifiuta i ping)
  - Definisce come creare la regola, bannare e rimuove i ban (succede quando il `bantime` è passato)
  - `Init` definisce alcuni parametri nel caso mancassero nella richiesta
- I paramentri sono definiti da `<...>`

```
[INCLUDES]
before = iptables-blocktype.conf
[Definition]
actionstart = iptables -N fail2ban-<name>
              iptables -A fail2ban-
<name> -j RETURN
              iptables -I <chain> -p
<protocol> -m multiport --dports <port>
-j fail2ban-<name>
actionstop = iptables -D <chain> -p
<protocol> -m multiport --dports <port>
-j fail2ban-<name>
actioncheck = iptables -n -L <chain> |
grep -a 'fail2ban-<name>[ \t]'
actionban = iptables -I fail2ban-<name> 1
-s <ip> -j <blocktype>
actionunban = iptables -D fail2ban-<name>
-s <ip> -j <blocktype>
[Init]
name = default
port = ssh
protocol = tcp
chain = INPUT
```

Riepilogo

- Caricamento del **servizio** e della configurazione iniziale
- Analisi delle *jails* in **tutti** i file
- Ricerca delle actions in tutti i file. In base alle azioni e ai relativi parametri, genera le **regole**
- Analisi dei **filter**
- In base ai dati letti nelle jails, analizza tutti i file di **log**.
- Cerca tutti i filters e li usa per **confrontare** ogni nuova riga aggiunta ai log
- Un **contatore** si incrementa all'interno della finestra definita: una volta che supera `maxretry`, il servizio lancia actioncheck e poi actionban
  - Una volta **raggiunto** il limite di `bantime`, allora l'azione `actionunban` viene automaticamente lanciata

Strumenti utili

- Abbiamo **già** discusso della utility `fail2ban-regex` per analizzare le regex
- **Altro** tool molto comodo è `fail2ban-client` lanciandolo con il parametro status e poi il nome di una jail
  - Vediamo, per esempio, il **risultato** di `fail2ban-client status nginx-http-auth`

```
$ sudo fail2ban-client status nginx-http-auth
Status for the jail: nginx-http-auth
|- Filter
|  |- Currently failed: 0
|  |- Total failed:     5
|  `- File list:        /log/nginx/error.log
`- Actions
   |- Currently banned: 1
   |- Total banned:     1
   `- Banned IP list:   108.172.85.62
```

Casi di studio

Caso di studio – Sorpresa!

- Un bel giorno mi sono svegliato con questa situazione:
  - load average: **21.64, 27.05, 29.16**
  - MediaWiki internal error.
    original exception: [19a9543617c606274c438eh9]
    /wiki/Pagina_principaIe
    wikimedialRdhms\DBConnectionError: Cannot access the database: **Too many connections** (localhost)

```
3.141.7.240 - - [28/Sep/2024:15:42:17 +0200] "GET /wt/index.php?
target=Jordan+Lund&title=Speciale%3APuntanoQui HTTP/1.1" 200 59560 "-" "Mozilla/5.0
AppleWebKit/537.36 (KHTML, like Gecko; compatible; ClaudeBot/1.0; +claudebot@anthropic.com)"
47.128.48.37 - - [28/Sep/2024:15:42:16 +0200] "GET /wt/index.php?
direction=prev&mobileaction=toggle_view_mobile&oldid=45445&title=Astronave_di_classe_K%27Vort_
%28disambigua%29 HTTP/1.1" 200 56033 "-" "Mozilla/5.0 (Linux; Android 5.0) AppleWebKit/537.36
(KHTML, like Gecko) Mobile Safari/537.36 (compatible; Bytespider; spider-
feedback@bytedance.com)"
100.24.149.244 - - [28/Sep/2024:15:42:26 +0200] "GET /wt/index.php?
article=Federation&dir=out&mobileaction=toggle_view_mobile&offset=0&title=Speciale:Esplora
HTTP/1.1" 200 46343 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_1) AppleWebKit/600.2.5
(KHTML, like Gecko) Version/8.0.2 Safari/600.2.5 (Amazonbot/0.1;
+https://developer.amazon.com/support/amazonbot)"
```

Caso di studio - non solo *intrusion*

- Vediamo un caso **reale** di un utilizzo non completamente *ortodosso* di fail2ban
- Un problema **comune** che sta assumendo dimensioni sempre più grandi è lo *scraping* dei siti effettuati dai bot AI
- Questo è il problema che è capitato anche al nostro sito
- Indagando su come affrontare il problema, mi sono imbattuto in **suggerimenti** – sia di umani che di AI – su come usare fail2ban
  - Ci sono varie considerazioni da fare, tra cui: soluzione **non** ottimale, il file `robots.txt`, l'analisi degli `user-agent` e via dicendo

Esempio pratico

- Impostiamo una *jail* per analizzare il log di Apache
  - **Giochiamo** un po' con i parametri sui tentativi, la finestra e il tempo di ban
- Nel **filtro** impostiamo una regex per identificare una serie di nomi di user-agent
- Problema risolto, forse in maniera un po' **drastica**

```
[apache-AIbadbots]
enabled = true
port = http,https
filter = apache-AIbadbots
logpath = /var/log/apache2/access.log
maxretry = 5
findtime = 600
bantime = 86400
```

```
[Definition]
failregex = ^<HOST> - - .*"(GET|POST|
HEAD).*(ClaudeBot|Bytespider|Amazonbot|
bingbot|AwarioBot|DataForSeoBot|GPTBot)
```

Caso di studio, una soluzione *completa*

- Fail2ban si presta a molti usi diversi, come abbiamo visto, ma la sua forza è quella di poter lavorare in maniera **sinergica** su un server
- Per questa ragione, è importante che i servizi principali conferiscano a fail2ban il maggior **numero** di informazioni per una migliore protezione
- SSH e webserver sono due esempi già visti dell'infrastruttura, ma anche le **applicazioni** possono contribuire
  - Esistono plugin ed estensioni per vari sistemi (esempio Wordpress)
  - Stranamente Mediawiki disponeva solo di una estensione **vecchia** e non più mantenuta

Applicazione completa: Wiki2Ban

- Mediawiki è l'applicativo più **importante** sulla nostra macchina
- Ho quindi sviluppato una estensione per **coprire** questa carenza
- L'estensione usa lo *hook* `AuthManagerLoginAuthenticateAudit` per intercettare e loggare gli accessi negati
- L'estensione **comprende** una *jail* e un *filter* preconfezionati che devono solo essere copiati nelle opportune cartelle

- Progetto open https://github.com/lucamauri/Wiki2Ban
  - Il codice è funzionale, ma **brutto**, per piacere non siate troppo severi nel giudicare
    - Al contrario, chiunque volesse contribuire sarà **ben** accetto

Grazie, ora è il vostro turno

t.me/**lucamauricom**